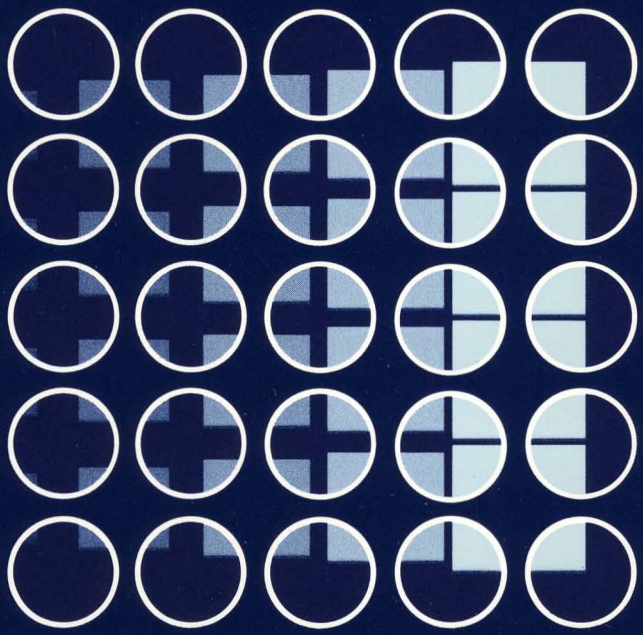
ULTRIX-32™
Programmer's Manual

ULTRIX-32™
UUCP Installation and Administration

Order No. AA-BG62A-TE



digital software

ULTRIX-32™
UUCP Installation and Administration

Order No. AA-BG62A-TE

---

digital equipment corporation, merrimack, new hampshire

First printing, May 1984

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by DIGITAL or its affiliated companies.

The postage-paid READER'S COMMENTS form on the last page of this document requests your critical evaluation to assist us in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DEC	ULTRIX-32
DECUS	UNIBUS
MASSBUS	VAX
PDP	VMS
ULTRIX	VT
ULTRIX-11	digital™

UNIX is a trademark of AT&T Bell Laboratories.

Information herein is derived from copyrighted material as permitted under a license agreement with AT&T Bell Laboratories.

This software and documentation is based in part on the Fourth Berkeley Software Distribution under license from the Regents of the University of California. We acknowledge the Electrical Engineering and Computer Sciences Departments at the Berkeley Campus of the University of California for their role in its development.

UUCP Installation and Administration

Table of Contents

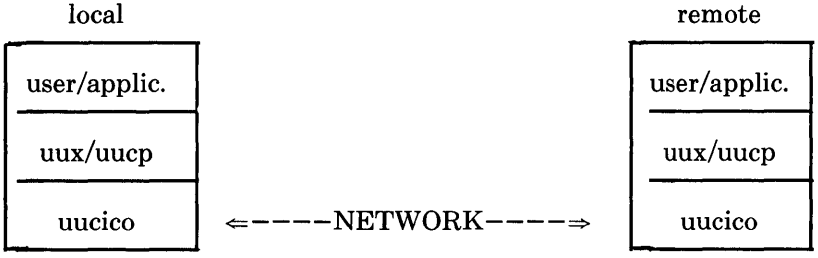
UUCP Installation and Administration

1. Introduction

The purpose of this document is to describe the installation and administration of the current version of *uucp*. The current release is an enhancement of the UNIX† version 7 *uucp* software. Major enhancements include improved performance, security, and numerous bug fixes. A quick reference guide is provided in the appendix.

1.1. Brief Overview of the UUCP System

The *uucp* system consists of three program levels as follows:



At the highest logical level is the user or application program. This level will use *uux* or *uucp* to initiate requests for file transfers and/or remote job execution. *uux* is the remote execution program and *uucp* is the program that spools user requests for file transfers. Both programs convert higher level requests into the format required by the file transfer daemon *uucico*. Each request is queued in the appropriate spool directory.

uucico is the program which performs the file transfer over the network. Before the transfer takes place though, *uucico* must go through several stages. First *uucico* must call up the destination system and log in. After a successful login, the handshaking stage takes place between the destination *uucico* daemon process and the local daemon. At this stage the daemons will determine if the opposing systems have permission to use the local resources at each machine. If the handshake succeeds the daemons will then select which protocol will be used to ensure that raw data is reliably

This document is partially based on work done by D. A. Nowitz and M. E. Lesk.

† UNIX is a trademark of Bell Laboratories.

transmitted over the network. Each daemon will then use the corresponding packet driver software to send and receive raw data. At this point the file transfer process begins. The local daemon will search the spool directories for job requests, build a list of files to transfer, and then start transmitting the files. A file transfer protocol is used to ensure that each file is successfully transmitted only once or to notify the user that the file could not be transmitted with the reason for failure. After all of the files have been transmitted the destination site will begin to transfer files back to the local system. When both systems have no more work to be done, the connection through the network is broken and the conversation is complete.

Throughout the conversation temporary files are created and removed. Lock files are created in the top level spool directory (usually /usr/spool/uucp) that correspond to the remote system (LCK..sys) and the hardware device used for communication (for example, LCK..cua0). Status files (STST. files) are updated that keep track of accessability of each system. Many other temporary files are created, all of which should be gone by the time the conversation is complete.

The *uucp* system also includes the remote execution programs: *uux* and *uuxqt*. *uux/uuxqt* will execute commands on a specified remote system. *uux* spools the command and associated data into the appropriate spool directories. *uucico* will then transfer the files to the remote system. At the remote system *uuxqt* will start when these execution files arrive. *uuxqt* will scan through the execution spool directories (X.) searching for commands to execute. If a command has arrived along with the data files needed by the command, *uuxqt* will try to execute the command. *uuxqt* also creates LCK files. LCK files are created for the type of command it is looking for (for example, LCK.XQTrmail or LCK.XQT for all commands) and a LCK file for the command *uuxqt* is currently working on (for example, LCK.X.decvaxX0123).

In the event that some temporary files do not get removed, shell scripts are run periodically to clean up the residue. In some instances manual intervention may be necessary. This document will provide the tools needed by the system manager to maintain the system and resolve problems when they arise.

1.2. Enhancements

1.2.1. Improved Spooling

The most significant enhancement is the improved file spooling algorithm. In the original version all work related files were placed in the SPOOL directory (usually /usr/spool/uucp). The SPOOL directory has now been split out into separate subdirectories including the option of per system subdirectories. This subdirectory scheme prevents a huge buildup of data files from slowing down the *uucp* transfer process (*uucico*). The per-system subdirectories prevent the failure of one remote system from effecting the performance of communications with other systems.

In addition to the subdirectory enhancements the methods for scanning the spool directories has been improved. All of these changes have significantly improved

uucp's ability to handle greater loads.

1.2.2. Improved Security

uucp will now check to see if the remote system has permission to login to the local system. The action taken by *uucp* (accept/reject) depends on the contents of the USERFILE and the existence of the file /usr/lib/uucp/INSECURE. Also, the USERFILE now provides remote execution security. A more complete discussion of the USERFILE and *uucp* security is presented later.

1.2.3. Miscellaneous Enhancements and Bug Fixes

In addition to the above enhancements numerous bug fixes have been made as well as some internal improvements. The *uustat* program has been installed and improved. *uustat* provides job status and control to *uucp* users. This version also supports several modems/auto call units (acu) including: the DF03/DF02, Bell 212/801, Hayes Smartmodems, and Ventel MD212.

1.3. Caveat

uucp has been improved over earlier versions. However, this does not imply that *uucp* is now self administering. A watchful eye is still needed. Given the loads that can be generated by the USENET or other high demand programs, a failure of one or more sites can cause sizable backlogs of files. These anomalous situations must still be handled manually. At least in this present version something can be done to rectify the problem (other than trashing files). The topic of what to do when something goes wrong will be discussed later.

2. Installation

uucp installation requires a three-step process. However, before you can install *uucp*, you must select the appropriate hardware.

