Installation and Operation of UUCP
4.3BSD Edition

D. A. Nowitz

Ross Green

Computer Systems Research Group
Computer Science Division
Department of Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, CA 94720

ABSTRACT

Uucp is a collection of programs designed to permit communication between UNIX† systems using either dial-up or hardwired communication lines. It is used for file transfers and remote command execution. The first version of the system was designed and implemented by M. E. Lesk (SMM:21).

There have been many changes to the implementation of UUCP since the release of 4.2BSD. Many problems been fixed, and several improvements to provide greater throughput have been incorporated. A number of new features and facilities have been added. These include:

- Improved administration.
- Extended modem support.
- New transfer protocols
- Security enhancements.

The first part of this document gives a detailed description of the use of UUCP. The command descriptions do not describe all the options available; see the manual pages for complete descriptions. The rest of the document indicates the changes that have been made to UUCP, and provides an update on the installation and implementation details. It is for use by an administrator or installer of the system; it is not meant as a user's guide.

Revised May 1986

1. Uucp Implementation Description

Uucp is a batch type operation. Files are created in a spool directory for processing by the uucp demons. For efficiency, the files are separated by type into subdirectories of this directory. The subdirectories will be described in section 9. There are three types of files used for the execution of work. *Data files* contain data for transfer to remote systems. *Work files* contain instructions for file transfers between systems. *Execution files* are instructions for UNIX command executions which involve the resources of one or more systems.

† UNIX is a trademark of Bell Laboratories.

The uucp system consists of ten primary (i.e. invoked by users) and four secondary programs. These programs are summarized in section 9. The three most important primary programs are:

uucp — This program creates work and gathers data files in the spool directories for the transmission of files.

uux — This program creates work files, execute files and gathers data files for the remote execution of UNIX commands.

uusnap — This program provides a snapshot of the current queue including transfers queued and commands to be executed locally.

The three most important secondary programs are:

uucico — This program actually performs the data transmission.

uuxqt — This program executes the execution files for UNIX command execution.

uuclean — This program removes old files from the spool directories.

The next six sections of this paper will describe the operation of each program. The remainder of this paper describes the installation of the system, the security aspects of the system, the files required for execution, and the administration of the system.

2. Uucp - UNIX to UNIX File Copy

The *uucp* command is the user's primary interface with the system. The *uucp* command was designed to look like *cp* to the user. The syntax is

uucp **[** option **]** ... source ... destination

where the source and destination may contain the prefix *system-name!* which indicates the system on which the file or files reside or where they will be copied.

The options interpreted by *uucp* are:

–f — Don't make directories when copying the file. The default is to make the necessary directories.

–C — Copy source files to the spool directory. The default is to use the specified source when the actual transfer takes place.

–g*letter* — Put *letter* in as the grade in the name of the work file. (This can be used to change the order of work for a particular machine.)

–m — Send mail on completion of the work.

–n*user* — Notify *user* on the destination system that a file was sent.

The following options are used primarily for debugging:

–r — Queue the job but do not start *uucico* program.

–s*dir* — Use directory *dir* for the top level spool directory.

–x*num* — *Num* is the level of debugging output desired.

The destination may be a directory name, in which case the file name is taken from the last part of the source's name. The source name may contain special shell characters such as ‘‘*?*[]*’’. If a source argument has a *system-name!* prefix for a remote system, the file name expansion will be done on the remote system. Quote or escape characters that have special meaning to your shell, for example, ‘!’ in csh.

The command

```
uucp *.c usg!/usr/dan
```

will set up the transfer of all files whose names end with ‘‘.c’’ to the ‘‘/usr/dan’’ directory on the ‘‘usg’’ machine.

The source and/or destination names may also contain a *~user* prefix. This translates to the login directory on the specified system. For names with partial path-names, the current directory is prepended to the file name. File names with *../* are not permitted.

The command

uucp usg!˜dan/*.h ˜dan

will set up the transfer of files whose names end with ‘‘.h’’ in dan’s login directory on system ‘‘usg’’ to dan’s local login directory.

For each source file, the program will check the source and destination file-names and the system-part of each to classify the work into one of five types:

[1] Copy source to destination on local system.

[2] Receive files from a remote system.

[3] Send files to a remote system.

[4] Send files from remote system to another remote system.

[5] Receive files from remote system when the source pathname contains special shell characters as mentioned above.

After the work has been set up in the spool directories, the *uucico* program is started to try to contact the other machine to execute the work (unless the –r option was specified).

Type 1

Uucp makes a copy of the file. The *–m* option is not honored in this case.

Type 2

A one line *work file* is created for each file requested and put in the appropriate spool directory with the following fields, each separated by a blank. (All *work files* and *execute files* use a blank as the field separator.)

[1] R

[2] The full path-name of the source or a ˜user/path-name. The *˜user* part will be expanded on the remote system.

[3] The full path-name of the local destination file. If the *˜user* notation is used, it will be immediately expanded to be the login directory for the user.

[4] The user’s login name.

[5] A ‘‘–’’ followed by an option list.

Type 3

For each source file, a *work file* is created. A ‘‘–C’’ option on the *uucp* command will cause the *data file* to be copied into the spool directory and the file to be transmitted from the copy. The fields of each entry are given below.

[1] S

[2] The full-path name of the source file.

[3] The full-path name of the destination or ˜user/file-name.

[4] The user’s login name.

[5] A ‘‘–’’ followed by an option list.

[6] The name of the *data file* in the spool directory.

[7] The file mode bits of the source file in octal print format (e.g. 0666).

[8] The user to notify on the remote system that the transfer has completed.

Type 4 and Type 5

Uucp generates a *uucp* command and sends it to the remote machine; the remote *uucico* executes the *uucp* command.

