Uucp Implementation Description

D. A. Nowitz*

ABSTRACT

Uucp is a series of programs designed to permit communication between UNIX systems using either dial-up or hardwired communication lines. This document gives a detailed implementation description of the current (second) implementation of uucp.

This document is for use by an administrator/installer of the system. It is not meant as a user's guide.

October 31, 1978

* On internship from Department 9444.

Uucp Implementation Description

D. A. Nowitz*

Introduction

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. This paper describes the current (second) implementation of the system.

Uucp is a batch type operation. Files are created in a spool directory for processing by the uucp demons. There are three types of files used for the execution of work. *Data files* contain data for transfer to remote systems. *Work files* contain directions for file transfers between systems. *Execution files* are directions for UNIX command executions which involve the resources of one or more systems.

The uucp system consists of four primary and two secondary programs. The primary programs are:

uucp	This program creates work and gathers data files in the spool directory for the transmission of files.
uux	This program creates work files, execute files and gathers data files for the remote execution of UNIX commands.
uucico	This program executes the work files for data transmission.
uuxqt	This program executes the execution files for UNIX command execution.

The secondary programs are:

uulog	This program updates the log file with new entries and reports on the status of uucp requests.
uuclean	This program removes old files from the spool directory.

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

1. 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:

−d	Make directories when necessary for copying the file.
−c	Don't copy source files to the spool directory, but 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.

The following options are used primarily for debugging:

---

* On internship from Department 9444.

Uucp is a series of programs designed to permit communication between UNIX†

†UNIX is a Trademark of Bell Laboratories.

E. Lesk.[1]

1 M. E. Lesk and A. S. Cohen, UNIX Software Distribution by Communication Link, TM-77-1274-3, TM-77-8234-5.

−r Queue the job but do not start *uucico* program.
−s*dir* Use directory *dir* for the 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.

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 other systems.
[3]Send files to a remote systems.
[4]Send files from remote systems to another remote system.
[5]Receive files from remote systems when the source contains special shell characters as mentioned above.

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

Type 1

A *cp* command is used to do the work. The *−d* and the *−m* options are not honored in this case.

Type 2

A one line *work file* is created for each file requested and put in the 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 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. (Only the −m and −d options will appear in this list.)

Type 3

For each source file, a *work file* is created and the source file is copied into a *data file* in the spool directory. (A "−c" option on the *uucp* command will prevent the *data file* from being made.) In this case, the file will be transmitted from the indicated source.) 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).

Type 4 and Type 5
Uucp generates a *uucp* command and sends it to the remote machine; the remote *uucico* executes the *uucp* command.

2. 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 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 as specified in the *uuxqt* program.

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

3. 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 *uucico* 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 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 the spool directory 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 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 program directory. 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 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,
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 the spool directory 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 the spool directory. If work for the remote 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.

4. 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 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.

5. Uulog - Uucp Log Inquiry

The *uucp* programs create individual log files for each program invocation. Periodically, *uulog* may be executed to prepend these files to the system logfile. This method of logging was chosen to minimize file locking of the logfile during program execution.

The *uulog* program merges the individual log files and outputs specified log entries. The output request is specified by the use of the following options:

−s*sys*	Print entries where *sys* is the remote system name;
−u*user*	Print entries for user *user.*

The intersection of lines satisfying the two options is output. A null *sys* or *user* means all system names or users respectively.

6. 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 directory 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.

7. Security

The uucp system, left unrestricted, will let any outside user execute any commands and copy in/out any file which is readable/writable by 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 ‘‘Files required for execution’’ section for the file description.)
- 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 specifed 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 have mode 0400 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 setuid bit set.)

8. Uucp Installation

There are several source modifications that may be required before the system programs are compiled. These relate to the directories used during compilation, the directories used during execution, and the local *uucp system-name.*

The four directories are:

lib	(/usr/src/cmd/uucp) This directory contains the source files for generating the *uucp* system.
program	(/usr/lib/uucp) This is the directory used for the executable system programs and the system files.
spool	(/usr/spool/uucp) This is the spool directory used during *uucp* execution.
xqtdir	(/usr/spool/uucp/.XQTDIR) This directory is used during execution of *execute files* .