- Install the appropriate hardware (See Subsection 2.1)
- Set up the special files corresponding to the hardware (See Subsection 2.2)
- Install the minimum set of *uucp* directories (See Subsection 2.3.8.1)
- Install *uucp* commands (if necessary)
- Update Administrative files: USERFILE, L.sys, L-devices, L.cmds

2.1. Installing Supported Hardware

uucp currently operates with the following hardware (only software written for DIGITAL hardware is supported by DEC):

- Bell System 801 type Auto Call Unit (ACU) with with a DN-11 line card and 212 type data set

- DEC DF02 or DF03 Auto Call Units with modem
- Direct connect with a null modem cable such as a BC03-M
- Direct connect with a modem link
- Hayes Smartmodem1200
- Ventel MD212 models
- Racal-Vadic 3450

A brief overview of hardware installation procedures follows. See the User's Guide for specific devices or the System Installation Guide for more detailed information.

2.1.1. Installing Bell System Hardware

The 801 ACU has four option switches that you must set as follows:

(0=open 1=closed)

S1 = 1000[1]
S2 = 0101
S3 = 11010
S4 = 11[00]

Set the 212 data set as follows:

S1 = [0]001
S2 = 110001100
S3 = 11110000
S5 = 00

Press the high speed button in for 1200 baud. Leave the button out for 300 baud.

2.1.2. Installing DEC ACUs

Connect the DF02-AC or DF03-AC auto call modem to a port on the local system using a straight-through cable. Connect the ACU to the phone line by following the instructions in the *DF02-AC or DF03-AC Modem User's Guide.* Set the ACU communications bit rate, see switch options on the DF02/03-AC ACU in the modem user's guide. If the ACU is connected to a DL11 interface, then the ACU communications bit rate must match the DL11 speed. If the operating system is ULTRIX-11 then the ACU dialing rate (not the modem speed) must be set to 300 BPS. Otherwise 1200 BPS can be used.

2.1.3. Installing Other Brands of Modems

Consult the manufacturer's installation guide for the specific modem to use. One hint is to configure the modem such that commands typed to the modem are not echoed back by the modem.

2.1.4. Installing Direct Connects

To install a hardwired direct link, connect a null modem cable from a port on the local system to a port on the remote system. For a direct link with a modem, connect the modem to a port on the local system to a port on the remote system with a straight-through cable. Follow the modem installation instructions to setup the direct modem link. A direct connect should be able to run at speeds as high as 9600 BPS.

2.2. Creating Special Device Files

After connecting the hardware, you create the corresponding special files. A special file must exist for both the ACU and modem line. You need only one special file for modems with integral dialers, such as DF03s, because the ACU and modem appear as one unit to the software. The following naming conventions are normally used:

cua#	for an ACU
cul#	for the associated modem line
ttyab	for a direct connect between systems a and b
tty#	another common convention for a direct connect

The modes of special files files should be 0666. Lines that are used to initiate conversations must not have a getty(2) process running on them at the local system. Also, when the system is configured, the tty lines must be set up correctly. Outgoing lines should ignore the carrier. Only incoming *modem* lines will wait for the carrier.

2.3. Installation of UUCP Software

These files and directories must be set up by the system *uucp* manager before any programs can be run:

LIB/USERFILE	defines *uucp* security
/etc/passwd	password file
LIB/L.sys	information needed to connect to a system
LIB/L-devices	devices used to connect to remote systems
LIB/L-dialcodes	dial code abbreviations
LIB/L.cmds	allowable remote execution commands
LIB/Makefile	*uucp* configuration and compilation
/etc/uucpname	local system name
spool directories	directories for depositing spooled files and temporary work files.

where,

LIB=/usr/lib/uucp

After the previous files have been created you can install the *uucp* commands using Makefile. NOTE: If the *uucp* software has already been installed (binary only systems) then the Makefile need not be modified.

2.3.1. USERFILE and UUCP Security

There are two mechanisms available to protect systems:

1) Normal UNIX system file access modes
2) The USERFILE

All UNIX system users should be well aware of method one. Each user should have their files properly protected. In its most insecure state *uucp* can grab files that are readable by *uucp* (for example, mode xx4). It can also overwrite files that are writable by *uucp*. Ultimately, the users are responsible for the security of their own files.

The USERFILE is the first line of defense from remote users. It provides three flavors of security:

- File access permission for remote and local users
- Log in security
- Remote execution permissions

Each line in the file has the format:

login,[sys] [X#] [c] path-name [path-name] ....

In this format

login	is the login name for a remote system or local user.
sys	is the name of the remote system and is optional (read following discussion)
X#	is the level of execution assigned to *sys* and is optional, except in the default entries, (See Subsection 2.3.1.2 for explanation of defaults).
c	is the optional call-back required flag
path-name	is a path-name prefix that *sys* can access

2.3.1.1. File Access Security with the USERFILE

The domain of accessibility of a remote system is restricted by the USERFILE. An entry should exist for each system that defines which paths the system can access. If no entries exist for a particular system the default entries will be used, (See Subsection 2.3.1.2 for explanation of defaults).

NOTE: The USERFILE must be set up with default entries that use this format:

```
remote, X#    /some_path_name_for_remote_systems
local, X#     /some_path_name_for_local_users
```

The letter X is a keyword used by *uucp* to identify the set of commands that a remote system may execute on the local system. In the following examples the X field is included for accuracy only; it is discussed in the section covering remote execution security. The words local and remote are also key words used by *uucp*.

remote - this entry is the default entry for remote systems that do not have an explicit entry in the USERFILE. Do not be too liberal with this entry. A typical path allowed for remote users would be:

```
remote, X1    /usr/spool/uucppublic
```

/usr/spool/uucppublic is a well known public repository. Users should not leave important files in that area.

local - this entry is the default entry for local users. Usually the normal UNIX system file access modes are all the security that is imposed on local users. Therefore, the typical default entry for local users is:

```
local, X9 /
```

Caution: The default entries must be supplied, otherwise the *uucp* software will fail.

In addition to the default entries, per-system entries may also be supplied. This provides the flexibility to give trustworthy systems less restrictive access to the local system. The following examples illustrate the alternatives:

Example 1:

```
remote, X1     /usr/spool/uucppublic
local, X9      /
max,systemX    /usr/sources
```

This example allows the remote system systemX, which has logged into the local system with the login name max, to access anything that has the prefix /usr/sources. All other systems can access the public directories only.

Example 2:

```
remote, X1    /usr/spool/uucppublic
local, X9     /
max, /usr     /usr/src/share
```

This example shows that ANY system or user that has successfully logged into the local system with login name max to access anything in or below /usr or /usr/src/share. Note that several systems could log in with this same login name so care should be take to restrict access rights appropriately. All other systems can access the public directories only.

2.3.1.2. LOGIN Security with the USERFILE

uucp tries to ensure that only legitimate systems log in to the local system. When a remote system logs in, it passes its node name to the local system. *uucp* crosschecks the node name and login name against the USERFILE. If an entry exists for that node name and the login name does not match the one specified in the USERFILE, the connection attempt will be aborted. The following example illustrates this situation:

example USERFILE:

```
remote, X1          /usr/spool/uucppublic
local, X9           /
max,systemY c       /usr
max,systemZ         /sys
blimpy,systemQ      /
nuucp,              /usr/spool/uucppublic
```

Scenario 1:

The competition across the street has unscrupulously obtained the login name blimpy and password for your *uucp* system. They successfully connect to the local *uucp* system. The local *uucp* then checks for a USERFILE entry for blimpy. It finds the entry and knows that only systemQ can connect with the login name blimpy. Since the nodename of the remote system is syszero, the connection attempt fails.

Scenario 2:

In this case the same competition stole the login entry for *nuucp*. Since no system name is provided in the USERFILE for the *nuucp* entry the connection attempt will succeed.

Scenario 3: (Default login security)

The situation may occur where a system logs in successfully and there is NO entry in the USERFILE that corresponds to either the login name or the remote node name. *uucp* handles this situation as follows:

> IF the file /usr/lib/uucp/INSECURE exists, the connection request is accepted. If /usr/lib/uucp/INSECURE does not exist, the connection request is rejected.