3. Uux - UNIX To UNIX Execution

The *uux* command is used to set up the execution of a UNIX command where the execution machine and/or some of the files are remote. The syntax of the uux command is

uux **[–] [** option **]** ... command-string

where the command-string is made up of one or more arguments. All special shell characters such as ‘‘<>|*?!’’ must be quoted either by quoting the entire command-string or quoting the character as a separate argument. Within the command-string, the command and file names may contain a *system-name!* prefix. All arguments which do not contain a ‘‘!’’ will not be treated as files. (They will not be copied to the execution machine.) The ‘‘–’’ is used to indicate that the standard input for *command-string* should be inherited from the standard input of the *uux* command. The options, essentially for debugging, are:

–r Don’t start *uucico* or *uuxqt* after queuing the job;

–x*num* Num is the level of debugging output desired.

The command

pr abc | uux – usg!lpr

will set up the output of ‘‘pr abc’’ as standard input to an lpr command to be executed on system ‘‘usg’’.

Uux generates an *execute file* which contains the names of the files required for execution (including standard input), the user’s login name, the destination of the standard output, and the command to be executed. This file is either put in the appropriate spool directory for local execution or sent to the remote system using a generated send command (type 3 above).

For required files which are not on the execution machine, *uux* will generate receive command files (type 2 above). These command-files will be put on the execution machine and executed by the *uucico* program. (This will work only if the local system has permission to put files in the remote spool directory as controlled by the remote ‘‘USERFILE’’.)

The *execute file* will be processed by the *uuxqt* program on the execution machine. It is made up of several lines, each of which contains an identification character and one or more arguments. The order of the lines in the file is not relevant and some of the lines may not be present. Each line is described below.

User Line

U user system

where the *user* and *system* are the requester’s login name and system.

Required File Line

F file-name real-name

where the *file-name* is the generated name of a file for the execute machine and *real-name* is the last part of the actual file name (contains no path information). Zero or more of these lines may be present in the *execute file*. The *uuxqt* program will check for the existence of all required files before the command is executed.

Standard Input Line

I file-name

The standard input is either specified by a ‘‘<’’ in the command-string or inherited from the standard input of the *uux* command if the ‘‘–’’ option is used. If a standard input is not specified, ‘‘/dev/null’’ is used.

Standard Output Line

O file-name system-name

The standard output is specified by a ‘‘>’’ within the command-string. If a standard output is not specified, ‘‘/dev/null’’ is used. (Note – the use of ‘‘>>’’ is not implemented.)

Command Line

C command **[** arguments **]** ...

The arguments are those specified in the command-string. The standard input and standard output will not appear on this line. All *required files* will be moved to the execution directory (a subdirectory of the spool directory) and the UNIX command is executed using the Shell specified in the *uucp.h* header file. In addition, a shell ''PATH'' statement is prepended to the command line.

After execution, the temporary standard output file is copied to or set up to be sent to the proper place.

4. Uusnap - Uucp Queue Snapshot

This program displays a synopsis of the current *uucp* situation. For each site that has work queued or that had an abnormal termination on the last connection, a line summarizing the work to be done is output. The line will indicate how many commands there are to be sent, how many data files have been received and not processed, and how many jobs received from the site there are to be executed. A status message describing the last connection will be included if the connection terminated abnormally.

5. Uucico - Copy In, Copy Out

The *uucico* program will perform the following major functions:

- Scan the spool directory for work.
- Place a call to a remote system.
- Negotiate a line protocol to be used.
- Execute all requests from both systems.
- Log work requests and work completions.

Uucico may be started in several ways;

a) by a system daemon,

b) by one of the *uucp, uux, uuxqt* or *uupoll* programs,

c) directly by the user (this is usually for testing),

d) by a remote system. (The *uucico* program should be specified as the ''shell'' field in the ''/etc/passwd'' file for the ''uucp'' logins.)

When started by method a, b or c, the program is considered to be in *MASTER* mode. In this mode, a connection will be made to a remote system. If started by a remote system (method d), the program is considered to be in *SLAVE* mode.

The *MASTER* mode will operate in one of two ways. If no system name is specified (−s option not specified) the program will scan the spool directory for systems to call. If a system name is specified, that system will be called, and work will only be done for that system.

The *uucico* program is generally started by another program. There are several options used for execution:

−r1 Start the program in *MASTER* mode. This is used when *uucico* is started by a program or ''cron'' shell.

−s*sys* Do work only for system *sys.* If **−s** is specified, a call to the specified system will be made even if there is no work for system *sys* in the spool directory. This is useful for polling systems which do not have the hardware to initiate a connection.

The following options are used primarily for debugging:

−d*dir* Use directory *dir* for the top level spool directory.

−x*num* *Num* is the level of debugging output desired.

The next part of this section will describe the major steps within the *uucico* program.

Scan For Work

The names of the work related files in a spool subdirectory have format

type . system-name grade number

where:

Type is an upper case letter, (*C* - copy command file, *D* - data file, *X* - execute file);

System-name is the remote system;

Grade is a character;

Number is a four digit, padded sequence number.

The file

C.res45n0031

would be a *work file* for a file transfer between the local machine and the ‘‘res45’’ machine.

The scan for work is done by looking through the appropriate spool directory for *work files* (files with prefix ‘‘C.’’). A list is made of all systems to be called. *Uucico* will then call each system and process all *work files*.

Call Remote System

The call is made using information from several files which reside in the uucp system directory (usually /usr/lib/uucp). At the start of the call process, a lock is set to forbid multiple conversations between the same two systems.

The system name is found in the ‘‘L.sys’’ file. The precise format of the ‘‘L.sys’’ file is described in section 10, ‘‘System File Details’’. The information contained for each system is;

[1] system name,

[2] times to call the system (days-of-week and times-of-day),

[3] device or device type to be used for call,

[4] line speed,

[5] phone number if field [3] is *ACU* or the device name (same as field [3]) if not *ACU*,

[6] login information (multiple fields),

The time field is checked against the present time to see if the call should be made.

The *phone number* may contain abbreviations (e.g. mh, py, boston) which get translated into dial sequences using the *L-dialcodes* file.

The *L-devices* file is scanned using fields [3] and [4] from the ‘‘L.sys’’ file to find an available device for the call. The program will try all devices which satisfy [3] and [4] until the call is made or no more devices can be tried. If a device is successfully opened, a lock file is created so that another copy of *uucico* will not try to use it. If the call is complete, the *login information* (field [6] of ‘‘L.sys’’) is used to login.

