TEK DESIGN AUTOMATION SOFTWARE

FLEXIBLE SOLUTIONS FOR A FUTURE OF CHANGE

Microcomputer Development Software

TNIX

UNICOM

Users Manual

For 8560 Series Host

Tektronix®

COMMITTED TO EXCELLENCE

TEK USERS MANUAL

Part No. 070-4536-00
Product Group 61

8560 Series Multi-User Software Development Unit

TNIX

UNICOM

This manual supports the following TEKTRONIX product:

8560U05

This manual supports a software module that is compatible with:

TNIX Version 1 (8560)
TNIX Version 2 (8560 Series)

Please check for change information at the rear of this manual

First Printing FEB 1982
Revised OCT 1983

Tektronix®
COMMITTED TO EXCELLENCE

ABOUT WARRANTY AND SUPPORT FOR THIS PRODUCT

This product is provided by Tektronix as category C software.

NOTE

Licensed Software for which the software support is specifed as Category C is furnished without warranty of any kind, and without any representation regarding quality, performance, or suitability.

TEKTRONIX SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Tektronix' liability for damages, if any, whether based upon contract, negligence, strict liability in tort, warranty, or any other basis, shall not exceed the fee paid by the Customer for the Licensed Software.

Category C software is provided on an "as is" basis. Any software services, if available, will be provided at the then current charges.

LIMITED RIGHTS LEGEND

Software License No.

Contractor: Tektronix, Inc.
Explanation of Limited Rights Data Identification Method
Used: Entire document subject to limited rights.

Those portions of this technical data indicated as limited rights data shall not, without the written permission of the above Tektronix, be either (a) used, released or disclosed in whole or in part outside the Customer, (b) used in whole or in part by the Customer for manufacture or, in the case of computer software documentation, for preparing the same or similar computer software, or (c) used by a party other than the Customer, except for: (i) emergency repair or overhaul work only, by or for the Customer, where the item or process concerned is not otherwise reasonably available to enable timely performance of the work, provided that the release or disclosure hereof outside the Customer shall be made subject to a prohibition against further use, release or disclosure; or (ii) release to a foreign government, as the interest of the United States may require, only for information or evaluation within such government or for emergency repair or overhaul work by or for such government under the conditions of (i) above. This legend, together with the indications of the portions of this data which are subject to such limitations shall be included on any reproduction hereof which includes any part of the portions subject to such limitations.

RESTRICTED RIGHTS IN SOFTWARE

The software described in this document is licensed software and subject to **restricted rights**. The software may be used with the computer for which or with which it was acquired. The software may be used with a backup computer if the computer for which or with which it was acquired is inoperative. The software may be copied for archive or backup purposes. The software may be modified or combined with other software, subject to the provision that those portions of the derivative software incorporating restricted rights software are subject to the same restricted rights.

ABOUT THIS MANUAL

This manual describes how to use the UNICOM software package with your 8560 Multi-User Software Development Unit. This manual contains the following sections:

Section 1, Introduction. Provides an overview of the UNICOM software package.

Section 2, Command Dictionary. Discusses the TNIX user commands that make up the UNICOM package.

Section 3, Installation. Explains to the system manager how to install the UNICOM package from the UNICOM installation disk.

Section 4, UNICOM Maintenance. Explains to the system manager how to perform the necessary software maintenance required by UNICOM and explains the UNICOM administrative commands.

Section 5, Technical Notes. Provides miscellaneous technical information to the system manager including the description of the cable adapters needed for connecting systems.

Section 6, Bell Labs Documentation Contains two articles with technical and general information on the UNIX communication commands. Most information in these articles also applies to the UNICOM commands. The portions of the articles that do not apply to UNICOM are noted.

Section 7, Index Contains an alphabetized list of major terms and concepts, giving the section and page number where they are discussed.

For information about the 8560 MUSDU and its TNIX operating system, refer to the 8560 MUSDU System Users Manual.

TABLE OF CONTENTS

Page

Section 1 Introduction

Product Overview 1-1
UNICOM Features 1-2
UNICOM Applications 1-3
Network Configuration 1-4
How UNICOM Works 1-5
Requirements for UNICOM 1-6
List of Commands 1-7
Source of Documents 1-8

Section 2 Command Dictionary

Introduction 2-1
CU -- Call UNIX 2-2
MAIL -- Send or Receive Mail Among Users 2-5
UUCP -- UNIX to UNIX Copy 2-7
UUNAME -- Name Remote Systems Connected to Local System 2-10
UUX -- UNIX to UNIX Command Execution 2-11

Section 3 Installation

Introduction 3-1
Installation Tasks 3-2
Network Overview 3-2
Installing UNICOM 3-7
Preliminary Steps 3-7
Using the Installation Disk 3-9
Using the Uucp-config Program 3-12
Modifying and Correcting Errors in the Installation 3-26
DEINSTALL Program 3-28
Adding a New System to an Existing UNICOM Network 3-29
Installing the Cu Command 3-30
Testing the Installation 3-31
Troubleshooting 3-32

Section 4 UNICOM Maintenance

Introduction 4-1
Checking and Restoring the UNICOM Commands 4-3
Maintaining the Uucp Software on Existing Systems 4-4
Modifying Current UNICOM Systems and Adding New Systems 4-5
Security 4-7
UNICOM Administrative Commands 4-9
UUCLEAN - Uucp Queue Directory Clean-up 4-10
UULOG - Uucp Administrative Program for Reading LOGFILE Information 4-11
UUPHONE - UUCP Phone Dialing Program 4-12

Section 5 Technical Notes

Note 1 -- Cabling Connections 5-1
Note 2 -- UNICOM and 8560 System Performance 5-3
Note 3 -- Notes on Uucp 5-4

Section 6 Bell Labs Documentation

Introduction 6-1
UNICOM - UNIX Differences in Bell Labs Documentation 6-2
A Dial-Up Network for UNIX Systems 6-2
Uucp Implementation Description 6-3

Section 7 Index

Section 1

INTRODUCTION

Page

Product Overview 1-1
UNICOM Features 1-2
UNICOM Applications 1-3
Network Configuration 1-4
How UNICOM Works 1-5

Requirements for UNICOM 1-6

List of Commands 1-7

Source of Documents 1-8

ILLUSTRATIONS

Fig. No. **Page**

1-1 A typical UNICOM communications network 1-4

TABLES

Table No. **Page**

1-1 UNICOM Commands 1-7

Section 1

INTRODUCTION

PRODUCT OVERVIEW

UNICOM (UNIX COMmunication) is a TNIX category C software package that permits communication and resource-sharing between connected TNIX--TNIX and UNIX(*)--TNIX systems. Systems may be connected through a cable or through telephone lines using a modem and an autodialer. Each system may be connected to one or more systems, which in turn may be connected to other systems.

The UNICOM communication commands are compatible with the UNIX commands of the same name and have similar syntax. The UNICOM **mail** command can send and receive messages with the UNIX **mail** command. You may create a computer communication network with the UNICOM software package.

To use this manual, you should be familiar with the TNIX operating system as described in the 8560 MUSDU System Users Manual.

UNICOM and the Uucp Command

The UNICOM software package consists of several commands used for communication and several commands used for maintenance of the UNICOM software. All the commands (except for **cu**) utilize or serve the **uucp** command. Many references in this manual refer to to the **uucp** command. These references are to be understood as applying to the other UNICOM commands also, unless otherwise noted.

This section includes the following topics:

- UNICOM Features. Explains in general how UNICOM is used and what the UNICOM commands do.
- UNICOM Applications. Gives some typical uses for UNICOM.
- Network Configuration. Describes a typical UNICOM network.
- How UNICOM Works. Explains how the UNICOM software functions.
- Requirements for UNICOM. Tells you what you need to use UNICOM.

(*) UNIX is a trademark of Bell Laboratories

UNICOM FEATURES

No special hardware is needed to connect TNIX--TNIX or TNIX--UNIX systems. New computers can easily be added to a network supported by the UNICOM package. TNIX and UNIX systems may be connected in any combination, either via a hard-wired link or via telephone lines using a modem and an autodialer.

UNICOM intersystem communication is performed as a background process (except for the **cu** command). Once you have issued a request for communication, (for example, a file copy between systems) you may immediately do other work. Any number of users can issue communication requests simultaneously. UNICOM will wait in the background until both systems are ready to communicate and then automatically complete its transfer of data.