This option is provided so that the system manager need not supply an entry for every system that can log in. The default (remote) entry is used for systems that do not have an entry in the USERFILE. An example of this is where the system manager only wants to worry about two *uucp* passwords: one login for trustworthy systems and another login for the other systems (for USENET perhaps).

Here is a sample USERFILE.

```
remote, X1    /usr/spool/uucppublic
local, X9     /
safeuucp,     /
unsafeuucp,   /usr/spool/uucppublic
```

Since no system names are specified, connection attempts that use the logins *safeuucp* or *unsafeuucp* succeed. Attempts to connect with other logins succeed only if /usr/lib/uucp/INSECURE exists and then will only receive the default path security.

Scenario 4: (call back option)

If the system manager believes that it is possible to forge remote node names, the call back option can be used. In the USERFILE example, if a successful login is made from a node that claims to be systemY, *uucp* will reject the request and then try to call the real systemY. This is the most secure form of USERFILE entry. Note that if both the destination and local system use the call back option, an infinite loop of call backs can occur. Make sure this does not happen.

2.3.1.3. Remote Execution Security

A new security enhancement allows the system manager to restrict remote execution to specified systems only. Also, systems that are allowed to execute commands on the local system can be limited to a subset of allowable commands.

The X# field of a USERFILE entry is used to provide levels of security. The # can range from zero to nine where zero is the most secure level and nine is the least secure level. When the execution daemon (*uuxqt*) processes a remote execution request, it uses this algorithm:

> *uuxqt* obtains the security level from the USERFILE that corresponds to the remote system making the request. The command to be executed also has a level number. (See Subsection 2.3.6 for information about the L.cmds file). If the level associated with the remote system is greater than

or equal to the number associated with the command, then *uuxqt* grants permission to execute that command. Otherwise the remote execution request fails.

NOTE: You must specify the execute field in the default USERFILE entries; otherwise, the *uucp* software will fail. All other USERFILE entries need not have the execute level specified. For entries that don't have an execution level listed, then the default execution level will be used for that system. The default execution level for remote machines is obtained from the remote USERFILE entry. The local USERFILE entry provides the execution level to local users.

NOTE: Execution levels are provided on a machine-name basis only, not to particular users. Thus, you cannot use a USERFILE entry that specifies a login name but no system name to determine the execution level of a system that logs in with this entry. Instead, machines that do not have their own USERFILE entry will use the default *remote* USERFILE entry.

The example below illustrates remote execution security:

Assume the following USERFILE and L.cmds file.

USERFILE:

```
remote, X0              /usr/spool/uucppublic
local, X9               /
maxuucp,sysmax X3       /usr
xuucp,systemx X1        /usr/spool/uucppublic
yuucp,systemy X1        /usr/spool/uucppublic
zuucp,systemz           /usr/spool/uucppublic
nuucp,                  /usr/spool/uucppublic/stockroom
ruucp, X5               /usr/spool/uucppublic/stockroom
```

L.cmds file:

```
rmail X1
rnews X1
uusend X2
```

Explanation: The default *remote* entry prevents any system that does not have its own USERFILE entry from executing any of the commands in L.cmds. They can send/receive files from the public directories. The systems *sysmax, systemx*, and *systemy* can execute rmail and rnews. *sysmax* is the only remote system that can execute uusend. Any system that logs in as *nuucp* is provided the default execution level (which in this case is zero) and thus can not execute any command. The important

point to notice is that these latter systems (those that log on as *nuucp*) are not provided the default path security. Therefore the systems that log on as *nuucp* can only send/receive files to and from

/usr/spool/uucppublic/stockroom

Any system that logs on as *ruucp* has the same execute permissions as those that log on as *nuucp*. Since no system name is provided in the *ruucp* USERFILE entry, the system ignores the X field and uses the default execution level instead.

Local users can access all of the commands in L.cmds.

2.3.2. Setting Up the Password File

Any system that wishes to connect to the local system must have an entry in /etc/passwd. System managers must agree on the login and password to be used. A typical password entry uses this format:

login:passwd:uid:gid:useful info:/usr/spool/uucppublic:/usr/lib/uucp/uucico

The most important detail to note is that the last field is the path of the transfer daemon (*uucico*). When a remote system successfully logs in, *uucico* is run in the slave mode, and the dialog between the peer *uucicos* begins. It is good practice to have a USERFILE entry for every /etc/passwd entry, (See Subsection 2.3.1.2 on USERFILE security).

In addition to the latter /etc/passwd entries, an administrative login must be created for user: *uucp*.

2.3.3. The L.sys File

The L.sys file contains entries for each remote system that the local system can call. 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 format of each entry is:

system_name time device class phone login_sequence

Separate each field by blanks or tabs. Here are descriptions of the fields.

system name

The name of the remote system.

time

time is a string that indicates the days-of-week and times-of-day when the

system can be called, for example, MoTuTh0800-1740.

The day portion may be a list containing:

Su Mo Tu We Th Fr Sa

Day may also be *Wk* for any week-day or *Any* for any day.

Indicate hours in a range, for example, 0800-1230. If you do not specify a time portion, 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. Multiple date specifications, separated by | are allowed.

For example, Any0100-0600|Sa|Su means that the system can be called any day between 1 am and 6 am or any time on Saturday or Sunday.

An optional subfield is available to indicate the minimum time (minutes) before retrying a failed connection. A failed connection attempt is a log in failure as opposed to a dialing (connection) failure. The subfield separator is a comma (,). For example, *Any,9* means call any time but wait at least 9 minutes after a failure has occurred.

device

The device is either the *ACU* or the hard-wired device used for the call. For the hard-wired device, use the last part of the special file name, for example, tty2.

class

Class is the line speed for the call, for example, 1200. 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, for example, ct5900, nh6511, or an unabbreviated phone number. 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 this format:

expect[-sendspecial-expect] send ...

Expect is the string expected to be read when logging into the remote system, and *send* is the string to be sent when the *expect* string is received. If *expect* is not received, the subfield can be set up so that special characters (*sendspecial*) can be transmitted to the remote site. After the special characters are transmitted the *expect* following *sendspecial* is the next expected string. Two special characters which can be sent when *expect* is not received are *EOT* and *BREAK*. EOT will send an EOT character, and the string BREAK will send three break sequences (the break is simulated by using line speed changes and null characters). A number from one to nine may follow the BREAK. For example, BREAK1 will send 1 break. Note that after every send string a \r (carriage return) is sent except as noted below. If *sendspecial* is two consecutive dashes (--), a carriage return will be sent. In some instances it is necessary to send characters to the remote system before expecting something to arrive, for example, some systems want a carriage return before issuing a login prompt. A sequence of two double quotes ("") is useful in this regard. If "" is used as the expect string then nothing is expected and the following send string is transmitted. For example, "" \r\c will expect nothing, then send a carriage return. \c means the default \r should not be sent. If we did not put the \c here, two carriage returns would be transmitted.

These examples illustrate alternative expect-send sequences.

EXAMPLE 1:

login:--login xuucp ssword: foobaz

Explanation: *login:* is expected. When it is received, *xuucp* is sent. Now the word *ssword* is expected (the first letter of password varies from system to system so it is safer to look for the tail end, for example, *ssword*). When *ssword* is received *foobaz* is sent. If the login is successful, the conversation between the peer transfer processes (*uucico*) begins. If the login fails the connection attempt fails.

EXAMPLE 2:

login:--login: xuucp ssword: foobaz

Explanation: expect *login:*, if not received then send a carriage return and expect *login:* again. If receive *login* send *xuucp* and so on and so on.

EXAMPLE 3:

login:-BREAK1-login: xuucp ssword: foobaz

Explanation: expect *login:*, if not received send one break sequence (to change the baud rate of the remote *getty* process) and expect *login:*. Proceed as in the previous examples.