The conversation between the two *uucico* programs begins with a handshake started by the called, *SLAVE*, system. The *SLAVE* sends a message to let the *MASTER* know it is ready to receive the system identification and conversation sequence number. The response from the *MASTER* is verified by the *SLAVE* and if acceptable, protocol selection begins. The *SLAVE* can also reply with a ‘‘call-back required’’ message in which case, the current conversation is terminated.

Line Protocol Selection

The remote system sends a message

P*proto-list*

where proto-list is a string of characters, each representing a line protocol.

The calling program checks the proto-list for a letter corresponding to an available line protocol and returns a *use-protocol* message. The *use-protocol* message is

U*code*

where code is either a one character protocol letter or *N* which means there is no common protocol.

Work Processing

The initial roles (*MASTER* or *SLAVE*) for the work processing are the mode in which each program starts. (The *MASTER* has been specified by the ‘‘−r1’’ *uucico* option.) The *MASTER* program does a work search similar to the one used in the ‘‘Scan For Work’’ section.

There are five messages used during the work processing, each specified by the first character of the message. They are;

S send a file,

R receive a file,

C copy complete,

X execute a *uucp* command, and

H hangup.

The *MASTER* will send *R*, *S* or *X* messages until all work from the spool directory is complete, at which point an *H* message will be sent. The *SLAVE* will reply with *SY*, *SN*, *RY*, *RN*, *HY*, *HN*, *XY*, *XN*, corresponding to *yes* or *no* for each request.

The send and receive replies are based on permission to access the requested file/directory using the ‘‘USERFILE’’ and read/write permissions of the file/directory. After each file is copied into the spool directory of the receiving system, a copy-complete message is sent by the receiver of the file. The message *CY* will be sent if the file has successfully been moved from the temporary spool file to the actual destination. Otherwise, a *CN* message is sent. (In the case of *CN*, the transferred file will be in a spool subdirectory with a name beginning with ‘‘TM’.) The requests and results are logged on both systems.

The hangup response is determined by the *SLAVE* program by a work scan of its spool directory. If work for the *MASTER*’s system exists in the *SLAVE*’s spool directory, an *HN* message is sent and the programs switch roles. If no work exists, an *HY* response is sent.

Conversation Termination

When a *HY* message is received by the *MASTER* it is echoed back to the *SLAVE* and the protocols are turned off. Each program sends a final ‘‘OO’’ message to the other. The original *SLAVE* program will clean up and terminate. The *MASTER* will proceed to call other systems and process work as long as possible or terminate if a *−s* option was specified.

6. Uuxqt - Uucp Command Execution

The *uuxqt* program is used to execute *execute files* generated by *uux.* The *uuxqt* program may be started by either the *uucico* or *uux* programs. The program scans the appropriate spool directory for *execute files* (prefix ‘‘X.’’). Each one is checked to see if all the required files are available and if so, the command line or send line is executed.

The *execute file* is described in the ‘‘Uux’’ section above.

Command Execution

The execution is accomplished by executing a *sh −c* of the command line after appropriate standard input and standard output have been opened. If a standard output is specified, the program will create a send command or copy the output file as appropriate.

7. Uuclean - Uucp Spool Directory Cleanup

This program is typically started by the daemon, once a day. Its function is to remove files from the spool directories which are more than 3 days old. These are usually files for work which can not be completed.

The options available are:

–d*dir* The directory to be scanned is *dir* .

–m Send mail to the owner of each file being removed. (Note that most files put into the spool directory will be owned by the owner of the uucp programs since the setuid bit will be set on these programs. The mail will therefore most often go to the owner of the uucp programs.)

–n*hours* Change the aging time from 72 hours to *hours* hours.

–p*pre* Examine files with prefix *pre* for deletion. (Up to 10 file prefixes may be specified.)

–x*num* This is the level of debugging output desired.

8. Changes to the UUCP Implementation

The demands placed on UUCP networking and new technology have prompted several changes and improvements to the UUCP software. Such things as low cost, autodial, autoanswer, high speed modems, and the availability of X.25 and TCP/IP as carriers, have encouraged new facilities to be developed for UUCP.

The following areas have been changed between the 4.2 and 4.3 BSD releases:

- General fixes and performance improvements.
- Administration control facilities.
- Modem and autodialer support has been extended.
- New protocols for different transport media.
- Security enhancements.

Fixes and performance improvements.

These include many fixes related to portability and general improvements as provided by the USENET community. In particular, the *sitename* truncation length has been extended to 14 characters from the original 7. This makes it compatible with the current System V version of UUCP.

An effort has been made to improve the overall performance of the UUCP system by organizing its workload in a more sensible way. For example the program *uucico* will not resend files it has already sent when the files are specified in one ‘‘C.’’ file.

Administration and control facilities.

There is a new program, *uuq*, to give more descriptive information on status of jobs in the UUCP spool queue. It also allows users to delete requests that are still in the queue.

In the past, on large UUCP sites, the spool directory could grow large with many files within the ‘‘/usr/spool/uucp’’ directory. To help the UUCP administrator control the system, a number of subdirectories have been created to ease this congestion.

The system status ‘‘STST’’ files are kept in a subdirectory.

Corrupted ‘‘C.’’ and ‘‘X.’’ files that could not be processed are placed in the ‘‘CORRUPT’’ subdirectory, instead of terminating the connection.

Lock files may be kept in a subdirectory, ‘‘LCK’’, if desired.

If an ‘‘X.’’ request fails, the notification is returned to the originator of the request, not to ‘‘uucp’’ on the previous system.

There is a new *system* file, ‘‘L.aliases’’, that may be used when a site changes its name. Most of the utilities check ‘‘L.aliases’’ for correct mapping.

Modem and autodialer support

In a short period of time, there has been an increase in the transfer rates and capabilities that are being provided with modern modems. Most modems allow several combinations of baud rate, and provide autodial and autoanswer facilities as well.

