UUCP Implementation Description

D. A. Nowitz

Bell Laboratories
Murray Hill, New Jersey 07974

ABSTRACT

Uucp is a series of programs designed to permit communication between UNIX† systems using either dial-up or hard-wired communication lines. This document gives a detailed implementation description of the current implementation of uucp. Is is designed for use by an administrator/installer of the system. It is not meant as a user's guide.

Introduction

Uucp is a series of programs designed to permit communication between UNIX systems using either dial-up or hard-wired communication lines. It can be used for file transfers and remote command execution. The first version of the system was designed and implemented by M. E. Lesk.[1] This paper describes the current (second) implementation of the system.

Uucp is a batch 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. *Execute files* are scripts for UNIX commands that involve the resources of one or more systems.

There are four primary programs:

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

There are a couple of administrative programs:

uulog	gathers temporary log files that may occur due to lockout of the uucp log file and reports some information such as copy requests and completion status.
uuclean	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.

---

† UNIX is a trademark of Bell Laboratories.

1. M. E. Lesk and A. S. Cohen, UNIX *Software Distribution by Communication Link*, private communication.

1. Uucp—UNIX to UNIX File Copy

The *uucp* command is the user's primary interface with the system. The command is 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 where the file or files reside or where they will be copied.

Uucp has several options:

- −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.
- −e*sys* Send this job to system *sys* to execute. (Note that this will only work when the system *sys* allows *uuxqt* to execute a *uucp* command. See the "Uuxqt" and "Security" sections.)
- −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 to the requester on completion of the work.
- −n*user* Notify *user* on the remote machine that a file has been sent.

There are several options available for debugging:

- −r Queue the job but do not start *uucico* program.
- −x*num* *Num* is a level number between 1 and 9; higher numbers give more debugging output.

The destination may be a directory name, in which case the file name is taken from the last part of the source's name. If the directory exists, it must be writable by everybody. (Note that if the destination is a directory name and the "−d" option is specified to create the directory, the directory name must be followed by "/".) The source name may contain special shell characters such as "?*[]". These will be expanded on the appropriate 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 of *user* on the specified system. File names beginning with "~/" translate into the public directory (usually /usr/spool/uucppublic) on the remote system. For names with partial path-names, the current directory is prepended to the file name. File names with ../ are not permitted for security reasons.

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, the system-part of each argument, and the options to classify the work into several types:

[1] Copy source to destination on local system.

[2] Receive files from other systems.

[3] Send files to a remote system.

[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.

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

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

Type 1 - local copy

The copy is done locally. The *−m* and *−n* options are not honored in this case.

Type 2 - receive files

A *work file* is created or appended with a one line entry for each request. The upper limit to the number of files per *work file* is set in *uucp.h*. (The default setting is 20.) After the limit has been reached, a new *work file* is created. (All *work files* and *execute files* use a blank as the field separator.) The fields for these entries are given below.

[1] R

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

[3] The full path-name of the destination file. If the *~something* notation is used, it will be immediately expanded.

[4] The user's login name.

[5] A "−" followed by an option list. The options −m and −d may appear.

Type 3 - send files

Each 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 for these entries are given below.

[1] S

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

[3] The full-path name of the destination or ~something/file-name.

[4] The user's login name.

[5] A "−" followed by an option list. The options −d, −m, and −n may appear.

[6] The name of the *data file* in the spool directory. A dummy name, "D.0" is used when the −c option is specified.

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

[8] The user on the remote system to be notified upon completion of the file copy when the "−n" option is specified.

Type 4 and Type 5 - remote uucp required

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

Type 6 - remote execution

This occurs when the "−e" option is used. In this case, the *uux* facility is used to create and send the request. This requires that the remote *uuxqt* program allows 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 that do not contain a "!" will not be treated as files. (They will not be copied to the execution machine.) An argument that contains a "!" but is not to be treated as a file at the present time, can be escaped by using "()" around the argument. (Note that the "()" symbols must usually be escaped with a "\" symbol.) The "−" is used to indicate that the standard input for *command-string* should be inherited from the standard input of the *uux* command. The following options are available for debugging:

−r Don't start *uucico* or *uuxqt* after queuing the job.