The names given in parentheses above are the default values for the directories. The italicized named *lib, program, xqtdir,* and *spool* will be used in the following text to represent the appropriate directory names.

There are two files which may require modification, the *makefile* file and the *uucp.h* file. The following paragraphs describe the modifications. The modes of *spool* and *xqtdir* should be made ‘‘0777’’.

Uucp.h modification

Change the *program* and the *spool* names from the default values to the directory names to be used on the local system using global edit commands.

Change the *define* value for *MYNAME* to be the local *uucp* system-name.

makefile modification

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

INSDIR	This is the *program* directory (e.g. INSDIR=/usr/lib/uucp). This parameter is used if ‘‘make cp’’ is used after the programs are compiled.
IOCTL	This is required to be set if an appropriate *ioctl* interface subroutine does not exist in the standard ‘‘C’’ library; the statement ‘‘IOCTL=ioctl.o’’ is required in this case.
PKON	The statement ‘‘PKON=pkon.o’’ is required if the packet driver is not in the kernel.

Compile the system The command

```
make
```

will compile the entire system. The command

```
make cp
```

will copy the commands to the to the appropriate directories.

The programs *uucp* , *uux* , and *uulog* should be put in ‘‘/usr/bin’’. The programs *uuxqt* , *uucico* , and *uuclean* should be put in the *program* directory.

Files required for execution

There are four files which are required for execution, all of which should reside in the *program* directory. The field separator for all files is a space unless otherwise specified.

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

line call-unit speed

where;

line	is the device for the line (e.g. cul0),
call-unit	is the automatic call unit associated with *line* (e.g. cua0), (Hardwired lines have a number "0" in this field.),
speed	is the line speed.

The line

cul0 cua0 300

would be set up for a system which had device cul0 wired to a call-unit cua0 for use at 300 baud.

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 would send 165−7777 to the dial-unit.

LOGIN/SYSTEM NAMES
It is assumed that the *login name* used by a remote computer to call into a local computer is not the same as the login name of a normal user of that local machine. However, several remote computers may employ the same login name.
Each computer is given a unique *system name* which is transmitted at the start of each call. This name identifies the calling machine to the called machine.

USERFILE
This file contains user accessibility information. It specifies four types of constraint;
[1]which files can be accessed by a normal user of the local machine,
[2]which files can be accessed from a remote computer,
[3]which login name is used by a particular remote computer,
[4]whether a remote computer should be called back in order to confirm its identity.
Each line in the file has the following format

login,sys **[** c **]** path-name **[** path-name **]** ...

where;

login	is the login name for a user or the remote computer,
sys	is the system name for a remote computer,
c	is the optional *call-back required* flag,
path-name	is a path-name prefix that is acceptable for *user*.

The constraints are implemented as follows.
[1]When the program is obeying a command stored on the local machine, *MASTER* mode, the path-names allowed are those given for the first line in the *USERFILE* that has a login name that matches the login name of the user who entered the command. If no such line is found, the first line with a *null* login name is used.
[2]When the program is responding to a command from a remote machine, *SLAVE* mode, the path-names allowed are those given for the first line in the file that has the system name that matches the system name of the remote machine. If no such line is found, the first one with a *null* system name is used.

[3]When a remote computer logs in, the login name that it uses must appear in the *USERFILE*. There may be several lines with the same login name but one of them must either have the name of the remote system or must contain a *null* system name.

[4]If the line matched in ([3]) contains a "c", the remote machine is called back before any transactions take place.