Most sites will have but a few modems; they are therefore a precious resource, and an effort has been made to use them to maximum potential. The *uucico* program now has code to place and receive calls on the same device, if that modem has both autodial and autoanswer support. There is a new dialing facility *acucntrl* that has been designed to handle some of the changes in modem technology. There are a number of new modems and autodialers that are now supported. Here is a list of some of the new devices:

Racal-Vadic 212
Racal-Vadic 811 dialer with 831 adapter
Racal-Vadic 820 dialer with 831 adapter
Racal-Vadic MACS 811 dialer with 831 adapter
Racal-Vadic MACS 820 dialer with 831 adapter
DEC DF112
Novation
Penril
Hayes 2400 Smartmodem
Concord Data Systems CDS 224
AT&T 2224 2400 baud modem

New protocols for different transportation mediums

The UUCP software has had provision for different protocols to be used for sending and receiving data, but originally only one was implemented and this is the one that is largely used throughout the UUCP community. It has a maximum throughput of around 9000 baud, regardless of the physical medium. The use of checksums and short data packets are of little use when the protocol is layered above another reliable protocol such as TCP or X.25. The UUCP system did not utilize LAN's and high speed carriers well. Two new protocols have been added to provide for this. The protocols now available to UUCP are:

't' protocol, optimized for use on TCP/IP carriers.
'f' protocol, optimized for use on X.25 PAD carriers.
'g' protocol, standard UUCP protocol used for dialup or hardwired lines.

The existing 'g' protocol code has been cleaned up in this version. The 't' protocol is essentially the 'g' protocol except that the channel is assumed to be free from errors. As such, no checksums are used and files are transferred without packetizing. The 'f' protocol relies on the flow control of the data stream. It is meant for use over links that can be guaranteed to be free from errors, specifically X.25/PAD links. The checksum is calculated over whole files only. If a transport fails the receiver can request retransmissions. This protocol uses a 7-bit data path only, so it may be used on carriers that do not handle 8-bit data paths transparently.

Changes to *uucico*

Uucico used to attempt to place a call using every dialer on the system. Since this could take a long time at large sites, the defined constant TRYCALLS now limits the number of attempts.

You can specify a maximum grade to send either on the command line using -gX option or by specifying the time to call in the ''L.sys'' file as follows:

Any/C,Evening

This will only send grade C or higher transfers, usually mail, during the day and will send any grades in the evening.

The code for the closing hangup sequence has been fixed.

Some new options were added to *uucico*. These include:

-R This flag reverses *uucico*'s initial role (lets the remote system be master first rather than slave).

-L *uucico* will only call ‘‘local’’ sites. Local sites are those sites having one of **LOCAL**, **TCP** or **DIR** in the CALLER field of ‘‘L.sys’’.

If ‘‘/etc/nologin’’ is present, usually created by *shutdown*(8), *uucico* and *uuxqt* will exit gracefully, instead of getting killed off when the system goes down.

Uucico now uses an exponential back off on the retry time if consecutive calls fail instead of always waiting 5 minutes. The default may be overridden by adding ";*time*" to the time field in ‘‘L.sys’’.

ucbvax Any;2

The preceding fragment indicates that a default retry time of 2 minutes will be used.

If *uucico* receives a SIGFPE while running, it will toggle debugging.

It will not send files to a remote system returning an out of temporary file space error.

More functionality has been added to the expect/send sequences. The ABORT command was added to the expect/send sequence so it does not have to wait for timeout if cannot get through a port selector. You can specify a time for the expect/send sequences with ˜ to override the default timeout. The expect/send sequences now allow escape sequences to specify characters that could not be specified before.

The time field in the ‘‘L.sys’’ file now handles ‘‘Evening’’, ‘‘Night’’, and ‘‘NonPeak’’ in addition to Any, Mo, Tu, We, Th, Fr, Sa, Su, and Wk.

The file *L-devices* now handles ‘‘chat’’ scripts, to help get through local port selectors and smart modems. This helps keep ‘‘L.sys’’ readable while using the increased functionality.

For compatibility with the System V UUCP, the following changes were made in the date fields of ‘‘L.sys’’:

‘|’ changed to ‘,’ (‘|’ is supported, but not encouraged)
‘,’ changed to ‘;’ (to allow ‘,’ to be the date separator)

For Honey DanBer compatibility, *uucico* now passes the maximum grade to the remote system as ‘‘-vgrade=X’’ instead of the old -pX

Support has been added for GTE’s PC Pursuit service. It is mainly the handling of the call back method they use.

Users must now have read access to ‘‘L.sys’’ in order to run *uucico* with debugging turned on.

9. The UUCP system.

Names

The name of a site is important since it provides a means of identifying a machine, and consequently, that machine’s users. There are two kinds of names used within the UUCP system; *loginname*s and *sitename*s.

It is important that the *loginname*s used by a remote machine to call into a local machine is not the same as that of a normal user of the local machine. Each *loginname* corresponds with a line in ‘‘/etc/passwd’’. It is the administrator’s decision whether each remote site should use the same login name or different ones.

Each machine in a UUCP network is given a unique *sitename*. The *sitename* identifies the calling machine to the called machine. A *sitename* can be up to 14 characters in length. It is useful to have a *sitename* that is unique in the first 7 characters, to be compatible with earlier implementations of UUCP. It is desirable that the *sitename* will convey this uniqueness and perhaps a real world identity to the rest of the network.

The UUCP system organization.

There are several directories that are used by the UUCP system as distributed. These are:

src	(/usr/src/usr.bin/uucp) This directory contains the source files for the UUCP system.
system	(/usr/lib/uucp) This directory contains the system binaries and system control files.
spool	(/usr/spool/uucp) This spool directory is used to store transfer requests and data.
command	(/usr/bin) This directory contains the user-level programs.

The system directory

The following files are required for execution, and should reside in the *system* directory, /usr/lib/uucp.

L-devices	Contains entries for all devices that are to be used by UUCP.
L-dialcodes	Contains dialing abbreviations.
L.aliases	Contains site name aliases.
L.cmds	Contains the list of commands that can be used by a remote site.
L.sys	Contains site connection information for each system that can be called.
SEQF	The sequence numbering and check file.
USERFILE	Remote system access rights.
acucntrl	The program used to control calling remote systems.
uucico	The actual transfer program.
uuclean	A utility to clean up after UUCP.
uuxqt	Executes commands received from remote systems.