An interactive program, **uucp-config**, is included to help you with installing and maintaining UNICOM.

UNICOM allows great flexibility in communication and resource-sharing between connected systems:

- Files may be copied from any system to another system. Several users may make file copy requests simultaneously. File copy requests are automatically queued and processed sequentially.

- Any system may cause another system to execute commands, using files in any system as input and output.

- A user on one system may log in directly into another system and use the other system's resources in real-time.

- A user on one system can transfer files between his system and another system in real-time with the **cu** command. File transfer requests are not queued with this command, but are processed immediately.

- Mail may be sent from one system to another. Mail may be routed to one system via other systems in network fashion.

UNICOM APPLICATIONS

UNICOM software is well suited for multi-system software project applications. Once the computers in the project are made into a UNICOM network, UNICOM can facilitate several tasks for your project:

- New versions of of software can be distributed easily and quickly and software tools can be exchanged.
- Data of all kinds can be shared.
- The project workload can be shared between systems. Design groups can easily pass messages, documents, test data and software between systems. If one system is busy, another system may be used to execute a program.
- One system may be used as a central project control system. The central system can maintain specifications, implement version control, build and provide master versions, and archive software.
- Resources existing on only some of the systems of the network can be shared by the whole network. For example, special peripherals on only one system may be shared by all.

NETWORK CONFIGURATION

A UNICOM communication network has many possible configurations. Each TNIX or UNIX system may be connected to several other TNIX or UNIX systems. Figure 1-1 shows a possible network configuration between TNIX and UNIX systems.

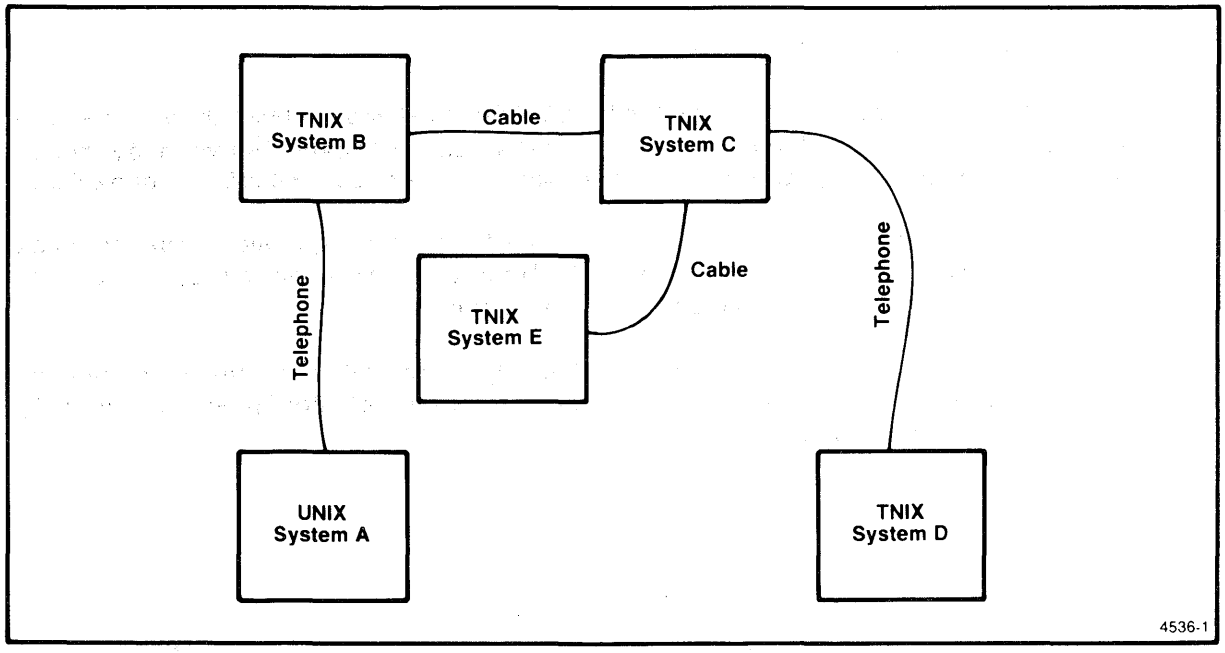


Fig. 1-1. A typical UNICOM communications network.

This figure shows the connections between TNIX and UNIX systems for a typical communication network created with the UNICOM software package.

Directly and Indirectly Connected Systems

Two systems are directly connected if there is a physical communication link between them (for example, an RS-232-C cable or telephone line).

Two systems are indirectly connected if there is some path of one or more directly connected systems between them. For example, two systems that are directly connected to a third system are indirectly connected to each other. Refer to Figure 1-1 for an example of directly and indirectly connected systems. In this figure, systems A -- B, B -- C, C -- D, and C -- E, are directly connected. Systems A -- C, A -- E, A -- D, B -- D, and B -- E are indirectly connected.

The UNICOM **mail** command permits messages to be sent between directly and indirectly connected systems. When used singly, the **uucp**, **uux**, and **cu** commands permit communication only between directly connected systems. However, these commands can be combined to communicate between indirectly connected systems. For example, the **uux** command may be used to execute the **uucp** command on another system. These commands are explained in detail in the Command Dictionary section of this manual.

Local and Remote Systems. The term local system refers to the system on which you are using UNICOM. The term remote system refers to a system that is directly or indirectly connected to your local system and that is part of your UNICOM communication network.

HOW UNICOM WORKS

The following paragraphs briefly describe how UNICOM works.

With the **cu** command, your local terminal behaves as though it is one of the terminals of the remote system, allowing you to execute commands on the remote system. With this command you can also transmit files immediately between your system and the remote system. Only one user at a time can use this link however, and there is no error checking during data transmission.

The UNICOM commands **uucp**, **uux**, (and **mail** to a remote system) place all requests for intersystem communication on a queue. The queue consists of a series of files in the /usr/spool/uucp directory. Queue files contain instructions and data for each communication request: sending files to other systems, executing commands on behalf of other systems, requests for other systems to send files, and requests for other systems to execute commands. **Uucp** works its way through the queue following the instructions in the files of the queue. Each user's communication request waits its turn in the queue. Although intersystem communication does not take place immediately, this queueing allows many users to use the same link at the same time. The following three paragraphs also apply to the **uucp**, **uux**, and **mail** commands.

Master and Slave Systems. Each direct connection between two systems forms a master-slave pair. The master system initiates communication to the slave system. The slave must wait for the master before information can be exchanged. The master and slave do not communicate at all times but only periodically. When the master has information to exchange with the slave, the master calls the slave through the link between the two systems. The two systems go through a handshaking protocol to establish a connection that includes error checking. Once the connection is established, the master and the slave can instruct each other to send files and execute commands until all the pending queued communication requests between both systems are finished. Each individual system may be master and slave at the same time to different systems.

Polling Systems. Even if no requests for communication are issued to the master, the master will periodically check or **poll** the slave to see if it has any requests for communications. The times at which a master calls ("polls") a particular slave are set by the system manager of the master system. Thus, if a slave system is not functioning when polled, it will be polled again later.

Throughput. The time taken for a particular communication request to be honored depends on many factors: the frequency of polling of a particular system, the length of both queues, the CPU loading of both systems, the data transmission rate over the link, and others. The **uucp** command can send mail to the user who initiates the request, indicating when the communication has taken place.

The article reprinted in the Bell Labs Documentation section of this manual, "A Dial-Up Network of UNIX Systems", provides an overall view of a computer network implemented using the UNIX versions of the UNICOM commands, and explains some typical applications of a network.

REQUIREMENTS FOR UNICOM

The UNICOM package may be installed on any version of TNIX on any 8560. The UNICOM software communicates with UNIX systems using a software protocol compatible with UNIX **uucp**, **cu**, and **uux** command formats. Any version of UNIX that supports these three commands will communicate with TNIX systems using UNICOM.

The system manager of the UNIX installation will have to modify the **uucp** files in the UNIX system to extend recognition to the TNIX system.

The correct cabling connections required by UNICOM must be observed when connecting systems. Refer to "Note 1 -- Cabling Connections" in the Technical Notes section of this manual for more information.

Communication between two systems via telephone lines requires an autodialer-modem on one system and a modem with autoanswer on the other.

A master system (to use the **uucp**, **uux**, and **mail** commands) must dedicate a separate port for each directly connected slave system connected via hard-wired link. **Cu** command communication requires a separate dedicated port on

the master system. One port dedicated to autodialer use may communicate with many remote systems.