The line

    u,m /usr/xyz

allows machine *m* to login with name *u* and request the transfer of files whose names start with "/usr/xyz".
The line

    dan, /usr/dan

allows the ordinary user *dan* to issue commands for files whose name starts with "/usr/dan".
The lines

    u,m /usr/xyz /usr/spool
    u, /usr/spool

allows any remote machine to login with name *u*, but if its system name is not *m*, it can only ask to transfer files whose names start with "/usr/spool".
The lines

    root, /
    ,   /usr

allows any user to transfer files beginning with "/usr" but the user with login *root* can transfer any file.

L.sys

Each entry in this file represents one system which can be called by the local uucp programs. The fields are described below.

system name

The name of the remote system.

time

This is a string which indicates the days-of-week and times-of-day when the system should be called (e.g. MoTuTh0800−1730).
The day portion may be a list containing some of
Su Mo Tu We Th Fr Sa
or it may be *Wk* for any week-day or *Any* for any day.
The time should be a range of times (e.g. 0800−1230). If no time portion is specified, any time of day is assumed to be ok for the call.

device

This is either *ACU* or the hardwired device to be used for the call. For the hardwired case, the last part of the special file name is used (e.g. tty0).

speed

This is the line speed for the call (e.g. 300).

phone

The phone number is made up of an optional alphabetic abbreviation and a numeric part. The abbreviation is one which appears in the *L-dialcodes* file (e.g. mh5900, boston995−9980).
For the hardwired devices, this field contains the same string as used for the *device* field.

login

The login information is given as a series of fields and subfields in the format
expect send **[** expect send **]** ...
where; *expect* is the string expected to be read and *send* is the string to be sent when the *expect* string is received.

The expect field may be made up of subfields of the form
expect[−send−expect]...
where the *send* is sent if the prior *expect* is not successfully read and the *expect* following the *send* is the next expected string.
There are two special names available to be sent during the login sequence. The string *EOT* will send an EOT character and the string *BREAK* will try to send a BREAK character. (The *BREAK* character is simulated using line speed changes and null characters and may not work on all devices and/or systems.)
A typical entry in the L.sys file would be

    sys Any ACU 300 mh7654 login uucp ssword: word

The expect algorithm looks at the last part of the string as illustrated in the password field.

9. 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 *crontab* entries. Others will require manual intervention. Some sample *shell files* are given toward the end of this section.

SQFILE – sequence check file

This file is set up in the *program* 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, 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/copied to the requested destination. If processing is abnormally terminated or the move/copy 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.

LOG – log entry files

During execution of programs, individual *LOG* files are created in the *spool* directory with information about queued requests, calls to remote systems, execution of *uux* commands and file copy results. These files should be combined into the *LOGFILE* by using the *uulog* program. This program will put the new *LOG* files at the beginning of the existing *LOGFILE.* The command

    uulog

will accomplish the merge. Options are available to print some or all the log entries after the files are merged. The *LOGFILE* should be removed periodically since it is copied each time new LOG entries are put into the file.
The *LOG* files are created initially with mode 0222. If the program which creates the file terminates normally, it changes the mode to 0666. Aborted runs may leave the files with mode 0222 and the *uulog* program will not read or remove them. To remove them, either use *rm*, *uuclean*, or change the mode to 0666 and let *uulog* merge them with the *LOGFILE*.

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 to machines are conversing. The form of the file name is

STST.sys

where *sys* is the remote system name.

For ordinary failures (dialup, login), the file will prevent repeated tries for about one hour. 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, 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 with a command to merge *LOG* files and started by a crontab entry on an hourly basis. The file could contain the commands

program/uulog
program/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

program/uuclean −pTM −pC. −pD.
program/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. A shell like

cp *spool*/LOGFILE *spool*/o.LOGFILE
rm *spool*/LOGFILE

can be used.

Login Entry

One or more logins should be set up for *uucp*. Each of the ‘‘/etc/passwd’’ entries should have the ‘‘*program*/uucico’’ as the shell to be executed. The login directory is not used, but if the system has a special directory for use by the users for sending or receiving file, it should as the login entry. 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.

The *L.sys*, *SQFILE* and the *USERFILE* which are put in the *program* directory should be owned by the *uucp* login and set with mode 0400.