The command directory

The *command* directory, /usr/bin, contains the following user available commands:

uucp	Spools a UNIX to UNIX file-copy request.
uux	Spools a request for remote execution.
uusend	Provides a facility to transfer binary files using mail.
uuencode	Binary file encoder (for *uusend*)
uudecode	Binary file decoder (for *uusend*)
uulog	Reports from log files.
uusnap	Provides a snapshot of *uucp* activity.
uupoll	Polls a remote system.
uuname	Prints a list of known remote UUCP hosts.
uuq	Reports information from the UUCP spool queue.

The spool directory

The *spool* directory, /usr/spool/uucp, contains the following files and directories:

C.	A directory for command (‘‘C.’’) files.
D.	A directory for data (‘‘D.’’) files.
X.	A directory for command execution (‘‘X.’’) files.
D.*machine*	A directory for local ‘‘D.’’ files.
D.*machine*X	A directory for local ‘‘X.’’ files.
CORRUPT	A directory for corrupted ‘‘C.’’ and ‘‘X.’’ files.

ERRLOG	A file where internal error messages are collected.
LCK	A directory for device and site lock files (optional).
LOG	A directory for individual site LOGFILE's (optional).
LOGFILE	The log file of UUCP activity (optional).
STST	A directory for per site system status files (''STST'').
SYSLOG	The log file of UUCP file transfers.
TM.	A directory for temporary (''TM.'') files.

This version has broken the spool directory into the above list of directories leaving only a few system files in the top level directory. The logs from each system may be kept together or in separate files in a subdirectory (LOG). This decision is made when the system is compiled.

There is an additional directory, /usr/spool/uucppublic, that is used as a general public access directory for UUCP. It is not used by UUCP directly but it is normally the home directory for the UUCP system owner. Most importantly this directory is owned by uucp, and the access permissions are 0777. This usually guarantees a place that files can be copied to, and retrieved from, on any site.

10. System file details.

The *system* files in the ''/usr/lib/uucp'' directory can contain comments, by putting a '#' as the first character on a line. Lines may be continued by placing a '\' as the last character of a line. This is helpful in making the files more readable.

L-devices

This file contains entries for the call-unit devices and hardwired connections which are to be used by UUCP. The special device files are assumed to be in the /dev directory.

The format for each entry is:

Type Device Useful Class Dialer [*Chat* ...]

where;

Type Is the type of connection to use.

ACU	Indicates that a dialing device is used.
LOCAL	Indicates an **ACU** with a ''preferred'' connection.
DIR	Indicates that a direct connection is used.
DK	Indicates that an AT&T Datakit is used.
MICOM	Indicates that a Micom terminal switch is used.
PAD	Indicates that a X.25 PAD connection is used.
PCP	Indicates that GTE Telenet PC Pursuit is used.
SYTEK	Indicates that a Sytek high-speed dedicated modem port is used.
TCP	Indicates that a TCP/IP connection is used.

Device Is the entry in ''/dev'' corresponding to a real device. UUCP should be able to access this device.

Call_Unit Is the device for dialing if different from the device used for the data transfer. This field must contain a place holder if unused (such as ''unused'').

Class is the line baud rate for dialers and direct lines or the port number for network connections.

Dialer is either **direct**, or from the list of available dialers. The list of available dialers includes:

DF02	DEC DF02 or DF03 modems.

DF112 DEC DF112 modems. Use a *Dialer* field of **DF112T** to use tone dialing, or **DF112P** for pulse dialing.

att AT&T 2224 2400 baud modem.

cds224 Concord Data Systems 224 2400 baud modem.

dn11 DEC DN11 UNIBUS dialer.

hayes Hayes Smartmodem 1200 and compatible autodialing modems. Use a *Dialer* field of **hayestone** to use tone dialing, or **hayespulse** for pulse dialing. It is also permissible to include the letters 'T' and 'P' in the phone number (in ''L.sys'') to change to tone or pulse midway through dialing. (Note that a leading 'T' or 'P' will be interpreted as a dialcode!)

hayes2400 Hayes Smartmodem 2400 and compatible modems. Use a *Dialer* field of **hayes2400tone** to use tone dialing, or **hayes2400pulse** for pulse dialing.

novation Novation ''Smart Cat'' autodialing modem.

penril Penril Corp ''Hayes compatible'' modems.

rvmacs Racal-Vadic 820 dialer with 831 adapter in a MACS configuration.

va212 Racal-Vadic 212 autodialing modem.

va811s Racal-Vadic 811s dialer with 831 adapter.

va820 Racal-Vadic 820 dialer with 831 adapter.

vadic Racal-Vadic 3450 and 3451 series autodialing modems.

ventel Ventel 212+ autodialing modem.

vmacs Racal-Vadic 811 dialer with 831 adapter in a MACS configuration.

Chat is a send/expect sequence that can be used to talk through dataswitches, or issue special commands to a device such as a modem. The syntax is identical to that of the Expect/Send script of ''L.sys'' and will be described later. The difference is that, the *L-devices* script is used before the connection is made, while the ''L.sys'' script is used after.

L-dialcodes

This file contains entries with location abbreviations used in the ''L.sys'' file (e.g. py, mh, boston). The entry format is:

abb dial-seq

where;

abb is the abbreviation,

dial-seq is the dial sequence to call that location.

The line

py 165–

would be set up so that entry py7777 in ''L.sys'' would send 165–7777 to the dial-unit.

L.aliases.

The *L.aliases* file provides a mapping facility for *sitename*s. This facility is useful when a *sitename* is changed temporarily, or until a permanent change becomes widely known by the users of the net. The format of the file is:

real_name alias_name

The ''L.aliases'' file may be used to map hosts with longer names in ''L.sys'' to 7 character names that some hosts send. This provides a mechanism to handle those sites, entries should be:

fullname 7-char-name

L.cmds

The L.cmds file contains a list of commands that are permitted for remote execution with *uux*. The commands are listed one per line. Most sites L.cmds will be something like:

```
rmail
rnews
ruusend
```

A line of the form:

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

can be used to set a search path.

