AT&T Bell Laboratories
Murray Hill, New Jersey 07974

Computing Science Technical Report No. 111

Another Try at Uucp

Robert T. Morris

September 6, 1984

Another Try at Uucp

Robert T. Morris

AT&T Bell Laboratories
Murray Hill, New Jersey 07974

ABSTRACT

Uucp, the major software package used to communicate between UNIX† systems, has a history of insecurity and inefficiency. The new implementation described here is smaller and cleaner than previous ones, trading excess and untrustworthy features for understandibility. This uucp also uses a faster and more compact method of storing spooled job requests.

September 6, 1984

† UNIX is a Trademark of AT&T Bell Laboratories.

Another Try at Uucp

Robert T. Morris

AT&T Bell Laboratories
Murray Hill, New Jersey 07974

1. Introduction

Uucp uses telephone lines and other networks to transfer data between UNIX† computers. It was originally concieved of and written by M. Lesk, and rewritten by D. Nowitz.

The old uucp was designed on small machines with light traffic and little concern for security. Time have changed. With many hundreds of machines running uucp, one cannot assume that no uucp requests will be malicious. It is too easy to intercept, alter, or forge information sent via old uucp. By default, the old uucp will allow users on other machines to access any file. Provisions may be made to prevent this, since it is utterly undesirable, but in practice many machines do not. This new uucp helps cure these problems and more, is designed for large machines, and can use networks that have computer interfaces unlike that of a telephone.

This paper has a companion which describes how to install and maintain uucp, so those aspects will not be stressed here.

2. Directories

Uucp uses three directories, whose abbreviations are *spool*, *lib*, and *public*. *Spool*, usually /usr/spool/uucp or /usr/spool/muucp, holds information about data to be transmitted to other machines and the data itself. *Lib*, /usr/lib/uucp or /usr/lib/muucp, contains files describing how to contact other computers. *Public*, conventionally /usr/spool/uucppublic, is a directory that any user on any system may send files to.

3. User Commands

The three major commands that users run are *uucp*, *uux*, and *uuget*. *Uucp* sends files to other machines, *uux* arranges for commands to be executed on other machines, and *uuget* allows a user to receive files sent specifically to him.

3.1. The *uucp* Command

The syntax of *uucp* is

```
uucp [−m] [−r] [−e] source-files system!destination
```

where *destination* may be a directory but *source-files* may not. If any of the files can not be sent, and the −e flag is not specified, mail will be sent to the requester. Mail will be sent when the file is actually transferred if the −m flag is present. The −r option prevents the daemon that actually transfers from being executed.

Uucp goes through the same set of actions for each *source-file*. The file must exist and be globally readable. *Uucp* copies the file to a temporary file in *spool* whose name is *D.number*; *number* is the second number in the file *spool/SEQ*, which is incremented after use. Then an entry is made at the end of the file *spool/J.system*, where *system* is the name of the computer the file is to be sent to. This entry includes the names of the files involved, the requesting user, the time, and flags indicating how much mail should be sent. Finally, the program *uuco* is invoked to

† UNIX is a Trademark of AT&T Bell Laboratories.

attempt the actual transfer.

Uucp may be used to receive files from another system instead of sending them, and to send to a file on the local system. The former is done by just making an entry in *spool/J.system,* and the latter by *uucp* itself. In general, *uucp* will only write on files in /usr/spool/uucppublic.

3.2. The *uux* Command

Uux arranges for commands to be executed on other systems. Its syntax is

```
uux [−] [−m] [−a mail-address] system!command [arguments]
```

If the − flag is given, uux will read its standard input and arrange for that to be fed to *command* on the remote system. The presence of the −m flag causes the output of the command to be mailed back to you. If a mail address is specified by the −a flag, that will be the destination of error reports and the output caused by −m. This is used by the *rmail* command. Any parentheses in *arguments* will be replaced with spaces to retain compatibility with older versions of *uux.*