−x*num* *Num* is a level number between 1 and 9; higher numbers give more debugging output.

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* that 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 send command (type 3 above).

For required files that 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 for execution by the *uucico* program.

The *execute file* contains a script that will be processed by the *uuxqt* program. It is made up of several lines, each of which contains an identification character and one or more arguments. The lines are 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 a unique name used for file transmission 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. The *uuxqt* program will check for the existence of all these 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. (Note that if there is a standard input specified, it will also appear in an "F" line.)

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 that 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 (usually /usr/lib/uucp/.XQTDIR) 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. (Note that a check is made to see that the command is allowed as specified in the *uuxqt* program.) After execution, the standard output is copied or sent to the proper place.

3. Uucico—Copy In, Copy Out

The *uucico* program will perform several 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 demon specified in a crontab entry,

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 logins used by remote systems to access uucp.)

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.

Uucico 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 that 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 a level number between 1 and 9; higher numbers give more debugging output.

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, zero 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 that reside in the uucp program 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 *L.sys* file contains information required to make the remote connection:

[1] system name,

[2] times to call the system (days-of-week and times-of-day) and the minimum time delay before retry,

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

[4] line class (this is the line speed on almost all systems),

[5] phone number if field [3] is *ACU* or the device if not *ACU*,

[6] login information (zero or more 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) that 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 each devices that satisfy [3] and [4] until a call is made, or no more devices can be tried. If a device is successfully opened, a lock file is created. If the call is completed, 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 *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 *MASTER* program does a work search similar to the one used in the "Scan For Work" section. (The *MASTER* has been specified by the "−rl" uucico option.) Each message used during the work processing is specified by the first character of the message:

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 for the remote system is complete, at which point an *H* message will be sent. The *SLAVE* will reply with *SY*, *SN*, *RY*, *RN*, *HY*, *HN*, *XY*, or *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 spool directory to the destination. Otherwise, a *CN* message is sent. (In this case, the file is put in the public directory, usually /usr/spool/uucppublic, and the requester is notified by mail.) The requests and results are logged on both systems.

The hangup response is determined by a work scan of the *SLAVE*'s spool directory. If work for the remote system exists 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 unless a "−s" option was specified.

4. Uuxqt−Uucp Command Execution

The *uuxqt* program is used to execute scripts generated by *uux*. The *uuxqt* program may be started by either the *uucico* or *uux* programs or a demon specified by a *crontab* entry. 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 is verified and executed.

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

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

When a *uucp* program can not make a log entry directly into the *LOGFILE* an individual log file is created: a file with prefix *LOG*. This will sometimes occur when more than one uucp process is running. Periodically, *uulog* may be executed to append these files to the *LOGFILE*.

The *uulog* program may also be used to request the output of *LOGFILE* entries. The request is specified by the use of the 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 uucp daily demon. Its function is to remove files from the spool directory that are more than 3 days old. These are usually files for work that can not be completed. The requester of this work is notified that the files have been deleted.

There are several options:

- −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. This mail is sometimes useful for administration.)
- −n*hours* Change the aging time from 72 hours to *hours* hours.
- −p*pre* Examine files with prefix *pre* for deletion. (Up to 10 of these options 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 out/in any file that is readable/writable by a 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 administrator of the *uucp* system.

- The login for uucp does not get a standard shell. Instead, the *uucico* program is started so that all work is done through *uucico*.
- The owner of the uucp programs should be an administrative login. It should not be one of the logins used for remote system access to uucp.
- A path check 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 of the caller's identity.
- 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 the uucp administrative login and have mode 0400 to protect the phone numbers and login information for remote sites.

— The programs *uucp*, *uucico*, *uux*, *uuxqt*, *uulog*, and *uuclean* should be owned by the uucp administrative login, have the setuid bit set, and have only execute permissions.

8. Uucp Installation

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 or the uucp administrative login. However, several remote computers may use the same login name.