Other special characters can be used as part of the send sequence when talking to systems that are either slow or that need to be placed in a sane state before the true login sequence can begin. Here are the special characters and their meanings:

PAUSE[#]	pause # of seconds (or five seconds by default).
\d	pause one second before sending next character.
\s	send a blank character.
\r	send a carriage return.
\c	do not send a \r at the end of send string.
\b	send a break character.
\###	send the character represented by octal number ### (for example, \05 is control-e)
P_ZERO	change parity from even (default) to zero.
P_EVEN	change parity to even.
P_ODD	change parity to odd.
P_ONE	change parity to one parity.

Here are examples that illustrate the use of these special characters.

EXAMPLE 4:

"" @ login: xuucp ssword: foobaz

Explanation: expect nothing, send an @ character (line kill), send a carriage return, continue normal login sequence. The @ character is often useful for clearing out line noise before starting to log on. The default *line kill* character varies from system to system.

EXAMPLE 5:

"" P_ZERO ""\d\b "" \005\c login: xuucp ssword: foobaz

Explanation: expect nothing, change output parity to zero parity, expect nothing, delay one second then send a break sequence, expect nothing, send the escape character \005 without the default carriage return, continue with normal login sequence.

Putting all the fields together the following examples illustrate complete L.sys entries:

sys1 Any ACU 1200 wy7777 "" \r\d ogin:-EOT-ogin: Ufoobaz ssword: secret

sys1 Any ttyae 9600 tty32 "" \r\d ogin:-EOT-ogin: Ufoobaz ssword: secret

sys2 Any ACU 300 456-1234 "" \r ogin:-EOT-ogin:-EOT-ogin: Usys1 ssword: testing

sys3 Any,10 ACU 1200 8=123456789 login:-\r\c-login:-\r\c-login:-BREAK-login:-EOT-login: Uconn ssword: huskies

sys4 Any0000-0700 ACU 1200 8=987654321 login:-BREAK-login:-BREAK-login:-BREAK-login: Usys2 ssword: foobar

If the remote system uses nonstandard hardware the L.sys entry can be become complex. To connect to some systems you may have to alter the L.sys entries until a successful combination is found. Section 3. contains information which is useful for debugging L.sys entries.

NOTE: The L.sys file must also contain an entry for the name of any system that calls you, but you do not call them. The form of entry is abbreviated to system name and one word in place of the time entry, such as *never* or *incoming*.

2.3.4. The L-devices File

This file contains information about call units and direct connections. It is used to map specifiers in the L.sys file to specific devices. The format for each entry is:

type line call-unit speed brand

In this format:

type is a device type such as ACU or DIR. DIR indicates that this is a direct connect, hard-wired line.

line is the device for the modem line or hard-wired line as named in */dev*, for example, cul0 or ttyab. If the *type* is a DF02 or a DF03, (or any modem with an integral Auto Call Unit), this field will contain the automatic call device, for example, acu0. The special device files are assumed to be in the */dev* directory.

call-unit
is the automatic call unit associated with *line*, for example, cua0. Hardwired lines should place the device for the line in this field, for example, ttyab.

speed
is the line speed.

brand
is the brand name of the modem/acu. Here are the currently acceptable brands: dn11 (for Bell 801), DF02 or DF03 (for DEC modems), Ventel, Hayes, and Vadic. For direct connections, place the word direct should be placed in this field.

Here are typical *L-devices* entries:

```
ACU cua0 cua0 300 DF02
ACU cua1 cua1 1200 DF03
ACU cul3 cua3 1200 dn11
DIR ttyab ttyab 9600 direct
```

2.3.5. L-dialcodes

This file contains the dial-code abbreviations used in the *L.sys* file, for example, nh, boston. The entry format is:

```
abb dial-seq
```

In this format:

abb is the abbreviation as used in the L.sys file.
dial-seq is the dial sequence to call that location.

The entry **nh 603** would force any L.sys entry that used the prefix *nh* in the *phone* field, to send *603* to the dial unit before the rest of phone number is dialed.

2.3.6. L.cmds File

The L.cmds file is a list of commands that a remote system can execute via **UUX**. The format of the file follows:

```
command  X#
```

command is a unix command or application program. *X#* is the execution level associated with *command*. The # can range from zero to nine. If the X field is not present then nine is provided as the default level. If X is present but a level number is not specified then zero is assumed (therefore, any system can execute this command). Care should be taken to limit the number and type of allowable commands, (See Subsection 2.3.1.3 on remote execution security). A typical list of commands would be:

rmail X1
rnews X1
uusend X1

To specify the path that *uuxqt* will use to search for commands, add the following line to L.sys (anywhere in L.sys will do):

PATH=path1:path2:path3:path4: (and so on)

For example,

PATH=/bin:/usr/bin:/usr/ucb

will tell *uuxqt* to first try /bin for a command. If the command is not in /bin, then try /usr/bin and then /usr/ucb. If no PATH entry is supplied, the default: PATH=/bin:/usr/bin will be used.

2.3.7. Makefile

The Makefile may need to be modified to reflect the desired *uucp* configuration options.

Add or delete -D fields in the CFLAGS line to change the configuration. The following options are available:

NDIR
: If the operating system is a BSD version that is older than 4.1c, then the new directory system calls will have to be simulated. To simulate calls, create and install a directory library and header file (See Subsection 2.3.9 on installing *uucp* commands). By defining NDIR, the new header file (ndir.h) will be used instead of the standard directory header file (dir.h).

V7M11
: Use V7M11 if the operating system in use is ULTRIX-11. NDIR is automatically defined by the software if V7M11 is defined.

UNAME
: Use UNAME if the gethostname(2) system call is to be used to determine the local host name.

UUNAME
: Use UUNAME if the file /etc/uucpname is to be used to determine the local host name. If this option is chosen the system manager must modify /etc/uucpname to reflect the correct local host name.

NOTE: *uucp* requires that node names contain seven characters at most. Longer names will be truncated to seven characters.

UUSTAT

Use UUSTAT if the *uustat* is to be turned on.

NOTE: If *uustat* is turned on, two files must be created before the *uucp* commands can be used. They are:

/usr/lib/uucp/L_stat #machine status file

/usr/lib/uucp/R_stat #uucp request status file

L_stat and R_stat must be owned and readable by *uucp*.

NOTE: If L_stat and R_stat exist from a previous version of *uucp*, THEY MUST BE CLEARED OUT.

The new version of *uustat* uses a different format (binary as opposed to ASCII) for these files. If they are not cleared out, the *uucp* commands will eventually fail. To clear out old L_stat and R_stat files enter the following commands:

```
cp /dev/null /usr/lib/uucp/L_stat
cp /dev/null /usr/lib/uucp/R_stat
```

The default CFLAGS line in the Makefile is:

CFLAGS=-O -DUUSTAT -DUNAME

The *uustat* program is turned on and the gethostname(2) system call is to be used. For ULTRIX-11 systems the default CFLAGS line is:

CFLAGS=-O -DUUSTAT -DUUNAME -DV7M11

The *uustat* program is turned on, all ULTRIX-11 specific code will be compiled, and /etc/uucpname will be used to obtain the local node name. A complete list of make command options is given in the beginning of the Makefile, for example, make *install*, make *uucp*, ..., (See Subsection 2.3.9 for instructions on how to use the Makefile for installing the *uucp* commands).

2.3.8. Spool Subdirectories

Subdirectories must be created for various *uucp* files as follows:

SPOOL/TM. /* temporary files */

SPOOL/STST. /* connection status files */

SPOOL/sys /* system subdirectories */

The SPOOL is normally /usr/spool/uucp.

In addition, the *sys* subdirectory is further subdivided into directory names that correspond to specific remote systems and a default directory (DEFAULT). The minimum configuration requires that the SPOOL/sys/DEFAULT directory be created. Per system subdirectories must be created manually. To facilitate the latter operation the command:

/usr/lib/uucp/uumkspool *systemname*

will create the subdirectories for the specified system. Each system directory will contain the following subdirectories:

D.localname /* outgoing data files */

D.localnameX /* outgoing remote execution files */

D. /* incoming data files */

C. /* work files */

X. /* incoming remote execution request files */

If a large amount of traffic is expected between specific systems, then you should create subdirectories for those systems. If, after a period of time, it becomes necessary to create new system directories, files will have to be moved out of the DEFAULT directory to the new system spool directory. The command */usr/lib/uucp/uurespool* will ease this process. *uurespool* is described below.