LIST OF COMMANDS

Table 1-1 lists the commands included in the UNICOM package, along with a brief description of the command's function, and a reference to more detailed information about the command elsewhere in this manual.

Table 1-1
UNICOM Commands

Command	Description	Reference
cu	Permits a log in to a directly connected system through your system. Transfers files in real-time (no queueing) and can use the directly connected system facilities.	Section 2
mail	Sends messages between users of directly and indirectly connected systems.	Section 2
uucp	Copies files between directly connected systems.	Section 2 and Section 6 (both articles)
uuname	Lists names of directly connected systems.	Section 2
uux	Executes commands on directly connected systems	Section 2 and Section 6
uucico	Used by system manager and uucp program	Sections 3 and 4, and 6 (Uucp Implementation article).
uuclean	Used by system manager	Section 3, 4 and Section 6 (Uucp Implementation article).
uulog	Used by system manager	Section 4 and Section 6 (Uucp Implementation article).
uuphone	Used by system manager and uucp program	Sections 3 and 4

SOURCE OF DOCUMENTS

The reference documents contained in Section 6 of this manual are reprinted by permission of Bell Laboratories.

Section 2

COMMAND DICTIONARY

Page

Introduction 2-1

CU -- Call UNIX 2-2

MAIL -- Send Or Receive Mail Among Users 2-5

UUCP -- UNIX to UNIX Copy 2-7

UUNAME -- NAME REMOTE SYSTEMS CONNECTED TO LOCAL SYSTEM 2-10

UUX -- UNIX to UNIX Command Execution 2-11

Section 2

COMMAND DICTIONARY

INTRODUCTION

This section explains the user commands included in the UNICOM software package. (Commands that are used by the system manager for the maintenance of UNICOM are explained in the UNICOM Maintenance section of this manual.) The following commands are explained in this section:

- **cu** (Call UNIX) -- Allows you to access a directly connected system in real-time through your own system. Only one user at a time may use this command through each intersystem link.

- **mail** -- Allows you to send mail to directly and indirectly connected systems. This command functions almost identically to the **mail** command that is provided as a standard TNIX command.

- **uucp** (UNIX to UNIX Copy) -- Allows you to copy files between directly connected systems.

- **uuname** (Name Remote Systems) -- Lists the remote systems connected directly to your system, and the name of your local system.

- **uux** (UNIX to UNIX Execution) -- Allows you to cause other directly connected systems to execute commands.

CU -- CALL UNIX

SYNTAX

cu [-d] [-x] [-s speed] [-l port]

DESCRIPTION

Cu calls up another UNIX or TNIX system.

OPTIONS

-x -- specifies XON-XOF flagging on input from the remote system. Use **-x** when calling a TNIX system. If files sent via **cu** are scrambled, try reversing the setting of this flag and sending the files again.

-d -- specifies DTR flagging on input from the remote system.

-s speed -- gives the transmission speed (300, 600, 1200, 2400, 4800, or 9600 baud); The default value is 2400 baud. **cu** can communicate using only the RS 232 protocol.

-l port -- specifies the communication line device. The default value is:

-l /dev/tty3

Ask your system manager which port (/dev/tty1 -- /dev/tty7) to use for communication with a particular system.

After the **cu** command is used, the response

Connected

will indicate a connection to the remote system. You may then log in to the other system as though your terminal were being used directly with that system. You must have a user name and a password for the remote system.

NOTE

Log out from the remote system when you are finished using it.

Once you are logged in on the remote system, you may issue commands to the system.

In addition to the standard system commands, you may enter commands that begin with the "~" character that and have the following effects:

~. Terminate the connection. This does not log you out from the remote system.

~^D Terminate the connection. This does not log you out from the remote system. (^D = CTRL D)

~<file Copy the contents of **file** to the remote system, as though typed at the terminal.

~! Invoke an interactive shell on the local system.

~!cmd ...
Execute the command on the local system (via **sh -c**).

~$cmd ...
Execute the command locally and send its output to the remote system. To exit the local system shell enter logout.

~%take fromfile [tofile]
Copy file "fromfile" (on the remote system) to file "tofile" on the local system. If "tofile" is omitted, the "fromfile" name is used both places.

For some remote systems a ^D is needed to end file reception before the system prompt returns. For example:

~%take filea fileb ^D

The use of **~%take** requires the existence of the **echo** and **tee** commands on the remote system. Also, **stty tabs** mode is required on the remote system if tabs are to be copied without expansion.

~%put fromfile [tofile]
copy file "fromfile" (on the local system) to file "tofile" on remote system. If "tofile" is omitted, the "fromfile" name is used both places.

For some remote systems a ^D is needed to end file transmission before the system prompt returns. For example:

~%put filea fileb ^D

The use of **~%put** requires the **stty** and **cat** commands on the remote system. It also requires that the current erase and kill characters

on the remote system be identical to the current ones on the local system.

~~... Send the line "~..." for execution on the remote system. You may use this command to cause a directly connected remote system to execute the **cu** command. This will permit your local system to use the **cu** command to communicate with indirectly connected systems.

~>[>][:]file
0 or more lines of text entered to file from terminal
~>

This sequence places text directly into a file (or appends it, if ">>" is used) on the remote system. If ":" is used, the diversion is **silent,** i.e., it is written only to the file. If ":" is omitted, output is written both to the file and to the standard output. The trailing "~>" terminates the diversion.

DIAGNOSTICS

Exit code is zero for normal exit, nonzero (various values) otherwise.

NOTES

File copy has no error checking.

The actual baud rate of data transmission between your terminal and the remote system is slower than the baud rate of the line. This occurs because the **cu** program looks for special characters in all data sent and received.

The **cu** command may use a port on a master system that is configured for **uucp** use. The conditions for this use:

- The port must use RS-232 protocol.
- You must be logged in as user "root" to use the **cu** command on this port.
- The **uucp** command and the **cu** command will compete for use of the port; therefore do not to allow the two commands to use the port simultaneously.

When calling out from TNIX, the connection must be made prior to setting characteristics for the port, because they are lost when the communication is finished.

TNIX is not able to detect and halt transmission to a full terminal input buffer, so copying long files to a busy TNIX system is not recommended.

MAIL -- SEND OR RECEIVE MAIL AMONG USERS

The syntax permitting system names to precede the user name (**person**) for users on remote systems is the only difference between the UNICOM **mail** command and the standard TNIX **mail** command.

SYNTAX

mail [-rqp] [-f file] ...
mail [system-name! ...]person ...

DESCRIPTION

Mail with no argument prints a user's mail, message-by-message, in last-in, first-out order. For each message, **mail** reads a line from the standard input to direct disposition of the message:

newline Go on to next message.

d Delete message and go on to the next.

p Print message again.

- Go back to previous message.

s [file] ...
Save the message in the named **files** ("mbox" default).

w [file] ...
Save the message, without a header, in the named **files** ("mbox" default).

m [person] ...
Mail the message to the named **persons** (yourself is default).

^D Put unexamined mail back in the mailbox and stop.

q Same as ^D.

x Exit, without changing the mailbox file.

! command
Escape to the shell to do **command** .

? Print a command summary.

An interrupt stops the printing of the current letter.

When **persons** are named, **mail** takes the standard input up to an ^D (or a line with just "."), and adds it to each **person's** "mail" file. The message is preceded by the sender's name and a postmark. Lines that look like postmarks

are prepended with ">". A **person** is a user name.

To denote a recipient on a remote system, prefix **person** by the **system-name** and exclamation mark (refer to the **uucp** command). Everything after the first exclamation mark in **person** is interpreted by the remote system. In particular, if **person** contains additional exclamation marks, it can denote a sequence of machines through which the message is to be sent on the way to its ultimate destination.

For example, specifying **a!b!fredy** as a recipient's name causes the message to be sent to user **b!fredy** on system **a**. System **a** will interpret the user name **b!fredy** as a request to send the message to user **fredy** on system **b**.

This might be useful, for instance, if the sending system is directly connected to system **a** but not system **b**, and system **a** is directly connected to system **b**.

Each user owns his own mailbox, which is by default generally readable but not writable. The command does not delete an empty mailbox nor change its mode, so a user may make it unreadable if desired.

When a user logs in he is informed of the presence of mail.

OPTIONS

-r causes the mailbox to be printed in first-in, first-out order rather than the default.