L.sys

Each entry in this file represents one system that communicates with the local system and has the form:

Sitename Times Caller Class Device [*Expect Send*]....

Sitename is the name of the remote system. Every machine with which this system communicates via UUCP should be listed, regardless of who calls whom. Systems not listed in ‘‘L.sys’’ will not be permitted a connection.

Times is a comma-separated list of the times of the day and week that calls are permitted to this site. This can be used to restrict long distance telephone calls to those times when rates are lower. List items are constructed as:

*keyword*hhmm**-**hhmm**/***grade***;***retry_time*

Keyword is required, and must be one of:

Any Any time, any day of the week.

Wk Any weekday. In addition, **Mo, Tu, We, Th, Fr, Sa,** and **Su** can be used.

Evening When evening telephone rates are in effect, from 1700 to 0800 Monday through Friday, and all day Saturday and Sunday. **Evening** is the same as **Wk1700-0800,Sa,Su**.

Night When nighttime telephone rates are in effect, from 2300 to 0800 Monday through Friday, all day Saturday, and from 2300 to 1700 Sunday. **Night** is the same as **Any2300-0800,Sa,Su0800-1700**.

NonPeak This is a slight modification of **Evening**. It matches when the USA X.25 carriers have their lower rate period. This is 1800 to 0700 Monday through Friday, and all day Saturday and Sunday. **NonPeak** is the same as **Any1800-0700,Sa,Su**.

Never Calling this site is forbidden or impossible. This is intended for polled connections, where the remote system calls into the local machine periodically.

The optional *hhmm-hhmm* subfield provides a time range that modifies the keyword. *hhmm* refers to *hours* and *minutes* in 24-hour time (from 0000 to 2359). The time range is permitted to "wrap" around midnight, and will behave in the obvious way. It is invalid to follow the **Evening**, **NonPeak**, and **Night** keywords with a time range.

The *grade* subfield is optional; if present, it is composed of a ‘/’ (slash) and single character denoting the *grade* of the connection. Grades are in the range [**0-9A-Za-z**]. This specifies that only requests of grade *grade* or better will be transferred during this time. (The grade of a request or job is specified when it is queued by *uucp* or *uux*). By convention, mail is sent at grade **C**, news is sent at grade **d**, and uucp copies are sent at grade **n**. Unfortunately, some sites do not follow these conventions consistently.

The *retry_time* subfield is optional; it must be preceded by a ';' (semicolon) and specifies the minimum time, in minutes, before a failed connection will be tried again. By default, the retry time starts at 10 minutes and gradually increases at each failure, until after 26 tries *uucico* gives up completely (MAX RETRIES). If the retry time is too small, *uucico* may run into MAX RETRIES too soon.

Caller is the type of device used. It may be one of the following:

ACU DIR LOCAL MICOM PAD PCP SYTEK TCP

The descriptions are the same as listed in ''L-devices'' above. If several alternate ports or network connections should be tried, use multiple ''L.sys'' entries.

Class is usually the speed (baud) of the device, typically 300, 1200, or 2400 for ACU devices and 9600 for direct lines. Valid values are device dependent, and are specified in the ''L-devices'' file.

On some devices, the speed may be preceded by a non-numeric prefix. This is used in ''L-devices'' to distinguish among devices that have identical *Caller* and baud, but yet are distinctly different. For example, 1200 could refer to all Bell 212-compatible modems, V1200 to Racal-Vadic modems, and C1200 to CCITT modems, all at 1200 baud.

On TCP connections, *Class* is the port number (an integer) or a port name from ''/etc/services'' that is used to make the connection. For standard Berkeley TCP/IP, UUCP normally uses port number 540.

Device varies based on the *Caller* field. For **ACU** devices, this is the phone number to dial. The number may include: digits **0** through **9**; **#** and ***** for dialing those symbols on tone telephone lines; **-** (hyphen) to pause for a moment, typically two to four seconds; **=** (equal sign) to wait for a second dial tone (implemented as a pause on many modems). Other characters are modem dependent; generally standard telephone punctuation characters (such as the slash and parentheses) are ignored, although *uucico* does not guarantee this.

The phone number can be preceded by an alphabetic string; the string is indexed and converted through the ''L-dialcodes'' file.

For **DIR** devices, the *Device* field contains the name of the device in /dev that is used to make the connection. There must be a corresponding line in ''L-devices'' with identical *Caller*, *Class*, and *Device* fields.

For **TCP** and other network devices, *Device* holds the network name for establishing a connection to the remote system, which may be different from its UUCP name.

The *Expect* and *Send* refer to an arbitrarily long set of strings that alternately specify what to *expect* and what to *send* to login to the remote system once a physical connection has been established. A complete set of expect/send strings is referred to as an ''*expect*/*send script*''. The same syntax is used in the *L–devices* file to interact with the dialer prior to making a connection; there it is referred to as a *chat script*. The complete format for one *expect/send* pair is:

expect˜timeout–failsend–expect˜timeout send

Expect, *failsend*, and *send* are character strings. *Expect* is compared against incoming text from the remote host; *send* is sent back when *expect* is matched. By default, the *send* is followed by a '\r' (carriage return). If the *expect* string is not matched within *timeout* seconds (default 45), then it is assumed that the match failed. The '*expect-failsend-expect*' notation provides a limited loop mechanism; if the first *expect* string fails to match, then the *failsend* string between the hyphens is transmitted, and *uucico* waits for the second *expect* string. This can be repeated indefinitely. When the last *expect* string fails, *uucico* hangs up and logs that the connection failed.

The timeout can (optionally) be specified by appending the parameter '˜*nn*' to the expect string, when *nn* is the timeout time in seconds.