Each computer should be given a unique *system name* that is transmitted at the start of each call. This name identifies the calling machine to the called machine. The *login/system* names are used for security as described later in the *USERFILE* section.

There are several source modifications that may be required before the system programs are compiled. These relate to the directories, local system name, and attributes of the local environment.

There are several directories used by the uucp system:

lib	(/usr/src/cmd/uucp) — This directory contains the uucp system source files.
program	(/usr/lib/uucp) — This is the directory used for some of the executable system programs and the system files. Some of the programs reside in "/usr/bin".
spool	(/usr/spool/uucp) — This is the uucp system spool directory.
xqtdir	(/usr/lib/uucp/.XQTDIR) — This directory is used during execution of the *uux* scripts.

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

There are two files that may require modification, the *makefile* file and the *uucp.h* file. (On some systems, the makefile is named *uucp.mk*.) In addition, the "uuxqt.c" program may be modified as indicated in the "Security" section above. The following paragraphs describe the modifications.

uucp.h modification

Several manifests in "uucp.h" may need modification for the local system environment:

UNAME	should be defined if the "uname" function is available.
MYNAME	should be modified to the name of the local system if UNAME is *not* defined.
ACULAST	is the character required by the ACU as the last character. For most systems, it is a "—".
DATAKIT	should be defined if the system is on a datakit network.
DIALOUT	should be defined if the "C" library routine "dialout" is available.

makefile modification

There are several *make* variable definitions that may need modification:

INSDIR	is the *program* directory (e.g., INSDIR=/usr/lib/uucp). This parameter is used if "make cp" or "make install" is used.
IOCTL	is required to be set if the "ioctl" routine is *not* available in the standard "C" library; the statement "IOCTL=ioctl.o" is required in this case.
PUBDIR	is a public directory for remote access. This is also the login directory for remote uucp users. It should be the same as that defined in "uucp.h".

SPOOL is the uucp spool directory. This should be the same as that defined in "uucp.h".

XQTDIR is the directory for uuxqt to use for command execution. It is also defined in "uucp.h".

OWNER is the administrative login for uucp.

Compile the system

The command

make install

will make the required directories, compile all programs, set the proper file modes, and copy the programs to the proper directories. This command should be run as *root*. The command

make

will compile the entire system.

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 that are required for execution. They should reside in the *program* directory. The field separator for all files is a space.

L-devices

This file contains call-unit device and hard-wired connection information. The special device files are assumed to be in the */dev* directory. The format for each entry is

type line call-unit speed

where

type is a device type such as ACU or DIR. The field can also be used to specify particular ACUs for some calls by using a suffix on the ACU field, e.g., ACU3. This names should be used in *L.sys*.

line is the device for the line (e.g., cul0).

call-unit is the automatic call unit associated with *line* (e.g., cua0). Hard-wired lines have a number "0" in this field.

speed is the line speed.

The line

ACU cul0 cua0 300

would be set up for a system that has device "/dev/cul0" wired to a call-unit "/dev/cua0" for use at 300 baud.

L-dialcodes

This file contains the dial-code 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.

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 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 *sys*.

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 on the first line in the *USERFILE* that has 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 on the first line in the file that has the system name that matches 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". (Note that this type restriction is seldom used.)

The lines

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

allows any remote machine to login with name *u*. If its system name is not *m*, it can only ask to transfer files whose names start with "/usr/spool". If it is system *m*, it can send files from paths "/usr/xyz" as well as "/usr/spool".

The lines

```
root, /
, /usr
```

allow any user to transfer files beginning with "/usr" but the user with login *root* can transfer any file. (Note that any file that is to be transferred must be readable by anybody.)

L.sys

Each entry in this file represents one system that can be called by the local uucp programs. More than one line may be present for a particular system. In this case, the additional lines represent alternative communication paths that will be tried in sequential order. The fields are described below.

system name

The name of the remote system.

time

This is a string that 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 allowed for the call. Note that a time range that spans 0000 is permitted, for example, 0800-0600 means all times are allowed other than times between 6 and 8 am.