-p prints the mailbox without questions.

-q causes **mail** to exit after interrupts without changing the mailbox, rather than just stopping printing of the current letter.

-f file causes the given file to be printed as if it was the mail file.

FILES

/usr/spool/mail/*	mailboxes
/etc/passwd	to identify sender and locate persons
mbox	saved mail
/tmp/ma*	temp file
dead.letter	unmailable text

UUCP -- UNIX TO UNIX COPY

SYNTAX

uucp [-d][-c][-m] [sysname!][~usrname/]srcpath [sysname!][~usrname/]destpath

OPTIONS

-d Creates directories in the destination system to contain the copied files. Only those directories in the total pathname to the destination files that do not exist will be created.

-c When **uucp** queues copy requests, it ordinarily makes copies of the files it will be copying to another system and puts them in the /usr/spool/uucp directory. This option allows the files themselves to be used (not a copy) for the queue's copy requests. This reduces memory use by the /usr/spool/uucp directory.

-m Sends mail to the user who starts the **uucp** command. The mail indicates whether or not the copy has succeeded.

srcpath The pathname of the file that will be copied to the destpath file. srcpath may contain the special shell characters "*","?" and "[ ]". These characters will be expanded on the system to which they apply.

destpath The pathname of the file that receives the copy of the srcpath file. destpath may contain the special shell characters "*","?" and "[ ]". These characters will be expanded on the system to which they apply. If destpath ends in a directory name, then the files copied will then have the same names as the source files and be placed in that directory.

sysname A name of a remote system directly connected to your local TNIX system. The system name must be followed by the "!" character. (To obtain a list of valid system names use, the **uuname** command. Your system manager will know which name corresponds to which physical system connected to your system.) If no system name is used, the file is assumed to reside on the system you are using, starting in your current directory.

username A user name on a system. The name must be preceded by the "~" (tilde) character. If ~username is used, then the specified path of the source or destination file starts in the home directory of username. With no username, srcpath or destpath defines the entire file path. srcpath and destpath may contain the special shell characters "*","?" and "[ ]". **~uucp** indicates the /usr/spool/uucppublic directory.

DESCRIPTION

The **uucp** command allows you to copy files between directly connected systems. Wildcard shell characters "*", "?" and "[ ]" may be used in filenames, allowing you to copy many files with one command.

Uucp File Access

The **uucp** program functions through its own user account on each of the remote systems of the communication network. The **uucp** command must access files for both reading and writing. Unlike many other commands, the **uucp** read and write access to files is only that of others. (Refer to the on-line man page for **chmod**.) This means that **uucp** can read files only if all the directories leading to the file are executable by "others" and if the file itself is readable by "others". **Uucp** can copy files to a directory only if all the directories leading to that directory are executable by "others" and if the directory itself is executable and writable by "others". The files and directories created by **uucp** are owned by **uucp** and are readable and executable by "others".

Uucp Public File

A directory for public **uucp** use is available: /usr/spool/uucppublic.

This directory can be used by "others" to receive or send files through **uucp**. For example to copy some files to another system:

1. Create a directory for yourself called username in the uucppublic directory:

   mkdir /usr/spool/uucppublic/username

2. Copy your "yourfile" to this directory:

   cp yourfile /usr/spool/uucppublic/username

3. Copy all files in this directory to the "friend" system:

   uucp -d ~uucp/username/* friend!~uucp/username/

4. The copy of these files will be found in the /usr/spool/uucppublic/username directory on the "friend" system.

How Long Does it Take to Copy Files?

The **uucp** command does not copy files at the time when it is used. Each user's communication request waits its turn in a queue in the /usr/spool/uucp directory. **uucp** will work its way through the queue, sending and receiving files with the remote systems as requested. The time required for **uucp** to work its way through the queue depends upon whether your system is a master or a slave relative to another system. (Refer to the "How UNICOM Works"

discussion in the Introduction section of this manual.) The time the command takes to finish its copy depends on many factors. Some of the major factors are:

- The number and size of the files preceding yours on the queue.
- The transmission rate of the link connecting the systems.
- Whether your system can initiate a transfer, or whether it must wait to be called by another system (master or slave system).
- The number and size of the files preceding yours on the system(s) you are calling.

Consult you system manager for the expected duration of file transmission between your system and others.

EXAMPLES

```
uucp filea sysb!/usr/username
```

This command copies **filea** in the current directory on the local system to the /usr/username directory on the directly connected remote system **sysb**. The file will be named **filea**.

```
uucp sysb!/usr/mikes/text1 sysc!/usr/fredw/text1.c
```

This command copies the file **/usr/mikes/text1** on remote system **sysb** to the file **/usr/fredw/text1.c** on the remote system **sysc**. Both **sysb** and **sys** must be directly connected to the local system.

UUNAME -- NAME REMOTE SYSTEMS CONNECTED TO LOCAL SYSTEM

SYNTAX

uuname [-l]

DESCRIPTION

Uuname lists the system names of all the directly connected remote systems that are connected to the local system. The **uucp** command (and **uux** and **mail**) may communicate with these systems.

The **-l** option returns only the name of the local system.

FILES

/usr/lib/L.sys - This file is referenced by **uuname** for the names of the remote systems.
/usr/include/whoami.h - This file is referenced by **uuname** for the name of the local system.

SEE ALSO

The descriptions of the **mail**, **uux** and **uucp** commands in this section.

UUX -- UNIX TO UNIX COMMAND EXECUTION

SYNTAX

uux [-] command-string

DESCRIPTION

Uux initiates command execution on the local system or on a directly connected system. The system executing on behalf of the **uux** command can copy files from directly connected systems if needed, and then send standard output to a file on a directly connected system.

The **uux** command uses the **uucp** command to perform its execution and thus has the same limitations on file access as the **uucp** command. Refer to the "Uucp File Access" discussion in the **uucp** command description in this section.

Note that, for security reasons, systems will limit the list of commands executable on behalf of an incoming request from **uux**. The standard commands that may be executed by **uux** on TNIX systems are **uucp** and **lp1r**. The system manager of the directly connected TNIX system may allow additional commands to be executed by **uux**. Ask the system manager for the remote UNIX systems which commands may be executed by a request from **uux**.

Uux will notify you if the requested command on the remote system was disallowed. The response comes by remote mail from the remote machine.

OPTIONS

command-string

One or more arguments that look like a TNIX (or UNIX) command line that will be executed on the target machine, with the addition that the command and file names may be prefixed by **system-name!**. A null **system-name** is interpreted as the local system. (Only the first command of a shell pipeline may have a **system-name!**. All other commands are executed on the system of the first command.) The valid **system-names** can be found with the **uuname** command. (Enter: **uuname**.) Only one system (local or remote) will execute the commands in a command line. The system that executes commands using files in other systems must recognize the **system-name(s)** which preface the files. **Uux** will attempt to copy all necessary files to the execution system.

Names of output files must be entered with surrounding parentheses, and the "\\" symbols are needed to enter the parentheses.

For example, the command

```
uux a!uucp b!/usr/file \\(c!/usr/file\\)
```

will have system "a" execute a **uucp** command to copy **/usr/file** from system "b" to system "c" File names in a command-string may be one of the following:

(1) a full path name;

(2) a path name preceded by **~xxx** where **xxx** is a user name on the specified system and is replaced by that user's login directory;

(3) anything else is prefixed by the current directory.

Any special shell characters such as "<>", ";", and "|" should be quoted either by quoting the entire command-string, or by quoting the special characters as individual arguments.

- The - option will cause the standard input to the **uux** command to be the standard input to the **command-string**.

EXAMPLES

```
uux sysb!lp1r sysc!~mikes/filea
```

This command will cause sysb to print the file filea which is on sysc in user mikes home directory.

```
uux "diff usg!/usr/dan/f1 pwba!/a4/dan/f1 > f1.diff"
```

This command will get the **f1** file from the "usg" and "pwba" machines, execute a **diff** command, and put the results in **f1.diff** in the local directory.

FILES

/usr/lib/uucp/spool - queue directory

SEE ALSO

The **uucp** command description in this section.
The article "Uucp Implementation Description" in the Bell Labs Documentation Section in this manual.

NOTES

The **uux** command uses the **uucp** command. Therefore, requests are queued and not executed immediately. The commands that may be executed by a **uux** request on a particular TNIX system are determined by the list of commands in the /usr/lib/uucp/L-xcmds file of that machine. The file access permissions for all **uux** command execution requests, (as with the **uucp** command) are that of **others**.