All that *uux* does is have one or two files sent to the system that the commands are to be executed on. One of these files holds information about the command. It is named *D.localZnnnn,* where *nnnn* is the first number in *spool/SEQ,* and *local* is the local system name. It is sent to *X.localZnnnn* on the remote system. The first line in the file, called the *X.* file, contains the letter *U,* the requesting user's name, and the local system name. It may also have the mail address specified after the −a option. Then there are lines describing the input and output of the command, and what files are neccessary for the execution. These lines begin with the letters *I, O,* and *F.* The last line is the command line prefixed by the letter *C.* A typical *X.* file looks like this:

```
U rtm research
F D.pantherX0000
I D.pantherX0000
C rmail ftg
```

which would run the *rmail* command with input taken from the file *D.pantherX0000.* If standard input is wanted, it is sent under separate cover by *uux,* from files like *D.remoteZ0000* to files like *D.remoteX0000.* The −m option is implemented by directing the output of the command to a file with an *O* line in the *X.* file, and sending another *X.* file to mail back that file.

Once *uux* has created these files in the *spool* directory, it adds lines to *spool/J.system,* just like *uucp.*

4. The *J.system* Files

There is a *J.* file in *spool* for each system for which there is work. The first line in the file describes the status of the system, and each of the rest indicates a file to be sent or received to or from the system.

4.1. The Status Line

The status line has this form:

```
system status ntries time delay comment
```

System is the name of the system. *Status* is zero if the system is thought to be accessible, and one if not. The rest of the fields are ignored if *status* is zero. The general scheme is that calls to a system are made every once in a while to see if it is up, and the information in the status line is used to tell when to try it. *Ntries* is the number of tries that have been made, *time* is the time (number of seconds since 1970) of the first unsuccessful try, *delay* is the number of seconds to wait between tries, and *comment* is a pithy statement about why the system can not be contacted. *Delay* is initially 3300 seconds (a little less than an hour), and is increased a little after each try. It is increased so that a system which is down for a little while will get data fairly quickly when it comes up, and a system which is down for many days, or forever, will not be called constantly after the first few tries.

4.2. The Rest of the *J.* File

Each of these lines is

    S-or-R flags source destination user time comment

S-or-R is S for sending and R for receiving. *Flags* is an octal number, which is non-zero only when the −e or −m arguments were passed to the *uucp* command. *Source* is the file the data is to be taken from, usually a file in *spool* starting with *D. Destination* is the file or directory to be written on, *user* is the user who initiated the request at time *time*, and comment tells, in terms comprehensible to humans, what is going on.

4.3. An Example of a *J.* File

```
usg 1 1 415134441 3300 Can't Connect
S 00 D.usgZ0224 D.usgX0224 rtm 415134406 uux (stdin) usg!rmail mjs
S 00 D.pantherZ0225 X.pantherZ0225 rtm 415134406 uux (cmd) usg!rmail mjs
```

This file might have been created by a *mail usg!mjs* command on the *panther* system, which in turn invoked *uux − usg!rmail mjs*.

4.4. Reading and Writing *J.* Files

Uux and *uucp* seek to the end of the *J.* file and append. They create a lock file named *spool/LT.J.system* in case appends are not atomic. Programs such as *uustat* which merely read the information create a lock to assure consistency.

5. The *uuco* Command

Uuco does the actual transfer of data to other systems. If invoked with no arguments, it scans the *spool* directory for *J.* files and tries to contact those sytems. If invoked with *−s system*, *uuco* only contacts *system*. If given the −f flag, *uuco* will ignore retry delay information.

5.1. Reading in the *J.* File

This is trickier than the method described in section 4.4, because *uuco* often runs for long periods of time, and there is a significant chance that some disaster will happen; the system may crash, *uuco* may crash, or somebody might kill uuco. The first thing *uuco* does is try to lock the system with the file *spool/LT.system;* if the system is already locked, it exits. Then *uuco* reads in the *J.* file for whatever system it is working on, and moves it to *K.system*. When *uuco* successfully finishes, it removes the *K.system* file and removes the lock. If some disaster happens, the *K.system* file is read in by the next *uuco* process that deals with that system.

5.2. Making a Connection

The information to make connections is in the files *lib/L.sys* and *lib/L.sys1*. First *L.sys* is examined, then *L.sys1*. If a connection succeeds, the *stat* field in the *J.* file is set to zero; if not, the *stat* field is set to one, and the other fields updated.