2.3.8.1. Installing Subdirectories on New Systems

The Makefile includes shell scripts to create the needed subdirectories.

i. Use the following shell script to create the minimum set of directories:

Enter the directory that contains the *uucp* Makefile.
Then type:
```
cd /usr/lib/uucp
su root
make mkdirs
```

NOTE: some directories may already exist.

If the above does not succeed try the following:

```
su root
cd /usr/spool/uucp/
mkdir TM. STST. sys
chown uucp TM. STST. sys
chmod 0755 TM. STST. sys
cd /usr/spool/uucp/sys
mkdir DEFAULT
chown uucp DEFAULT
chmod 0755 DEFAULT
cd DEFAULT
for i in C. D.localname D.localnameX D. X.
do
        mkdir $i
        chown uucp $i
        chmod 0755 $i
done
```

In this entry *localname* is the *uucp* node name of the local system.

ii. To create per-system subdirectories, type:

```
/usr/lib/uucp/uumkspool sys1 sys2 ...
```

where sys1 sys2 ... are the names of the system subdirectories

All of the required subdirectories should now exist.

2.3.8.2. Installing Subdirectories on Old Systems

If subdirectories are to be installed on a system that has been running with a different subdirectory scheme, follow these steps:

i. Follow steps i.-ii. in the previous section.

ii. It is now necessary to move files from the old directory to the new subdirectories.

Use the command */usr/lib/uucp/uurespool* to move old spool files to new spool directories. The *-t#* option allows you to specify the type of spool that was used prior to installing the new *uucp* system.

will be 1, 2, 3, or 4 as follows:

1 - if all spool files are located in /usr/spool/uucp. This is the format of the old *uucp*s. There are no subdirectories.

2 - if the spool directory has been split out into several subdirectories: D.local D.localX D. X. C.

3 - if the spool directory has been split as in 2 with the addition of C./OTHERS and STST.

4 - if a new system directory has been created and the spool files have to be moved from DEFAULT to the new system directory.

Execute the following command line:

/usr/lib/uucp/uurespool -t#

2.3.8.3. Adding Additional Per System Subdirectories at a Later Time

If additional per system subdirectories are to be added at a later date, the new directory needs to be created and spooled files moved from DEFAULT to the new directory. Run the following commands:

```
/usr/lib/uucp/uumkspool sys1 sys2 sys3 ....
/usr/lib/uucp/uurespool -t4
```

2.3.9. Installing UUCP Commands

If your site has object code only, ignore this section. Binary only sites have to install directories and administrative files only.

Follow these steps to install the *uucp* executables:

i. This step is required by systems that do not have the new directory system calls, (for example, versions older than 4.2BSD, and ULTRIX-11). Execute the following shell commands:

```
cd /usr/src/usr.lib/uucp      (or wherever you've stored theuucp sources)
cd ndir
make -f Makefile.V7M11 install
cd ..
```

ii. For BSD versions older than 4.2, Makefile must be modified to include:

LIBNDIR=/usr/lib/libndir.a

Also, the CFLAGS line must include NDIR and UUNAME (See Subsection 2.3.7 for description of Makefile).

In all systems, to save the old commands and install new commands, type:

make save

To install new commands only, type:

make install

You are now finished installing *uucp*.

3. UUCP Administration and Maintenance

As stated in the introduction, *uucp* is not self administering. If the system is to be used extensively, then an assigned system manager does the daily monitoring. To some extent *uucp* cleans up after itself using **crontab** shell scripts.

If the system uses the supported hardware, most problems should either be administrative (for example, files not set up correctly, wrong phone numbers) or hardware failures. Problems with the software may still exist but they should be much more rare. The discussion below should help determine the source of the problem.

3.1. Administrative Files

These administrative files contain *uucp* statistics to use for diagnosing problems:

/usr/spool/uucp/LOGFILE

/usr/spool/uucp/ERRLOG

/usr/spool/uucp/SYSLOG

3.1.1. LOGFILE

The LOGFILE contains information logged by a transfer process (*uucico*) concerning a conversation with a remote site. If a problem occurs during the connection stage, it noted here. The entry format is:

login_name (time of transaction-process_number) message

Here are log messages that you might see when a problem occurs during the connection stage:

LOGIN FAILED

- The *uucico* got carrier but could not log in.

FAILED (call to some system)

- The connection attempt failed as a result of a previous message.

NO CALL (RETRY TIME NOT REACHED)

- You attempted to call a system too soon after previous failed attempt.

CAN NOT CALL (SYSTEM STATUS)

- The call to a remote system aborted because of previous connection status (for example, too many failed attempts to log in to remote system, or too soon after a previous failed connection attempt). Connection status information is kept in /usr/spool/uucp/STST. There is a separate STST. file for each remote system. If the STST. file, corresponding to the remote system, is removed then a call will be permitted to that remote system (immediately).

NO CALL (MAX RECALLS)

- The local system has failed to log in to the remote system MAX times (currently 20). No further attempts can be made by *uucico* until the STST.remote file is removed from /usr/spool/uucp/STST./ .

Direct line is already in use (device name)

- *uucico* determined that there was no LCK file for this device and tried to use it (as it should) but then found that some other process (most likely tip/cu) was still using the device.

ALREADY OPEN (device name)

- This also gets the log message, direct line is already in use, except this message is logged for ACUs.

FAILED (HSM no carrier)

- The carrier was not detected when using a Hayes Smartmodem.

READ ON df02/df03 (FAILED acu=/dev/cua2, char=102)

- A dial failure occurred while using a DF02 or DF03 acu. The DF0s return a character that indicates the nature of the failure. *char=* is that character. 102 is

an ASCII B that means that carrier was not detected within a prespecified amount of time (settable on the DF0s).

df02/df03 illegal return (FAILED acu=/dev/cua2, char=0)

- This message indicates that the specified acu was in a strange state, resulting in an unknown return character. This problem usually goes away when the current uucico process exits.

WRONG TIME TO CALL (systemname)

- An attempt was made to call a system at a time outside the range specified in the L.sys file.

NO DEVICE

- An attempt to call a system was made but no devices (acus or hard-wired lines) were available.

using device (device, fd=#)

- *uucico* is using *device* to call a remote system. *fd* is the file descriptor that corresponds to *device*.

LOCKED (call to systemname)

- An attempt to call a system was made for which a conversation was already in progress (for example, a lock file for that system exists in the spool directory).

FAILED (LOGIN VS MACHINE)

- A remote system tried to log in, but it failed the USERFILE security test (See Subsection 2.3.1).

TIMEOUT (DIALUP DN write)

- There was no carrier detect after dialing a number

FAILED (DIALUP ACU write)

- An error occurred while *writing* the phone number to the acu. If this error is constantly showing up it could be the result of an acu hardware failure. There might also be something wrong with the mode of the special device file.

TIMEOUT (systemname)

- A remote system has initiated a connection attempt and then stopped communication (reason unknown to the local site).

The following messages can occur at any time:

ret (#) from system!user (MAIL FAIL)

- A *mail* command failed. The exit status is indicated by *ret*. The originator is *user* on the remote node: *system*.

cmd: *command*; ret: signal #, exit # (CMD FAILED)

- A remote execution request failed. The *command* is the command that failed. The *signal* is the signal that caused the command to fail. The *exit* is the value returned by an *exit* system call in *command.*

XQT DENIED (command)

- A remote system tried to execute *command* but the request was denied by the local system.

cmd xqt'ing > 55 minutes (touch lock file)

- *uuxqt* has been executing a command for 55 minutes. The lock files associated with this command are refreshed so that they will not be removed by another *uuxqt* process.

command terminated - exceeded time limit (command)

- A command that is being processed by *uuxqt* has been running longer than 8 hours (approximately). The *command* is assumed to be a runaway process and is terminated.

system_name (execute level too low)

- The *system_name* was not allowed to execute a command because it did not have a high enough execute level. The specific command should be named by a subsequent LOGFILE entry. (See Subsection 2.3.1.3 about remote execution security).