The use of the shell metacharacter * will probably not do what you want it to do. The shell tokens << and >> are not implemented.

Section 3

INSTALLATION

	Page
Introduction	3-1
Installation Tasks	3-2
Network Overview	3-2
System Connections and Protocol	3-3
System Names	3-4
Master and Slave Systems	3-4
Installing UNICOM	3-7
Preliminary Steps	3-7
Software Installation Introduction	3-8
Software Installation Procedure	3-8
Using the Installation Disk	3-9
Installation Disk Messages	3-10
Using the Uucp-config Program	3-12
Uucp-config Questions	3-13
Question Flowchart	3-13
Login Stream Syntax	3-19
Modifying and Correcting Errors in the Installation	3-26
DEINSTALL Program	3-28
Adding a New System to an Existing UNICOM Network	3-29
Installing the cu Command	3-30
Testing the Installation	3-31
Troubleshooting	3-32

ILLUSTRATIONS

Fig. No.		Page
3-1	UNICOM Network	3-5
3-2	Uucp-config question flowchart	3-14

TABLES

Table No.		Page
3-1	Special Characters in Login Stream Syntax	3-22

Section 3

INSTALLATION

INTRODUCTION

This section tells you how to install and modify the UNICOM software package on your 8560 TNIX system. This section assumes that you are the system manager for your 8560, and that you are familiar with the contents of both the 8560 MUSDU System System Users Manual.

Terminology. In this section the term installation refers to the installation of both the executable UNICOM code and the creation of database files containing parameters used by UNICOM.

A user must have an account on a TNIX or UNIX system to use that system. The term user name refers to the name entered in response to the UNIX or TNIX prompt **login:**, which identifies the user account to the machine. For example, the "uucp" user account refers to the account with user name "uucp". A user account may have a password. The **uucp** program has an account and a user name and may have a password. The user account is set up to start a shell or to start some other program once the account is entered via a login procedure. The user account for the **uucp** program does not start a shell, but starts the /usr/lib/uucp/uucico program.

The term login procedure refers to the process necessary to start using a TNIX or UNIX system. After a login procedure you are in the account identified by user name. The following example shows a typical login procedure:

1. Wait for the prompt "login:"

2. Enter a user name and <CR>.

3. Wait for the prompt: "password"

4. Enter the password for the user name and <CR>.

The term to log in to a system means using a login procedure to enter a user account on the system.

INSTALLATION TASKS

There are several tasks you may wish to do involving installation. Refer to the following list of tasks and follow the steps explained under each task.

- If you are installing UNICOM on a 8560 system for the first time read the entire Installation section. Follow all the steps in the "Preliminary Steps" and the "Software Installation Procedure" discussions in this section.

- If you are modifying an already existing UNICOM installation in order to correct an error in your installation or to modify some parameter of your present installation. Follow the steps in the "Modifying and Correcting Errors in the Installation" discussion in this section.

- If you are adding a new remote system to an already existing UNICOM installation you will need to modify the installation in all other systems that are directly connected to the new remote system. Follow the steps in the "Adding a New System to An Existing UNICOM Network" discussion in this section.

This section contains the following topics:

- Introduction. Gives a general overview,for both hardware and software, of what you need to create a UNICOM communication network.

- Installing UNICOM. Explains the specific procedures for installing UNICOM, and modifying or correcting a UNICOM installation.

- Installing the cu Command. Explains the procedure for installing the **cu** command on your 8560 system.

- Testing the Installation. Explains how you may verify that the UNICOM software package is working correctly.

- Troubleshooting. Explains how to fix any errors you made in your installation procedure and how to get **uucp** to work correctly.

NETWORK OVERVIEW

UNICOM allows you to set up a communication network between TNIX--UNIX systems, and TNIX--TNIX systems. Each system may be connected to one or more systems, and these systems in turn may be connected to one or more systems. A network of any form (e.g. star) may be made with UNICOM, but only the mail command may send messages to both directly and indirectly connected systems. It is possible to utilize the facilities of indirectly connected systems with the use of the **uux** and cu commands when used in combination with other commands.

System Connections and Protocol

Connections between systems may be hard-wired or be done through the telephone lines using an autodialer-modem on one end and a modem with autoanswer on the other. A system may use both methods for interconnection to other systems.

For the **uucp**, **uux**, and the **mail** commands:

- 8560 communications support HSI and RS-232-C protocol. HSI protocol can be used only under restricted conditions. Refer to the "Choosing HSI or RS-232-C Protocol for 8560 Communications" discussion following for these restrictions.

- 8560--8560 communications via autodialer-modem support only RS-232-C communications.

- 8560--UNIX host communications support RS-232-C protocol.

For the **cu** command:

- The communication protocols are the same as for the other UNICOM user commands, except that HSI protocol is not supported.

The UNICOM package supports the Hayes Smartmodem autodialer (300 baud version), and the Racal-Vadic 3451 with auto-dial option. You may modify the C language program uuphone.c (included with the UNICOM package) to support other autodialers.

The effective baud rate for 8560 HSI communications is 7000 baud. The RS-232-C protocol will support rates of 300, 600, 1200, 2400, 4800 and 9600 baud. The 9600 RS-232-C baud rate protocol has an effective communication rate of about 2000 baud.

TNIX is a time-shared system, and any CPU system load in addition to **uucp** will slow down the rate of transmission of information.

Choosing HSI or RS-232-C Protocol for 8560 Communications

You may use either RS-232-C or HSI protocol for communications between 8560 systems. HSI protocol has the advantage of an effective communication rate that is about three times the maximum effective rate for the RS-232-C (with 9600 baud) protocol. However, HSI protocol is subject to several restrictions:

- The slave system of the two systems using a HSI protocol link must be left on and connected at all times. If it is not, an attempt to use UNICOM communications from the master side will cause the master system to crash at a later, unpredictable time. You must reboot the master system if it accidentally attempts to call a slave system that is turned off, even if the master system has not yet crashed.

- You cannot use the **cu** command over HSI protocol links.
- HSI communication is subject to communication errors. If an error occurs during a file transfer, the file will not be transferred, and you must request the transfer again.

System Names

You assign each system a name when you install UNICOM on that system. All those commands that use intersystem communication use that system name when addressing that system. Each system knows the system names of all the systems which are directly connected to it and whether that system is a master or a slave. Refer to the "Using the Installation Disk" discussion in this section for the syntax of a system name.

Master and Slave Systems

When a port on each of two systems are connected, these two ports form a master-slave communication pair of ports or systems. A connected port on a system is either a **master** or a **slave**. **Uucp** communication is done between a **master** and a **slave** port on different systems.

A **master** system sends a call to a **slave** system to start communications. A **master** system must have a port dedicated for its exclusive use. This port will have a hard-wired connection to a **slave** system or to the phone lines via an autodialer. This port will not permit a login procedure from a terminal.

A **slave** system must wait for a master system to call it before the two systems can communicate. A **slave** system does not require a dedicated port (unless the HSI protocol is used.) This port may be reconnected to a terminal and the port will function normally. A **slave** system will require a modem if it communicates via the phone lines.

It is possible for two systems to both be masters and slaves to each other. This can be done by dedicating a port to be a master on both systems and another port on both systems to be used as a slave. In this case both systems can initiate communication. Refer to Figure 3-1 for a diagram of a possible UNICOM communications network.