Backslash escapes that may be embedded in the *expect* or *send* strings include:

\b	Generate a 3/10 second BREAK.
\b*n*	Where *n* is a single-digit number; generate an *n*/10 second BREAK.
\c	Suppress the \r at the end of a *send* string.
\d	Delay; pause for 1 second. (*Send* only.)
\r	Carriage Return.
\s	Space.
\n	Newline.
\xxx	Where *xxx* is an octal constant; denotes the corresponding ASCII character.

As a special case, an empty pair of double-quotes **" "** in the *expect* string is interpreted as ‘‘expect nothing’’; that is, transmit the *send* string regardless of what is received. Empty double-quotes in the *send* string cause a lone ‘\r’ (carriage return) to be sent.

One of the following keywords may be substituted for the *send* string:

BREAK	Generate a 3/10 second BREAK
BREAK*n*	Generate an *n*/10 second BREAK
CR	Send a Carriage Return (same as " ").
EOT	Send an End-Of-Transmission character, ASCII \004. Note that this will cause most hosts to hang up.
NL	Send a Newline.
PAUSE	Pause for 3 seconds.
PAUSE*n*	Pause for *n* seconds.
P_ODD	Use odd parity on future send strings.
P_ONE	Use parity one on future send strings.
P_EVEN	Use even parity on future send strings. (Default)
P_ZERO	Use parity zero on future send strings.

Finally, if the *expect* string consists of the keyword **ABORT**, the following string is used to arm an abort trap. If that string is subsequently received any time prior to the completion of the entire *expect/send* script, then *uucico* will abort, just as if the script had timed out. This is useful for trapping error messages from port selectors or front-end processors such as ‘‘Host Unavailable’’ or ‘‘System is Down.’’

An example expect/send sequence might look something like this:

```
" " \d\r CLASS HOST ABORT Down GO \d\r ogin:~30-\b-ogin: uucp word: password
```

First, *uucico* will expect nothing, wait 1 second (\d), and then send a carriage return. The next expected message is ‘‘CLASS’’, in response to which *uucico* sends ‘‘HOST’’. From then on, if it sees the word ‘‘Down’’ before finishing logging in, it will hang up immediately. In the mean time, it looks for ‘‘GO’’. After this is received, it delays 1 second and then sends a CR. *Uucico* resets the timeout to 30 seconds while whating to receive ‘‘ogin:’’. If there is no response, a break will be sent and the program will wait for 45 seconds for ‘‘ogin:’’ again. When this is received, ‘‘uucp’’ will be sent. The sequence ends by waiting for ‘‘word:’’ and responding with ‘‘password’’. At this point, UUCP has completed the login and continues with the protocol for establishing the connection..

USERFILE

This file contains user accessibility information. It specifies the file system directory trees that are accessible to local users and to remote systems via UUCP

Each line in ‘‘USERFILE’’ is of the form:

[*loginname*]**,**[*sitename*] [**c**] *pathname* [*pathname*] [*pathname*]

The first two items are separated by a comma; any number of spaces or tabs may separate the remaining items.

The *loginname* is a user name (from ''/etc/passwd'') on the local machine.

The *sitename* is the name of a remote machine. This is the same name used in ''L.sys''.

The *c* denotes the optional *callback* field. If a **c** appears here, a remote machine that calls in will be told that callback is requested, and the conversation will be terminated. The local system will then immediately call the remote host back.

The *pathname* is a pathname prefix that is permissible for this *loginname* and/or *sitename*.

When *uucico* runs in master role or *uucp* or *uux* are run by local users, the permitted pathnames are those on the first line with a *loginname* that matches the name of the user who executed the command. If no such line exists, then the first line with a null (missing) *loginname* field is used. (Beware: *uucico* is often run by the superuser or the UUCP administrator through *cron*.

When *uucico* runs in slave role, the permitted pathnames are those on the first line with a *sitename* field that matches the hostname of the remote machine. If no such line exists, then the first line with a null (missing) *sitename* field is used.

Uuxqt works differently; it knows neither a login name nor a hostname. It accepts the pathnames on the first line that has a null *sitename* field. (This is the same line that is used by *uucico* when it cannot match the remote machine's hostname.)

A line with both *loginname* and *sitename* null, for example

, /usr/spool/uucppublic

can be used to conveniently specify the paths for both ''no match'' cases if lines earlier in ''USERFILE'' did not define them.

11. Installing the UUCP system.

There are several source modifications that may be required before the system programs are compiled.

Two files which may require modification, the ''Makefile'' file and the ''uucp.h'' file. The following paragraphs describe some of the options available at build time.

Uucp.h modifications

The installer of UUCP may wish to change some of the defines in ''uucp.h''. Some of the interesting defines are mentioned below.

if **DIALINOUT** is defined then *acucntrl* will allow modems to be used in both directions.

If **DONTCOPY** is defined in ''uucp.h'', *uucp* will not make a copy of the source file by default.

if **LOCKDIR** is defined then lock files will be stored in the ''/usr/spool/uucp/LCK'' directory.

If **LOGBYSITE** is defined, *uucp* logging is done with a log file per site, instead of one LOGFILE.

If **NOSTRANGERS** is defined in ''uucp.h'', the remote site must be in your ''L.sys'' or the call will be rejected.

Makefile modification

There are several *make* variable definitions which may need modification.

LIBDIR   the directory where low level binaries, site information, and dialing information are stored

BIN	The directory in which the user utilities reside.
PUBDIR	A directory where files can almost always be sent. This should be UUCP's home directory and writable by everyone.
SPOOL	The top level spool directory.
XQTDIR	The directory where temporary files will be stored by *uuxqt*.
CORRUPT	The directory where corrupted ''C.'' and ''D.'' files end up.
AUDIT	The directory where debugging traces are stored by *uucico* when debugging is remotely enabled or enabled by a signal.
LCK	The directory where lock files are kept. *Tip*(1) and other programs may need to be modified if this is changed as the lock files are shared.
LOG	The directory where the log files are placed if ''LOGBYSITE'' is defined in ''uucp.h''.
STST	The directory where the remote system status files (''STST'') are stored.
HOSTNAME	The machine's name.

Building the system

The command

```
make
```

will compile the entire system.