5.2.1. Retry Times

If the *status* field of the *J.* file is non-zero, *uuco* checks to see if enough time has passed since the last attempt. Enough time has passed if *delay* times *ntries* plus *time* is greater than the current time.

5.2.2. Format of the *L.sys* File

Each system that *uuco* knows about has one or more lines in *lib/L.sys*. Each line looks like

    name days−times how speed phone login-info

Name is the system name. *Days−times* is one of *Any*, *SaSu*, or *Never* optionally followed by two times separated by a dash. If *days* is *Any,* the system may be called on any day, if *SaSu,* then only on weekends, and if *Never* then the system may never be called. The system may only be called between the two times, if specified, which must be in twenty four hour time. *How* tells what medium is to be used; it is usually one of ACU, DK, or DIR. *Speed* and *phone* have meanings dependent on *how. Login−info* is pairs of space separated strings; when the first one is read from the other system, the second one is sent followed by a newline. The first may be "" which is seen before the first character. The first also may be *string1−stuff−string2[−morestuff−etc.],* which causes *stuff* to be sent if *string1* is not received, after which *string2* is expected. The second string of a pair may be BREAK, EOT, or "", which send a break, an at-sign followed by a control−d, and a bare newline, respectivly. If the second string has a backslash in it followed by one of the letters s, c, b, d, n, or a three digit octal number, it is converted into a space, an indication that no newline should be sent after the string, a break, a one second delay, a newline, and the ascii character represented by that number, respectivly. Otherwise, fields represent themselves. If no login information is present, *uuco* assumes it is already logged in.

5.2.3. Dialout Connections

Dialout connections are specified by *ACU* in the third field of an *L.sys* line. The fourth field is the line speed or class, and the fourth is the phone number, possibly prefixed by an alphabetic dialcode. The dialcode is looked up in *lib/L.dialcodes,* and then in *lib/L.dialcodes1.* If the *dialout()* subroutine is not available, *lib/L.devices* must contain information about dialing devices. Each line contains *ACU* possibly follow by a prefix and a suffix separated by commas, then the line device name, then the dialer, and finally the speed or class. For instance, this describes a device /dev/cul0 at 1200 baud with associated dialer /dev/acu0; numbers written on the dialer must be followed by the character <.

```
ACU,,< cul0 acu0 1200
```

The null string between the two commas means that nothing should be prefixed to the number.

5.2.4. Direct Connections

A *DIR* in the third *L.sys* field means a direct connection through some device. The device is specified by the fifth field, and the speed by the fourth. For compatibility with older uucp programs, if the third and fifth field are the same string, *uuco* pretends the third was a *DIR*.

5.2.5. Datakit Connections

DK indicates a datakit connection. The fourth field is the traffic type, either 1 or 2, and the fifth is the datakit address, for instance, *area/exchange/machine.*

5.2.6. Connections via Other Commands

If the third *L.sys* field is *CMD,* the fourth field is taken as a command to run to send a file. The command is invoked with these arguments:

```
command remote-host from-file to-file user local-host
```

The command should return 0 for success, 1 for an error, and 2 for denial of the request. If the command exists, it alone is used to transfer files, and none of the succeeding sections in the paper are relevant.

5.3. Logging In

The login information on the *L.sys* line is used to log into the machine if neccessary. The end result of logging in is that *uuco* is connected to a *uuci* process on the other machine.

5.4. Protocols

Most methods of connection require protocols. There is one standard one that is used to set things up, and then another one, named *g*, is invoked to do error-correcting transmission. *Uuco* has a table indicating which protocols are appropriate for each network. This can be overridden by placing a comma followed by the letter of the protocol after the *how* field in the *L.sys* file. For instance,

```
eagle Any DK,d 0 eagle login--login nuucp password a1b2c3
```

causes the *d* protocol to be used.