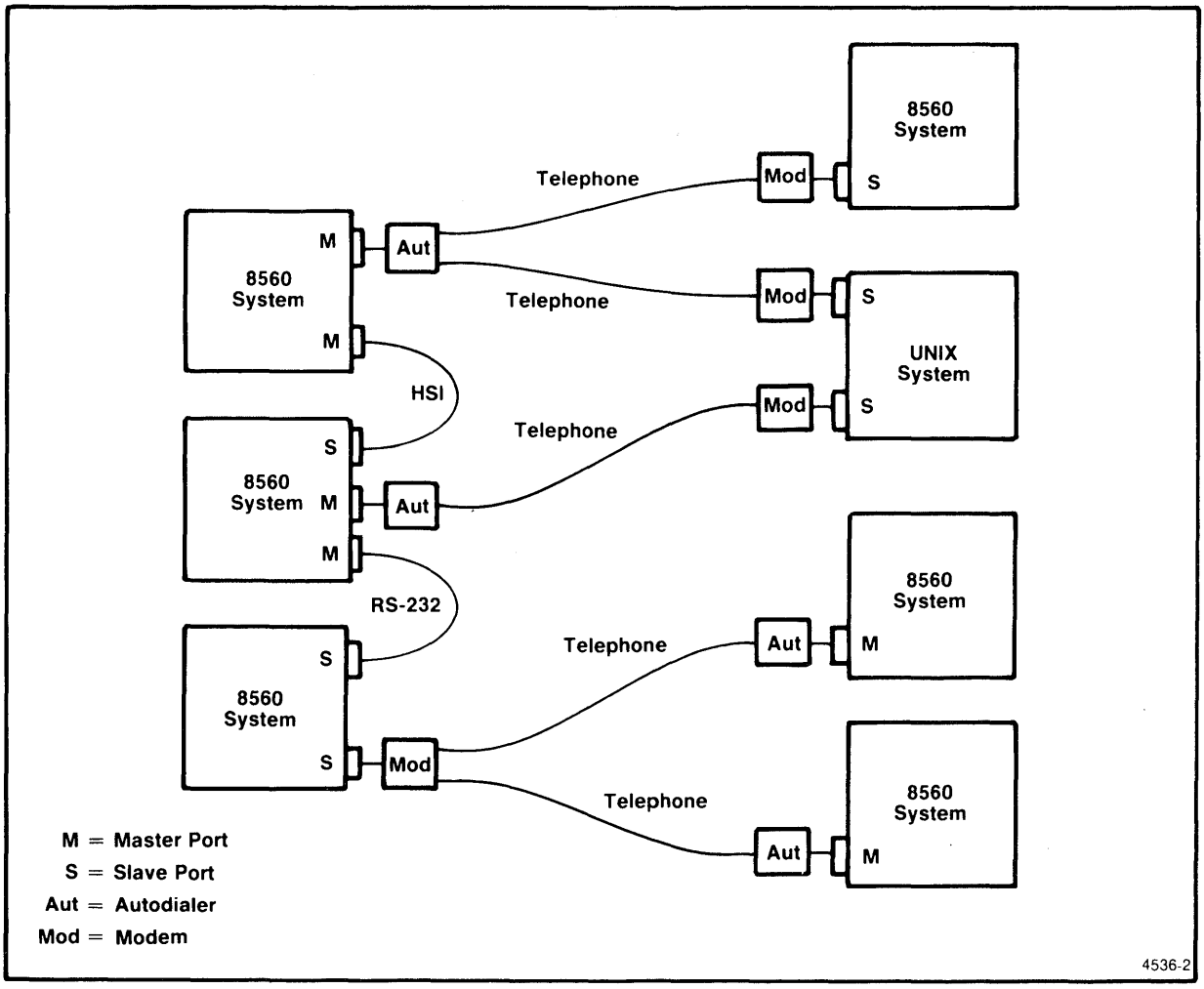


Fig. 3-1. UNICOM Network.

This figure shows a typical UNICOM communication network. Each system has some port(s) used for master(M)-slave(S) pair use. Each port is connected either via hard-wire or telephone lines (with an autodialer-modem combination) to a port on another system. The connection protocol is either HSI or RS-232-C for hard-wired connections or RS-232-C for telephone line connections. Cu communications can not be used with HSI protocol.

Message Display Syntax

The messages displayed on the terminal during UNICOM installation are shown in this section using the following syntax:

- The explanation of a message will be in plain text.
- Messages as shown on the terminal are indented. For example:

  That file does not currently exist. It will be created.

- Portions of messages inside "< >" brackets are echos of parameters you have entered. For example:

  The name '<sysname>' will be entered in the file '/usr/include/whoami.h'

- A message that ends in ??? requires a response from you. For example:

  Is this OK? ???

INSTALLING UNICOM

The following discussion will guide you through the steps needed to install the hardware and software for your UNICOM communications package on your TEKTRONIX 8560 MUSDU.

(If you are not installing UNICOM for the first time on your system, refer to either the "Modifying and Correcting Errors in the Installation" or the "Adding a New System to an Existing UNICOM Network" discussions in this section.)

If you are installing UNICOM for the first time, continue reading this section.

PRELIMINARY STEPS

Before beginning your software installation:

1. Decide which systems (TNIX and UNIX) will be master and which will be slave. Drawing a diagram of your network may be helpful. Remember that a slave must wait for a master to call it before it can communicate.

2. Decide which ports will be used by UNICOM on your 8560s. Master systems and the **cu** command require at least one dedicated port. HSI protocol also requires a dedicated port on the slave system. The **cu command** can communicate only via RS-232-C protocol.

3. Decide upon the system name for each of your systems.

4. Have the system manager of any directly connected system update the **uucp** database files of that system to recognize the system name (and login procedure if yours is a master system) of the 8560 system on which you are installing UNICOM.

5. Any UNIX system to be connected for UNICOM communication may have a password for the "uucp" user name. Obtain this password, if any for each system. Obtain the login procedure for the UNIX machines. Obtain a list of commands the **uux** command may execute on each UNIX system.

6. Obtain an autodialer-modem for each master port that will be using the telephone lines.

7. Obtain a modem with autoanswer for each slave port that will be using the telephone lines.

8. Prepare cables for the RS-232-C, modem, or HSI hardwire connections. (Refer to "Cable Connections" in the Technical Notes section of this manual.) Connect the systems using the ports you choose in step two.

9. Change the jumper inside the 8560 to HSI on any port (on both master and slave) that you wish to use with HSI protocol. Refer to the 8560 MUSDU Installation Guide for instructions on how to reconfigure a jumper.

SOFTWARE INSTALLATION INTRODUCTION

After you have completed the "Preliminary Steps", you may begin the installation of the UNICOM software. UNICOM must be installed on each 8560 system in the network.

To install UNICOM, you will use an interactive program to enter parameters describing the specific details of your UNICOM network. These parameters are entered into database files used by UNICOM. These parameters are necessary for installing the **uucp** and the **cu** user commands.

The other UNICOM user and maintenance commands are installed automatically as the **uucp** command is installed. Therefore, references in this section to the installation of the **uucp** command also refer to the installation of the whole UNICOM set of commands.

If You Make an Error During Installation

To correct an error in parameter entry (if it is too late to backspace over the error) refer to the "Modifying and Correcting Errors on the Installation" discussion in this section.

SOFTWARE INSTALLATION PROCEDURE

The software installation is performed in four steps:

1. Use the UNICOM installation disk to copy all the necessary software onto your system. Refer to the "Using the Installation Disk" discussion in this section.

2. Use the interactive **uucp-config** program (included in the installation disk) to create the proper environment for **uucp** on your 8560. Refer to the "Using the Uucp-config Program" discussion in this section.

3. Install the **cu** command on master systems that use RS-232-C protocol. Refer to the "Installing the Cu Command" discussion in this section.

4. Test your installation. Refer to the "Testing Your Installation" discussion in this section.

USING THE INSTALLATION DISK

This discussion explains how to use the installation disk. The software installation disk requires that you answer two questions. The following paragraphs explain the steps you take to use the installation disk, the two questions you must answer, and the messages from the installation disk.

This discussion does not explain or show all the messages that the installation disk sends to your terminal. Only those messages that concern you will be explained.

Follow the numbered steps below. Read the "Installation Disk Messages" discussion. This discussion explains the parameters that are automatically created by the installation disk. You may wish to change some of these parameters at a later time.

1. Log in on your TNIX system as user "root". Be sure that you are the only user on the system.

2. Place the installation disk in the disk drive and type install .

3. The following message is the first of two that require a reply:

   Enter your local system name: ???

   Enter a name consisting of eight or fewer alphanumeric characters. You must use either all uppercase or all lowercase letters. The name may begin with any numeral or letter. A "-" (minus sign) may be used as a character in the name (except as the first character). Some UNIX systems do not recognize uppercase system names. For example, a system name could be: hi-ho40z

   You will get a response:

   The name '<sysname>' will be entered in the file '/usr/include/whoami.h'.

   Is it alright to create /usr/include/whoami.h (y or n) ? ???

   Answer "y".

   (If you already have a file /usr/include/whoami.h the message will be:

   Is it alright to replace /usr/include/whoami.h (y or n) ? ???