The command

```
make mkdirs
```

will build all the directories needed for the system, giving them appropriate owners and permissions.

The command

```
make install
```

will install the commands in the correct directories, setting ownership and permissions.

12. Connecting new systems to the network.

When first connecting a new machine to a UUCP network, it is advisable to try and establish a connection with *tip* or *cu* first. The administrator should then be aware of any special facilities that are going to be required, things like; What lines and modems are to be used? Is the connection through different hardware and carriers? Does the remote system care about parity? What speed lines are being used and do they cycle through several speeds? Is there a line switch front end that will require special Chat dialogue in ''L.sys'' ?

Once a *login* connection can be completed the administrator should have enough information to allow the correct setup of the *system* files in /usr/lib/uucp.

The UUCP administrator should then negotiate with the remote site's UUCP administrator as to who will do polling and when. Both administrators must set up the relevant accounts and passwords. The UUCP administrator should decide on what permissions and security precautions are to be observed. Testing time and facilities will need to be arranged to complete initial connection testing between the systems.

13. Security

The uucp system, left unrestricted, will let any outside user execute any commands and copy any files that are accessible to the uucp login user. It is up to the individual sites to be aware of this and apply the protections that they feel are necessary.

There are several security features available aside from the normal file mode protections. These must be set up by the installer of the *uucp* system.

- The login for uucp does not get a standard shell. Instead, the *uucico* program is started. Therefore, the only work that can be done is through *uucico* .
- A path check is done on file names that are to be sent or received. The ‘‘USERFILE’’ supplies the information for these checks. The ‘‘USERFILE’’ can also be set up to require call-back for certain login-ids. (See the description of ‘‘USERFILE’’ above.)
- A conversation sequence count can be set up so that the called system can be more confident that the caller is who he says he is.
- The *uuxqt* program comes with a list of commands that it will execute. A ‘‘PATH’’ shell statement is prepended to the command line as specified in the *uuxqt* program. The installer may modify the list or remove the restrictions as desired.
- The ‘‘L.sys’’ file should be owned by uucp and only readable by uucp to protect the phone numbers and login information for remote sites. (Programs uucp, uucico, uux, uuxqt should be also owned by uucp and have the set user id bit set.)

14. Administration

This section indicates some events and files which must be administered for the *uucp* system. Some administration can be accomplished by *shell files* which can be initiated by *cron* (8). Others will require manual intervention.

SQFILE − sequence check file

This file is set up in the *library* directory and contains an entry for each remote system with which you agree to perform conversation sequence checks. The initial entry is just the system name of the remote system. The first conversation will add two items to the line, the conversation count, and the date/time of the most resent conversation. These items will be updated with each conversation. If a sequence check fails, which could indicate that an unauthorized connection has been attempted, the entry will have to be adjusted.

TM − temporary data files

These files are created in the *spool* directory while files are being copied from a remote machine. Their names have the form

TM.pid.ddd

where *pid* is a process-id and *ddd* is a sequential three digit number starting at zero for each invocation of *uucico* and incremented for each file received. After the entire remote file is received, the *TM* file is moved to the requested destination. If processing is abnormally terminated or the move fails, the file will remain in the spool directory.

The leftover files should be periodically removed; the *uuclean* program is useful in this regard. The command

uuclean −pTM

will remove all *TM* files older than three days.

STST − system status files

These files are created in the spool directory by the *uucico* program. They contain information of failures such as login, dialup or sequence check and will contain a *TALKING* status when two machines are conversing. The file name is the remote system name in the ‘‘STST’’ directory.

For ordinary failures (dialup, login), the file will prevent repeated tries too frequently. For sequence check failures, the file must be removed before any future attempts to converse with that remote system.

If the file is left due to an aborted run, it may contain a *TALKING* status. In this case, the file must be removed before a conversation is attempted.

LCK − lock files

Lock files are created for each device in use (e.g. automatic calling unit) and each system conversing. This prevents duplicate conversations and multiple attempts to use the same devices. The form of the lock file name is

LCK..str

where *str* is either a device or system name. The files may be left in the spool directory if runs abort. They will be ignored (reused) after a time of about 24 hours. When runs abort and calls are desired before the time limit expires, the lock files should be removed.

Shell Files

The *uucp* program will spool work and attempt to start the *uucico* program, but the starting of *uucico* will sometimes fail. (No devices available, login failures etc.). Therefore, the *uucico* program should be periodically started. The command to start *uucico* can be put in a ''shell'' file and started by *cron* on an hourly basis. The file could contain the command:

```
uucico −r1
```

Note that the ''−r1'' option is required to start the *uucico* program in MASTER mode.

Another shell file may be set up on a daily basis to remove *TM*, *ST* and *LCK* files and *C.* or *D.* files for work which can not be accomplished for reasons like bad phone number, login changes etc. A shell file containing commands like

```
uuclean −pTM −pC. −pD.
uuclean −pST −pLCK −n12
```

can be used. Note the ''−n12'' option causes the *ST* and *LCK* files older than 12 hours to be deleted. The absence of the ''−n'' option will use a three day time limit.

A daily or weekly shell should also be created to remove or save old *LOGFILE*s. One can use a command like

```
mv spool/LOGFILE spool/o.LOGFILE
```

Login Entry

One or more logins should be set up for *uucp*. Each of the ''/etc/passwd'' entries should have the *uucico* as the shell to be executed. The login directory is normally ''/usr/spool/uucppublic''. The various logins are used in conjunction with the ''USERFILE'' to restrict file access. Specifying the *shell* argument limits the login to the use of UUCP (*uucico*) only.

File Modes

It is suggested that the owner and file modes of various programs and files be set as follows.

The programs *uucp*, *uux*, *uucico* and *uuxqt* should be owned by the *uucp* login with the ''setuid'' bit set and only execute permissions (e.g. mode 04111). This will prevent outsiders from modifying the programs to get at a standard *shell* for the *uucp* logins.

''L.sys'', ''SQFILE'', and the ''USERFILE'' which are put in the *program* directory should be owned by the *uucp* login and set so that they can only be read by the *uucp* login and are writable by no one.