5.5. Sending Files

There are three possible results from an attempt to send a file, success, denial, and error. Success and denial result in a line being added to *spool/LOGFILE* and possibly some mail sent to the requesting user. If an error occurs in the transmission, the data files will not be removed and the entry will be left in the *J.* file. Lines added to the log file have the system, user, date of transmission, success or failure, and a description of what the user requested.

5.6. Hanging Up

Uuco notifies the remote system when it is finished sending files. The other system then has the option of sending files to the local system. When both systems are finished, *uuco* adds an entry to *spool/SYSLOG* saying that the remote system was successfully contacted. If a problem occured which prevented a normal hangup exchange, such as a lost telephone connection, *uuco* adds a line to *spool/SYSLOG* describing the problem.

5.7. Misc.

Uuco ignores hangup, quit, and interrupt signals. If it is hit by a terminate signal, it will try to gracefully clean up as fast as it can. If it is hit by some other signal, it will immediatly do a minimal cleanup and then kill itself with that signal.

6. The *uuci* Program

Uuci is a link to *uuco*, since they are very similar. It takes no options, since it is designed to be used as the login shell for *nuucp*. The scheme is that a *uuco* process on machine *A* will use phone lines or a network to log into machine *B* as user *nuucp*, and thus be connected to a *uuci* process on machine *B*.

6.1. Setting Things Up

Uuci first makes its terminal unwritable and puts it raw mode. Then it and the *uuco* process exchange information, settle on a common protocol, invoke it, and commence trading requests to send files. Part of the information *uuci* recieves is the name of the calling machine. Since old versions of uucp truncate system names to seven characters, *uuci* will find the best match in *lib/L.sys* for the calling name if its length is seven. Once it knows the system name, *uuci* tries to lock it using the file *spool/LT.sys* to prevent multiple connections; if the lock file already exists, the calling system is notified and the connection is dropped.

6.2. Requests to Send Files

Uuci only allows files to be sent to /usr/spool/uucppublic, files in *spool* that begin with *D.* or *X.*, which are temporaries for *uux*, and /dev/null. If the file name to be sent to starts with ~/ (tilde followed by a slash), the ~/ is replaced by */usr/spool/uucppublic*. If the file name consists of a tilde followed by a user's name, it is allowed, and is expanded later into a private holding place for that user. If the file name was illegal, the caller is notified, and may try to send another file. Otherwise, a temporary file is created, the data is read into it, and it is copied to the destination. The temporary is mode 0600 and the destination is mode 0644. The request is logged in *spool/LOGFILE* whether successful or not.

If the destination file name is a tilde followed by a user's name, the file is copied to *spool/userfiles*, a record that the file exists is appended to *spool/userfiles/U.user*, and mail is sent to the user if that was the first file sent to him in the last few days. All files involved here are made inaccessible to normal users. In effect, this allows files to be sent to people instead of path names.

6.3. Requests to Receive Files

No such requests are honored by *uuci*, though all are recorded in *spool/LOGFILE*.

6.4. Polling and Hanging Up

When the caller is done, it asks *uuci* if the called system has any work to send; this is called polling. Because *uuci* can not be sure that the caller is who it claims to be, the answer should always be no. This is not realistic, though, because some systems have no way of establishing connections, and must rely on other machines to call them and let work be sent both ways. In order to minimize the risk, *uuci* looks up the caller in *lib/L.logins*; if there is no entry for it, the caller is not allowed to poll. If there is an entry, and the caller is logged in as specified in *L.logins*, *uuci* believes the caller to be who it says it is. Thus each polling system may be given a separate login and password. This is a *L.logins* file that allows systems *jello* and *mango* to poll if logged in as *Umango* and *Ujello*, respectivly –

```
jello    Ujello
mango    Umango
```

Finally, if files beginning with *X.* have been sent, the program *lib/uuxqt* is started to mediate the execution of *uux* jobs.