An optional subfield is available to indicate the minimum time (minutes) before a retry following a failed attempt. The subfield separator is a ",". (e.g., Any,9 means call any time but wait at least 9 minutes after a failure has occurred.)

device

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

class

This is usually the line speed for the call (e.g., 300). The exception is when the "C" library routine "dialout" is available in which case this is the dialout class.

phone

The phone number is made up of an optional alphabetic abbreviation and a numeric part. The abbreviation should be one that appears in the *L-dialcodes* file (e.g., mh5900, boston995−9980). For the hard-wired 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] ...

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. (e.g., login--login will expect *login*; if it gets it, the program will go on to the next field; if it does not get *login*, it will send *null* followed by a new line, then expect *login* again.)

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 number from 1 to 9 may follow the *BREAK* for example, *BREAK1* will send 1 null character instead of the default of 3. Note that *BREAK1* usually works best for 300/1200 baud lines.

A typical entry in the L.sys file would be

sys Any ACU 300 mh7654 login uucp ssword: word

The expect algorithm match all or part of the input string as illustrated in the password field above.

9. Administration

This section indicates some events and files that must be administered for the uucp system. Some administration can be accomplished by *shell files* 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 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 manually.

TM − temporary data files

These files are created in the *spool* directory while a file is 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. After the entire file is received, the *TM* file is moved/copied to the requested destination. If processing is abnormally terminated 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

program/uuclean −pTM

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

LOG − log entry files

During execution, log information is appended to the *LOGFILE*. If this file is locked by another process, the log information is placed in individual log files which will have prefix *LOG*. These files should be combined into the *LOGFILE* by using the *uulog* program. This program will append the *LOGFILE* with the individual log files. 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.

The *LOG*. files are created initially with mode 0222. If the program that 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 into the *LOGFILE*.

STST – system status files

These files are created in the spool directory by the *uucico* program. They contain information such as login, dial-up or sequence check failures or will contain a *TALKING* status when two machines are conversing. The form of the file name is

STST.sys

where *sys* is the remote system name.

For ordinary failures, such as dial-up or login, the file will prevent repeated tries for about 55 minutes. This is the default time; it can be changed on an individual system basis by a subfield of the time field in the *L.sys* file. For sequence check failures, the file must be removed before any future attempts to converse with that remote system.

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 device. 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 (usually only on system crashes). They will be ignored (re-used) after 1.5 hours. When runs abort and calls are desired before the time limit, the lock files should be removed.

ERRLOG – uucp system error file

This file is created in the *spool* directory to record uucp system errors. Entries in this file should be rare. The messages come from the *ASSERT* statements in the various programs. Wrong modes on files or directories, missing files, and read/write system call failures on the transmission channel may cause entries in the *ERRLOG* file.

Shell Files

The *uucp* program will spool work and attempt to start the *uucico* program, but *uucico* will not always be able to execute the request immediately. 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

```
/usr/bin/uulog
program/uucico  -r1  -sinter
program/uucico  -r1
```

The "−r1" option is required to start the *uucico* program in *MASTER* mode. The "−s" option can be used for polling as illustrated in the second line where machine *inter* is being polled. The third line will process all other spooled work.

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 that 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 that 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

Two or more logins should be set up for *uucp*. One should be an administrative login: the owner of all the uucp programs, directories and files. All others are used by remote systems to access the uucp system. Each of the "/etc/passwd" entries for the *access* logins should have "*program*/uucico" as the shell to be executed. The login directory should be the public directory (usually /usr/spool/uucppublic). The various *access* login names are used in the *USERFILE* to restrict file access.

File Modes

The programs *uucp*, *uux*, *uucico*, *uulog*, *uuclean* and *uuxqt* should be owned by the *uucp* administrative login with the "setuid" bit set and only execute permissions (e.g., mode 04111). The *L.sys*, *SQFILE* and the *USERFILE*, which are put in the *program* directory should be owned by the *uucp* administrative login and set with mode 0400. The mode of *spool* should be "0755". The mode of *xqtdir* should be "0777". The *L-dialcodes* and the *L-devices* files should have mode 0444.

January 1981