The following messages can occur at any time:

CAUGHT (SIGNAL #)

- A signal was caught by *uucico* which caused it to terminate.

intrEXIT (signal: #)

- A signal was caught by *uucico*. This signal was probably the result of damaged software, an undetected bug, or some other system problem.

closed df0 type acu (fd=#)

- *uucico* has finished using a DF02/DF03 type of acu. *fd* is the file descriptor associated with the acu.

hasyes: closing (fd = #)

- *uucico* has finished using a Hayes Smartmodem. *fd* is the file descriptor associated with the acu.

could not close acu (errno = #, fd = #)

- *uucico* could not close the file descriptor (*fd*) associated with a DF02/DF03 type of acu. *error* is the system *errno* that was given as the reason for the failure. Consult Section 2 of the ULTRIX-32 User's Manual for the meaning of *errno.*

These messages occur during the file transfer stage after the connection has been made.

FAILED (CAN'T READ DATA)

- A work file (C.file) has specified a file to transfer but that file is not readable by *uucp* or does not exist.

FAILED (CAN'T READ SPOOLED DATA)

- A work file (C.file) has a reference to a spooled data file (D.file) and the spooled file is either unreadable by *uucp* or does not exist.

NOINPUT set no incoming files allowed (Remote name)

- /usr/lib/uucp/NOINPUT exists. The existence of this file is used as a flag to *uucico*. If NOINPUT exists do not allow incoming files.

ACCESS (DENIED)

An attempt was made by a remote system to access a file for which it does not have USERFILE permission.

REQUEST

The remote system might have run out of space. An attempt was made to write to a directory that was not writable by *uucp*, or some USERFILE restriction was violated.

DENIED (CAN'T OPEN)

A remote site tried to receive a file that the local *uucico* process can not access.

BAD READ# (expected *'char'* got *something_else*)

the local *uucico* process was waiting for a reply from the remote system but either the return message was corrupted or the remote process did not reply.

FAILED (CAN'T CREATE TM)

An attempt to create a temporary file failed, which may indicate that the system is running out of space.

FAILED (conversation complete)

A conversation with a remote system stopped before all of the spooled files were transferred. The reason for the failure is not known, but it could have been caused by something or someone killing the remote process, or possibly by a lost line. If a conversation has lasted about 90 minutes and the remote site is running an older version of *uucp*, the problem may have been the premature removal of a LCK.file at the remote site. This problem is cured in the current version.

Non error messages that occur in the LOGFILE.

Occasionally a message is placed in the LOGFILE that is not an error message. Some more error messages are:

REQUEST (char srcname dstname owner)

- A request to transfer a file to a remote site was made. If no followup message was made to indicate some kind of failure, the request was successful.

The *char* is the type of request. S for send, R for receive, X for remote execution.

The *srcname* is the name of the file on the local system.

The *dstname* is the name the file will have on the remote system.

The *owner* is the owner of the file.

REQUESTED (char srcname dstname owner)

- A remote site has requested to transfer a file to the local site. If no subsequent failure message then the transfer was successful.

OK (startup)

- The local and remote sites have successfully agreed upon what low level packet protocol to use to transfer raw data.

OK (conversation complete)

- All transactions with a remote system completed successfully.

uucp XQT (PATH.....;cmd)

- A request from a remote site to execute *cmd* was successful.

XQT QUE'D (cmd)

- A command (cmd) to be executed at a remote site was spooled.

QUE'D (S srcfile dstfile owner)

- A user request to transfer a file was spooled.

3.1.2. ERRLOG

The ERRLOG contains error messages that are less likely to occur during the normal operation of *uucp*. If a *uucp* support file such as the USERFILE is improperly set up the problem is noted here in the ERRLOG. If administrative problems such as this occur, the software will most likely stop executing, so it is important that administrative problems that are listed in the ERRLOG get cleared up quickly. When the system has run out of space preventing the creation of files or referenced files do not exist the problem will be noted here. Problems that occur during the transmission of raw data packets are also noted here. Packet problems occur infrequently and will not halt the *uucico* process. They are more indicative of a poor connection or over-loaded

system and sometimes may point to a problem in the hardware.

The format of an ERRLOG message is:

ASSERT ERROR (process name) process# time of entry message return code

where

ASSERT_ERROR
- is redundant information indicating that an ASSERT ERROR has occurred

process_name
- The process_name is the name of the process in which the error occurred. It could be *uucico*, *uucp*, *uuxqt*, *uux*, *uustat* or any other uucp related program.

process#
- The number is the process i.d.

time_of_entry
- The time_of_entry is the time the ASSERT error occurred.

message
- The message indicates the nature of the error.

assert_return_code
- The assert_return_code is a returned value which may be helpful for finding the source of the error (it generally is not though).

Some typical ASSERT messages are:

PKassert
- Any message that begins with PK came from the packet transmission software. Most likely due to noisy lines or a failure at the remote system.

NO default entry for remote machines
- The USERFILE does not have the default entry for remote machines.

NO default entry for local users
- The USERFILE does not have the default entry for local users.

xeq level undefined in USERFILE
- The remote execution security level (X#) was not specified for a default USERFILE entry, (See Subsection 2.3.1.3 for details on remote execution security).

CAN'T OPEN filename
- The named file probably does not exist.

WRONG ROLE
- During the file transfer stage the *uucico* process managed to reverse roles (from master to slave or vice versa) at the wrong time.

ARG COUNT<#

- Indicates that a work file (C.file) has been corrupted.

STAT FAILED filename

- A stat system call failed. If it happens continuously the named file is probably missing (and should not be).

BAD LINE

- A bad entry in the L-devices file has been encountered.

TOO MANY SUBDIRECTORIES

- The maximum number of per system subdirectories has been created. No more can be made.

TOO FEW LOG FIELDS

- The login/expect sequence of the L.sys entry for the remote system is incorrect.

BAD SPEED

- The desired line speed is not allowed.

RETURN FROM STTY

- An attempt to set the terminal I/O parameters of the outgoing line has failed. This could be either a hardware or a system problem.

BAD WRITE genbrk#

- An error occurred while generating a BREAK character on the outgoing line.

BAD DIRECTORY directory_name

- The named directory does not exist or is not set to the correct modes.

CAN NOT GET sequence file lock

- A lock file can not be created so that the sequence file (/usr/spool/uucp/sys/*sysname*/SEQF) can be accessed.

PERMISSION DENIED (Incoming C. file)

- *uucico* does not permit work files (C. files) to be transmitted to the local site because of security reasons.

3.1.3. SYSLOG

The SYSLOG records the number, size and source of each data transmission. Each entry has the following format:

uid rmte_sys (date) (int_time) sent/rec'd_data #b dur, Pk: # Rxmt #

where

uid	is the effective user id of the running process.
rmte_sys	is the name of the remote system where the data is sent or received.
date	is the date of the transaction including the time of day.
int_time	is the internal representation of the date.
sent/rec'd_data	indicates whether the data was sent or received from the remote site.
#b	is the number of bytes transferred.
dur	is the duration of the transfer in seconds.
Pk	is the number of packets needed to send the data.
Rxmt	is the number of packets that had to be retransmitted.

3.2. Guide to Debugging

When a connection to a remote system does not seem to be working, you may need to do some debugging. The amount and type of debugging may depend on whether or not the *uucp* source files are available. In any event the first place to look is the LOGFILE and ERRLOG. They may give some clue as to the nature of the problem. The system manager will have to determine if the problem is in the dialing stage, the login/handshaking stage, the file transfer stage, or whether it is a hardware problem. The most common errors occur when the remote site has set up its USERFILE incorrectly. An incorrect USERFILE results in messages sent back to local users saying that remote access to path files is denied. At any time the remote system could have changed its password or phone number. The system manager of the remote system should be contacted for updated information. When error messages constantly refer to one (out of many ACUs) the problem is probably a bad ACU. If the SYSLOG has recorded a lot of retransmissions, the hardware may be faulty, the communications line may be noisy, or the baud rate may be too high for the transmission media.