   If you want to save your copy of this file, answer "n". The

installation will continue but will not create the file /usr/include/whoami.h. The whoami.h file is necessary for UNICOM. In order to have the installation disk create the file, allow the installation to finish and then rename your whoami.h file. Use the **DEINSTALL** (enter: /usr/lib/uucp/DEINSTALL) command to remove the installed software (refer to the "DEINSTALL Program" discussion in this section) and use the installation disk again. This time answer "y" to the "Is it alright ..." question and allow the installation to finish again.

Installation Disk Messages

The following installation disk messages indicate that the installation disk is automatically placing parameters in certain database files. You may wish to change these parameters at a later time. In order to change these parameters entered by the installation disk you must log in as user "root" and edit the affected database files directly with a text editor.

/etc/passwd Parameters Message.

```
Creating users "uucp" and "uucpa"
Creating user uucp with this password line:

uucp::9:4:unix to unix copy:/usr/spool/uucppublic:
                                        /usr/lib/uucp/uucico
Creating user uucpa with this password line:

uucpa::9:4:uucp administrator:/usr/lib/uucp:
```

The installation creates two new user accounts with user names "uucp" and "uucpa". This is done by placing the displayed password lines in the /etc/passwd file. Both these user accounts are created without passwords. (Refer to the on-line man page passwd for information on the syntax of the lines in the /etc/passwd file.)

The local system user account for the **uucp** program has the user name "uucp" This user account is used by the local and remote systems **uucp** program for data transmission. The remote **uucp** program communicates by logging in to the user account "uucp" on the local system. The "uucp" user account starts the "uucico" program, not a command interpreting shell. You may give the "uucp" account a password with the **passwd** command (enter: passwd uucp while logged in as user "root"). Each system that will call the local system via uucp must know this password.

"uucpa" is the user name for the account used by the **uucp** administrator. This account starts a standard shell. The **uucp** administrator maintains the UNICOM software. Both "uucp" and "uucpa" user accounts have the same group and user numbers. Thus, the "uucpa" account has read and write access to all **uucp** files. You may give the "uucpa" user account a password by using the **passwd** command while you are loged in to the "uucpa" user account.

/usr/lib/crontab Parameter Message.

```
Adding uuclean command line to CRONTAB
/usr/lib/crontab: 5   1   *   *   *   /usr/lib/uucp/uuclean -p -m
```

The UNICOM installation disk installs the **cron** program if your system does not already have it. The installation disk also places a line in the /etc/rc file that starts **cron** when the system is booted. The cron program uses the file /usr/lib/crontab as a data base. The format of the /lib/crontab file may be found in the on-line man page **cron** (enter: man cron). **Cron** will execute commands periodically as determined by the lines in the crontab file. This command line displayed is added to crontab and performs the **uuclean** maintenance at 1:05 am every morning. You may wish to modify either the crontab syntax portion of this line or the **uuclean** portion. (The **uuclean** command with the options -p and -m removes all files from the /usr/spool/uucp directory that are older than 72 hours.) To change the parameters in this line, log in as user "root" and modify the line with a text editor.

Refer to the **uuclean** command discussion in the UNICOM Maintenance section of this manual.

Replacing the Mail Command Message.

```
Copying program "rmail" to /usr/bin
and replacing old /bin/mail
```

The UNICOM installation replaces your present version of the **mail** command with a new version that allows mail to be sent to other systems. The on-line man page mail(1) is also replaced. The old version of mail is not erased but is placed in the file /usr/lib/uucp/DISTBIN/OLD-mail. For information on how to restore the old version of mail, refer to the "DEINSTALL PROGRAM" discussion in this section.

Now that you have completed the installation of the software you must create the proper entries in the **uucp** database files, as described next.

USING THE UUCP-CONFIG PROGRAM

Introduction

The **uucp** command requires database files to function, and the help of several auxiliary programs that also use database files. The following paragraphs tell you how to use the uucp-config program to create entries in these files.

The **uucp** database files require maintenance after installation. Refer to the UNICOM Maintenance section in this manual for further information.

Uucp-config Procedure

uucp-config is an interactive program requiring responses to a number of queries. Follow these steps to use **uucp-config**.

1. Log in as user "root", and make the current directory /usr/lib/uucp. Be sure that you are the only user on the system.

2. Enter: **uucp-config**

3. Reply to the questions the **uucp-config** program asks you. Refer to "Uucp-Config Questions" discussion in this section.

4. While logged in as user "root" add the following line to the /usr/lib/crontab file:

   ```
   0 * * * * /usr/lib/uucp/uulog
   ```

   This allows the **uulog** program to perform a maintenance function hourly.

5. Enter: shutdown

6. Reboot the 8560 system.

UUCP-CONFIG QUESTIONS

The following paragraphs explain all the questions **uucp-config** will ask you that require responses.

The uucp-config is a loop that asks the same questions each time the loop is cycled. The answers to your questions will determine the set of questions in the cycle. Each cycle through the loop will add one additional system to those you may communicate with using **uucp**. The loop is completed when it returns to the first question:

Name of neighbor system:

When you are finished adding systems to your installation and have returned to the beginning of the question cycle, exit the **uucp-config** program by entering ^C (CTRL-C) in response to this question.

Question Flowchart

Use the flowchart shown in Figure 3-2 as an aid in determining which questions you will be answering to the **uucp-config** program. Each question, along with its explanation, is designated with a letter in the following "Question and Explanation" discussion. The lettered boxes in the flowchart tell you what information you need to supply for the corresponding question in the **uucp-config** program.

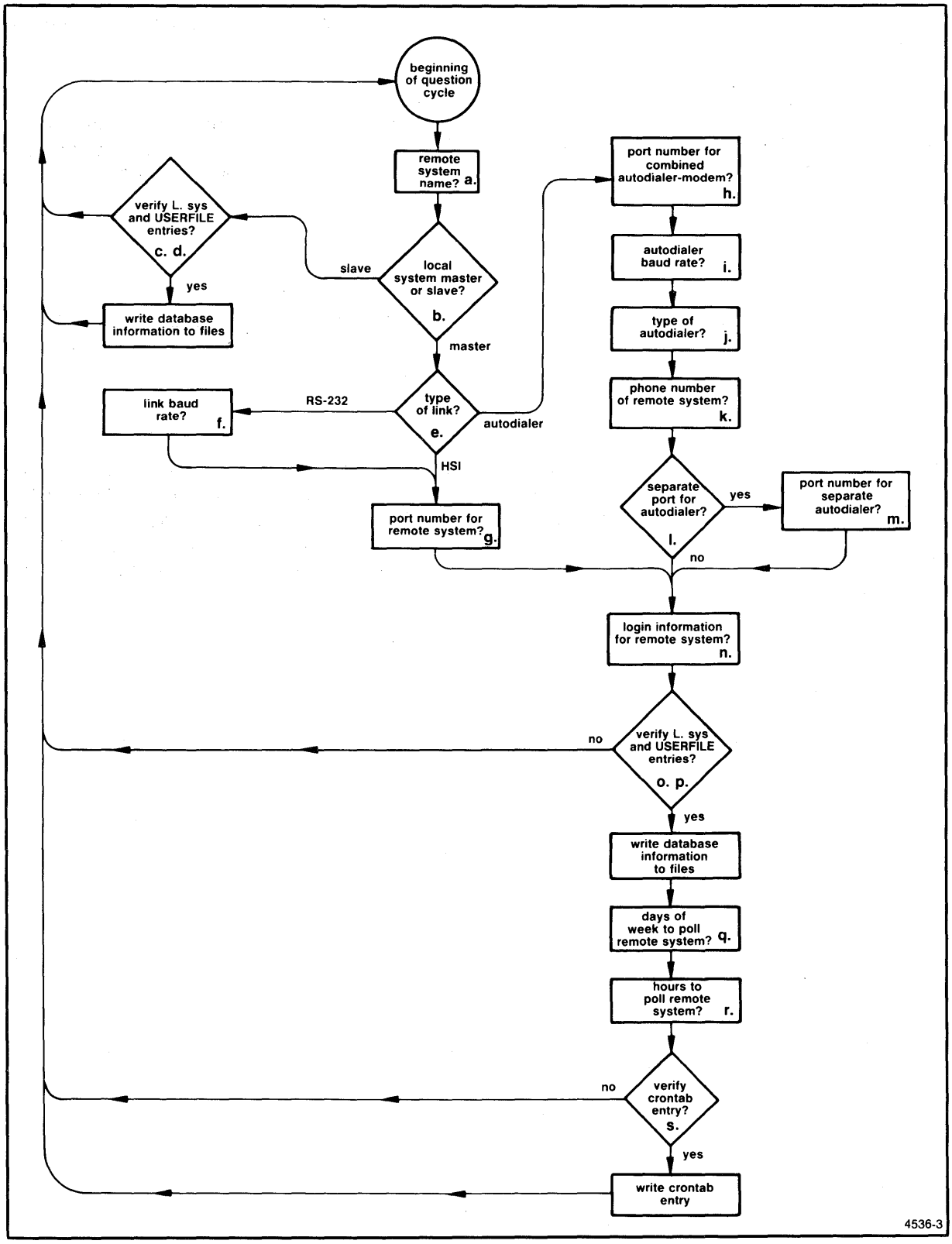


Fig. 3-2. Uucp-config question flowchart.