7. Remote Execution

Uuxqt searches through the spool directory for files beginning with *X.*, which were sent by *uux* programs on other systems. To prevent multiple *uuxqt* processes, it uses *spool/LT.XQT* as a lock file. The format of the *X.* files is described in the section about *uux*. The file *lib/L.cmds* contains a list of allowable command names and where they are found; *uuxqt* does not search for them in any standard set of places. This list should be very limited, and include only commands that pose no security risk. Thus the *rmail* command is permissable because its functions are limited, but the *cp* command is a bad candidate because it allows unlimited access to files. A typical *L.cmds* file might look like this.

```
rmail /bin/rmail
rnews /usr/bin/rnews
```

Uuxqt passes the names of the requesting host and user in the environment variables *RHOST* and *RUSER*. Either of these may be forged.

8. Cleaning Up

If left to itself, *uuco* will keep trying to contact a system for which it has work forever. The *uuclean* program puts bounds on this by deleting work more than three days old. Mail is sent to the user who requested each deleted job. Other files in the spool directory are also deleted by uuclean if they are too old; the affected files are those that begin with *D.*, *X.*, or *TM.*; *J.* files are only removed if all jobs within them are too old. *Uuclean* accepts pairs of arguments specifying file type and an age in seconds; this command line has uuclean delete jobs older than a week –

```
uuclean j 604800 d 604800
```

Note that times for both jobs and *D.* files must be given, because otherwise jobs descriptions might be preserved but not the corresponding data.

9. Other Things

In which facets shared by most uucp programs are detailed.

9.1. Locking

There are three resources that need locking – connections to other machines, files, and processes that should only be running in one instance. All locks are named *spool/LT.resource*, where resource is a machine, file, or process name. Care is taken that processes remove locks even upon abnormal exit – all signals that can be are trapped by a subroutine that removes locks. Because this may not work, locks more than an hour old are invalid. All uucp programs that might run an hour touch their locks periodically. Locks are readable, though not writable, and their contents are the process number and program name of the creating process.

9.2. Signals

All uucp programs but *uuci* ignore quit, interrupt, and hangup signals. *Uuci* is the exception because it must not linger after abnormal hangups, and can not be assured of read errors on hung up terminals. *SIGTERM* is caught by *uuco*, *uuci*, and *uuxqt*, and causes them to attempt a fairly fast and clean exit; *SIGTERM* should be tried before *SIGKILL* if one of these programs causes trouble.

9.3. Log Files

The log files, *LOGFILE*, *SYSLOG*, *TIMELOG*, and *CLEANLOG*, are all in the spool directory, and will not be automatically created if not present. The format the lines in each of these files is identical, and includes system name, user name, date and time, *LCL* or *RMT* to indicate local or remote work, and finally the text to be logged. These files should be regularly be truncated. They can be used to monitor traffic, problems with connections to other systems, and problems with security.

9.4. System Names

System names may be aliased with the files *L.genequiv*, *L.sysequiv*, and *L.netequiv*, in the *lib* directory. Each contains pairs of names, and alises the first to the second in various contexts. *L.genequiv* does global aliasing, *L.sysequiv* aliases only names for lookup in *L.sys*, and *L.netequiv* affects names passed to network interface subroutines. An entry aliasing *local* to the local system name should exist in *L.genequiv* **if no *uname()* system call is available.** This file might look like this –

```
local  this-system
vax  unixvax
a  aunix
b  bunix
```

which allows certain abbreviations as well as supplies the local system name.

10. Other Administrative Information

At least two logins are needed for uucp; one to own the files and programs, and one or more for other systems to log in with. The latter need to have the shell *lib/uuci*.

Arrangements must be made to run *lib/uuco* every hour, *lib/uuclean* every day, and truncate the log files once every week. This is best done via *cron*.

11. Security

Past versions of uucp have had a history of being insecure. This version should be secure if installed correctly. One should avoid adding commands to *L.cmds*. File access is restricted, but this is offset by the ability to send to a user; the recipient is notified by mail and may retrieve the files with the *uuget* command. The files are kept in a secure location, *spool/userfiles*, in the period between reciept and retrieval.

This *uucp* gives few priviledges to local or trusted systems; provisions may be made to allow them to poll if the local machine has no outgoing connections or in the interestes of efficiency, but they can't access a wider range of files or commands. This is because machines under the same administration are usually connected by networks providing wide ranges of features to local machines; such functionality in *uucp* is misplaced, since in most cases it is used mostly to forward mail and news to foreign sites.