If the source files are available the local system manager should try initiating a conversation with the remote system in question. To initiate a conversation, start a *uucico* process in the MASTER mode as follows:

/usr/lib/uucp/uucico -r1 -x# -X# -s*system*

x#	is the debugging level. The higher the number the more debugging output. No packet level debugging is printed.
X#	is used to obtain packet level debugging output. The higher the number the more packet level debugging output.
system	is the system to be contacted.

r1 puts *uucico* into the master role.

Another option to *uucico* is **-f** (**f** for force). This will force *uucico* to start a conversation with a specified system regardless of the any previous connection status (as provided by the STST. files). Even if the source code is not available some useful information may be observed. The output of *uucico* follows the progress of the conversation. Debugging output from the slave *uucico* is placed in the file: AUDIT, in the spool directory, at the remote site. The output tends to be less meaningful than the LOGFILE unless the source code is available to help interpret the messages. No detailed explanation of the messages is given here since the messages are likely to change from version to version.

3.3. UUCP Self Administration Using CRONTAB Shell Scripts

The *crontab* should be modified to include shell scripts that are executed on an hourly, daily, and weekly basis. The purposes of the shell scripts are:

1. Poll remote systems for files to transfer.

Some sites do not have call units. You should poll these sites periodically to receive files from them.

2. Cleanup any temporary files.
3. Delete *uucp* transfer requests that could not be transmitted within a specified time period (currently one week).

The shell scripts are located in */usr/lib/uucp* and are discussed below:

uucp.day

This script is run at the start of the day. It cleans any lingering temporary files, saves the the previous days LOGFILE and SYSLOG in LOGFILE.yesterday and SYSLOG.yesterday, contacts any systems listed in /usr/lib/uucp/LIST.DAY, and then starts a general poll of all system for which there is work. Messages indicating the progress of this shell script are placed in /usr/lib/uucp/LOG.shells

The system manager should modify LIST.DAY so that the correct system are called. If LIST.DAY is empty or nonexistent only the general poll will be performed.

uucp.hour

This script initiates a general poll every hour. Messages indicating the start and finish of the poll are sent to the console. If any systems are listed in LIST.HOUR those systems will be contacted on an hourly basis.

uucp.noon

This script will contact any system listed in LIST.NOON at noon time.

uucp.night

uucp.night will clean up the spool directories in the early morning hours via

uuclean. Any spool files older than 168 hours are removed. The time limit can and should be adjusted by the system manager to conform to local conditions. After the cleanup, any systems listed in LIST.NIGHT will be contacted. A general poll contacts any remaining systems for which requests are still queued. Shell script messages will be sent to LOG.shells.

uucp.week
- LOG.shells is saved in LOG.shells.week and a general poll is started.

Here are typical *crontab* entries:

```
30 * * * * su uucp < /usr/lib/uucp/uucp.hour
0 12 * * * su uucp < /usr/lib/uucp/uucp.noon
0 6 * * * su uucp < /usr/lib/uucp/uucp.day
0 1 * * 5 su uucp < /usr/lib/uucp/uucp.week
```

3.4. Monitoring the Network

The cleanup shell scripts keep a well tuned network clean of any nontransferrable or miscellaneous files. There are times when the network starts to backlog due to some of the reasons mentioned in the previous section. It is necessary therefore, for the system manager to regularly monitor the network. Two programs are useful in this regard: *uumonitor* and *uustat*.

3.4.1. UUMONITOR

/usr/lib/uucp/uumonitor is a program that creates a snapshot of the *uucp* system. The format of the output is as follows:

system_name #C #X most_recent_status CNT:# time

where

system_name
is the remote system for which the entry applies.

#C is the number of C.files queued for the remote system.

#X is the number of requests for remote execution from the remote system.

most_recent_status
is the result of the most recent attempt to connect to the remote system.

CNT:#
is the number of times that a failure to login to the remote system has occurred. This does NOT include the number of failed dial attempts.

time is the time of the last status entry was made for this system.

This command is helpful for detecting systems that have backlogs, that have gone away for awhile, have changed phone numbers, etc. The CNT: field is useful for detecting a system whose login/passwd has changed. If CNT: gets larger than the maximum allowable failures (currently 20) no further attempts to connect will be made to this system. If the number of C.files queued starts getting unusually large (depending on the system, *large* could mean anywhere from 100-1000) try to determine the cause of the backlog. If a separate subdirectory does not exist for the backlogged system, create the subdirectory should and move the spooled files from the DEFAULT directory to the per-system subdirectory. When the problem is resolved, make an entry in /usr/lib/uucp/LIST.HOUR to help clear out the backlog.

3.4.2. UUSTAT - UUCP Status Inquiry and Job Control

The *uustat* command is also useful for monitoring the network. *uustat* displays the status of, or cancels previously specified *uucp* commands, or provides general status on *uucp* connections to other systems.

NOTE: In the current implementation *uux* requests on not recorded in the *uustat* logging files. This implies that *mail* and *news* requests are not recorded by *uustat*. Some of the options are:

-**m***mch* Report the status of accessibility of machine *mch*. If *mch* is specified as **all**, then the status of all machines known to the local *uucp* are provided.

-**k***jobn* Kill the *uucp* request whose job number is *jobn*. The killed *uucp* request must belong to the person issuing the *uustat* command unless that person has superuser privilege.

-**c***hour* Remove the status entries which are older than *hour* hours. This option can only be executed by the user *uucp* or the superuser.

-**u***user* Report the status of all *uucp* requests issued by *user*.

-**s***sys* Report the status of all *uucp* requests that are destined for remote system *sys*.

-**o***hour* Report the status of all *uucp* requests that are older than *hour* hours.

-**y***hour* Report the status of all *uucp* requests which are younger than *hour* hours.

-**j***all* Report the status of **all** *uucp* requests or the specified job (request) number.

-v Report *uucp* status in English words rather than code. If this option is not specified, a status code is printed with each *uucp* request.

When no options are given, *uustat* prints the status of all *uucp* requests issued by the current user. Note that only one of the options **-j, -m, -k, -c,** can be specified along with the remaining options. For example, the command

```
uustat -usteve -slimbo -y63 -v
```

will print the status, in English, of all *uucp* jobs issued by user *steve* that were destined for system *limbo* within the last 63 hours. The format of each job status entry is:

job# user destination spool_time status_time status

The *status* may be either an octal number or a description using English words. The octal code corresponds to this description:

OCTAL	STATUS
00001	the copy failed for unknown reasons.
00002	permission to access local file is denied.
00004	permission to access remote file is denied.
00010	bad *uucp* command is generated.
00020	remote system cannot create temporary file.
00040	cannot copy to remote directory.
00100	cannot copy to local directory.
00200	local system cannot create temporary file.
00400	cannot execute *uucp*.
01000	copy succeeded.
02000	copy finished, job deleted.
04000	job is queued.

The format for the machine accessibility status entries is:

system status_time last_success_time status

where

system
is the system in question

status_time
is the time the last status entry was made.

last_success_time
is the time of the last successful connection to this system. Note that this does not imply that the entire session completed. It does mean the transfer daemons successfully completed the handshaking phase and were able to begin transferring files.

status
is a self-explanatory description of the machine status.

3.4.3. Periodic Maintenance

Periodically it may be necessary to compact the spool directories.
The command: */usr/lib/uucp/uucompact* can be used to compact *uucp* spool directories. If *uucompact* is not available the following shell script outlines the procedure to pack any directory:

```
mkdir tempdir
chown uucp tempdir
mv directory/* tempdir
rm directory
mv tempdir directory
```

The **tempdir** is a temporary directory

The **directory** is the directory to be packed.

APPENDIX - Summary of Enhancements and Quick Installation Guide

Enhancements

Dialers

Dialers that will work with this version are: Ventel MD212, Hayes Smartmodem, DF02/DF03, Bells 212/801, Racal Vadic 3450.

Subdirectories

/usr/spool/uucp/sys contains per system subdirectories and a DEFAULT directory. This is a huge help on busy systems.

/usr/lib/uucp/L.cmds

List of commands permitted for remote execution. A line of form 'PATH=...' sets the search path. Execution levels must be assigned to each command.

expect-send sequence

Escape characters now permitted: \r, \n.
\r, not \n, is default char sent at end of string.
\c (put at end of string). Do not send ending \r.
\d pause 1 second (\d\d pauses 2 seconds)
"" P_ZERO 'expect nothing, start sending zero parity.'
P_EVEN (default), P_ODD, P_ONE other parity modes.
\05 Send a control-E
"" "" 'expect nothing, send a \r'.

uupoll [sysname]

Polls named system.

uumonitor

Displays spooled files, and pending *uuxqts*.

uuxqt

uuxqt can operate on a per command type basis by using -c option, for example,

uuxqt -cname_of_command. It will default to a general poll of commands. Several *uuxqts* can now run concurrently.

uuclean

The -s option has been added. -sALL will clean all system subdirectories. -ssystem_name will clean spool directories belonging to system_name. *uuclean* will only mail back files if the -m option is used. Otherwise results will be mailed to *uucp*.

uucp

The -W option prevents expansion of file names that reside on remote systems. Normally files names are prepended with the current working directory if the full path of the file was not specified.

uumkspool

Creates all subdirectories for the specified system(s).

uurespool

Move files from old spool directories to new spool format. Also useful for moving files out of DEFAULT into newly created per system spool directory.

uucompact

Compact spool directories. -sALL or -sname_of_system. The *uucp* subsystem must be inactive for this to function properly. Single user is best. If *uucompact* is stopped it can be started again, picking up where it left off.

uucico

The -f option has been added. -f forces *uucico* to override any previous connection status as provided by the STST. files. It is no longer necessary to remove a STST.file to ensure that a connection attempt is made.

USERFILE

The format of the USERFILE has changed to increase security and make it more legible.

Debugging

The -x option of the uu commands has been split out to -x and -X. -X provides packet level output, (for example, from the pk routines). -x provides all other debugging output.

This version runs on all VAXs and PDPs under ULTRIX-11 and 4.1BSD, 4.2BSD.

uucp installers should read the two documents (by Dave Nowitz) in Volume 2B of Version 7 manuals. They should then read the *UUCP Installation and Administration Guide* (this document). Understand each step below before executing. Some steps will vary slightly from system to system.

INSTALLATION

If you are starting with a new binary only ULTRIX-32 kit you only need to perform steps 6, 7, 11, and 12.

1. If you are currently running *uucp*, save the old programs:
   su root
   cd directory_where_makefile_resides
   make save

2. Editing Makefile and uucp.h - you should skip this step if you are running ULTRIX-32 and are not adding/changing source code.

   non-ULTRIX-32:

   a) Sites need to install the Berkeley directory reading library. Try (cd uucp_source_directory; cd ndir; make install). Edit Makefile to have LIBNDIR= -lndir define NDIR in uucp.h.
   b) Check LDFLAGS, OWNER, GROUP, and LIBUUCICO.
   c) Pick a method to allow *uucp* to know its system: Check out UNAME/UUNAME/WHOAMI/CCWHOAMI in uucp.h
   d) Define SYSIII if appropriate in uucp.h.
   e) Your *make* may fail because the Makefile is so large. If so, in /usr/src/cmds/make/defs, change

3. Make the new commands: (Binary only sites should skip this step.)

   make

4. WAIT UNTIL THE UUCP SYSTEM IS IDLE!! Single-user is best.

   su root (it is important that chmod and chown work below)

5. Install the new commands: (Binary only sites skip this step.)

    make install

6. Edit and install the control files:

    Edit and install into /usr/lib/uucp if necessary USERFILE, L.cmds, L.sys, L-devices, L-dialcodes.

    NOTE: These files must be owned by the same owner and group as the *uucp* commands and *uucp* spool files. The format for dialers is slightly different so that any dialer can be handled.

7. Make new subdirectories:

    For safety: cd /usr/spool/uucp; tar c . (Save queued files on tape.) The following assumes your site name is produced by 'uuname -l'.

    cd directory_that_contains_Makefile (/usr/lib/uucp on binary kits)

    make mkdirs

    This will make all spool directories including the DEFAULT system spool directory: /usr/spool/uucp/sys/DEFAULT.

    For each additional site that will have its own subdirectory:

    /usr/lib/uucp/uumkspool system

8. Move old queued files:

    If you have spooled files, they must be moved into the appropriate subdirectories. Assuming all spool files are in /usr/spool/uucp, (for example, you did not have subdirectories before), the following command will move the spool files to the new system directories:

    /usr/lib/uucp/uurespool -t1

    For each system directory that was created with uumkspool the following actions occur (assuming the name of your system is duke):

    Files beginning C. are put in the C. subdirectory.
    Files beginning with D.dukeX are put in that directory, not D.

Files beginning with D.duke are put in the D.duke directory.
All other D.files are put in the D. directory.
X.files are put in the X. directory.

Delete other old directories if you had any (for example, LOG).

9. Compact spool directories:

   From time to time it may be necessary to compact spool directories. The following command facilitates this:

   /usr/lib/uucp/uucompact -sALL or -sname_of_system

   NOTE: The *uucp* subsystem should be quiescent before and during the execution of *uucompact*.

10. Clear out status files:

    cd /usr/lib/uucp
    cp /dev/null L_stat
    cp /dev/null R_stat

11. Test the new system.

    Test by mailing a letter somewhere and back (this assumes you have mail working properly of course). If it works, the new system is probably fine. Otherwise, figure out what is wrong. Start by examining LOGFILE. Try /usr/lib/uucp/uucico -r1 -sname -x7 If things are no-go, you can back out the changes by restoring the old uu programs and the spooled files.

12. Install administrative scripts (uucp.hour, uucp.day, ...) as described in this manual. These scripts should be run using *cron*.

HOW TO ORDER ADDITIONAL DOCUMENTATION

DIRECT TELEPHONE ORDERS

In Continental USA and Puerto Rico call **800–258–1710**

In Canada call **800–267–6146**

In New Hampshire, Alaska or Hawaii call **603–884–6660**

DIRECT MAIL ORDERS (U.S. and Puerto Rico*)

DIGITAL EQUIPMENT CORPORATION
P.O. Box CS2008
Nashua, New Hampshire 03061

DIRECT MAIL ORDERS (Canada)

DIGITAL EQUIPMENT OF CANADA LTD.
940 Belfast Road
Ottawa, Ontario, Canada K1G 4C2
Attn: A&SG Business Manager

INTERNATIONAL

DIGITAL EQUIPMENT CORPORATION
A&SG Business Manager
c/o Digital's local subsidiary
or approved distributor

Internal orders should be placed through the Software Distribution Center (SDC), Digital Equipment Corporation, Northboro, Massachusetts 01532

*Any prepaid order from Puerto Rico must be placed with the Local Digital Subsidiary:
809–754–7575

Reader's Comments

Note: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement. _____

_____

_____

_____

Did you find errors in this manual? If so, specify the error and the page number.

_____

_____

_____

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code or Country _____

Do Not Tear - Fold Here and Tape

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO 33 MAYNARD MASS

POSTAGE WILL BE PAID BY ADDRESSEE

Documentation Manager
ULTRIX-32™ Documentation Group
MK02-1/H10
Continental Blvd.
Merrimack, N.H.
03054

Do Not Tear - Fold Here and Tape

Cut Along Dotted Line

Notes:

Notes:

Notes:

Notes:

Notes:

Notes:

Notes:

Notes:

Notes:

Notes:

Notes:

Notes:

Notes:

Notes:

digital™