Questions and Explanation

a.

Name of neighbor system: ???

Enter the system name of the directly connected "neighbor" system. If the system name on the neighbor system is already defined, you may discover it by logging in to the neighbor system and typing: uuname -l. If the neighbor system name is not yet defined, enter a name consisting of eight or fewer alphanumeric characters. You must use either all uppercase or all lowercase letters. The name may begin with any numeral or letter. A "-" (minus sign) may be used as a character in the name (except as the first character). (Some UNIX systems do not recognize uppercase system names.) When you install UNICOM on the neighbor system, you must give it this name.

When the **uucp-config** question loop is completed, you will be able to communicate with this system.

b.

Is local site to be Master or Slave?
Type 'M' for master, 'S' for Slave: ???

This defines whether your system is to be a master or a slave relative to the directly connected system (neighbor system) you have just named.

c.

Local Site is SLAVE
Updating USERFILE with
uucp,<sysname> /

Is this OK? ???

This indicates that the line "uucp,<sysname> /" is placed in the file /usr/lib/uucp/USERFILE. Answer "y". **sysname** is the name of the neighbor system you entered earlier. If you answer "n", **uucp-config** will return to the beginning of the question cycle. For information on the syntax of the lines in this file, refer to the "Database Files" discussion in the UNICOM Maintenance section of this manual.

d.

L.sys line is: <sysname> Any Slave 2400 notty

Is this OK? ???

This indicates that the above line is placed in the file

/usr/lib/uucp/L.sys. Answer "y". **sysname** is the name of the neighbor system you have entered earlier. If you answer "n" **uucp-config** will return to the beginning of the question cycle. For information on the syntax of the lines in this file, refer to the "Database Files" discussion in the UNICOM Maintenance section of this manual.

e.

```
Is link to remote site Direct TTY, Direct HSI, or Dialup?
Type 'T' for tty 'H' for HSI, or 'D' for dialup: ???
```

Answer "T" if the link between systems will be hard-wired and will use RS-232-C protocol. Answer "H" if it will be hard-wired and use HSI protocol. Answer "D" if the link will be through an autodialer-modem via phone lines.

f.

```
Enter default baud rate of  tty: ???
```

Enter the baud rate for the port. You may enter 300, 600, 1200, 2400, 4800, and 9600. The slave port on the directly connected machine must be set to the same baud rate. The default rate for the slave port is 2400. You may change the baud rate for the slave system by editing the /etc/ttys file in the slave system. Refer to the on-line man page **getty** for information on the syntax of this file (type: man getty).

g.

```
Enter tty port for direct connection
Or corresponding tty port to desired HSI port
in the form ttyX, where X is the tty number: ???

Disabling port ttyX
```

Enter the port number to be connected to the neighbor system. Enter in the form "ttyX", where X is between 1 and 7. The response indicates that port ttyX has been dedicated to **uucp** use.

h.

```
Enter tty port for connection in the form ttyX,
where X is the tty number: ???
```

Enter the port number to be connected to the neighbor system. Enter in the form "ttyX", where X is between 1 and 7.

i.

```
Enter primary baud rate of dialer: ???
```

Enter the autodialer's baud rate.

j.

8560 UUCP currently supports the HAYES and VADIC Autodialers
Enter "HAYES" for the HAYES 300, and "VADIC" for the VADIC
3451 Modem with Autodial Option

Enter type of modem/dialer (see UNICOM User's Manual): ???

UNICOM supports the HAYES and VADIC autodialers. Enter "HAYES" or "VADIC". If you have another autodialer refer to the **uuphone** command description in the UNICOM Maintenance section of this manual.

k.

Enter phone number of '<sysname>': ???

Enter the telephone number of the system named "sysname". Both the HAYES and VADIC can use the syntax:

xxx-xxxx.

The "x" is a digit in the telephone number. If you modify **uuphone** to accept another autodialer, follow the telephone number syntax for that autodialer.

l.

Is dialer on same port as modem (y or n)? ???

The HAYES and VADIC autodialer-modems do not require a separate port. If your autodialer-modem combination are separate, each will require its own port. If the two are not separate, answer "y". If they are separate answer "n".

Disabling port ttyX

This response indicates that port X is dedicated to the autodialer. X is between 1 and 7.

m.

Enter tty port for dialer (ttyX): ???.

Enter the port number X to be connected to the autodialer portion of your modem that has a separate autodialer and modem. Enter in the form "ttyX", where X is between 1 and 7. This port will be disabled.

Disabling port ttyX
Disabling port ttyY

This response indicates that port X is dedicated to the autodialer portion and that port Y is dedicated to the modem portion of your autodialer-modem. Both X and Y are between 1 and 7.

n.

Enter the login stream for the target machine
It has the form:
EXPECT-SEND-EXPECT-SEND... EXPECT SEND ...
For 8560s and most UNIX machines on direct lines
the line:

'-EOT-ogin:-EOT-ogin: uucp assword: XXXX'

will work, with the machine's password put in place
of the XXXX, or 'assword: XXXX' omitted completely
if the machine has no password.

Use 'EOT' to indicate a CTRL-D, use '\r' for a carriage return,
and use '\d' for a 1 second delay. If you use these characters,
be sure to escape the backslashes when entering them (e.g. '\\d').

If you type a Carriage Return, the default login stream will be used.

Consult the UNICOM Users Manual for more info.

Login stream: ???

When **uucp** attempts to transmit information to a machine it first logs in to the other machine. The login stream is the login procedure for the neighbor system. The login procedure is written in a special syntax that is translated by **uucp**.

The default login stream, suitable for a TNIX system with HSI protocol and without a "uucp" account password will be used if you type <CR>. This is:

-EOT-ogin:-EOT-ogin: uucp

If the TNIX system has a password, enter for the login stream:

-EOT-ogin:-EOT-ogin: uucp assword: XXXX

where XXXX represents the password.

If you cannot use these standard login streams to log in to the neighbor system, read the following discussion, "Login Stream Syntax", and write a login stream for the neighbor system and enter it.

After entering the login stream, continue answering the uucp-config questions.

Login Stream Syntax

In order to write the login stream for the remote system you must first write down the login procedure for the remote system. You will then convert this procedure into a login stream using the rules for the login stream syntax.

You will probably need to do some experimentation to get your login stream to work properly.

Login Procedure. The login procedure will consist of all the characters that you expect to send in order to log in on the remote machine. You will be sending some of the characters only after you receive certain prompts. Delays in responding to received characters may be needed and repetition of sent characters may be necessary until certain prompts are received from the remote system. Here is an example of a simple login procedure typical of many UNIX and TNIX systems:

1. (receive) "login:"
2. (send) "uucp<CR><LF>"
3. (receive) "Password:"
4. (send) "uucppass<CR><LF>"

After you have written down the login procedure for the remote machine you will translate the procedure into the login stream syntax and then enter the syntax as a response the the "Login stream:" **uucp-config** prompt:

Login Stream Syntax.

SendReceiveAttempt1 sendchar1 [SendReceiveAttemptN sendcharN] ...

Syntax Explanation

Your login stream will be one or more pairs of the fields SendReceiveAttempt and sendchar. These fields are separated by spaces. The SendReceiveAttempt field itself has a syntax that results in a send--receive character interchange with the remote system. If the expected characters occur in this interchange, the characters in the sendchar field are sent to the remote system. If the expected characters do not occur, the login has failed. The SendReceiveAttempt syntax may be as simple as a single set of characters that are expected from the remote system; "login:" for example.

We repeat the login procedure shown earlier and show the login stream that is the translation of the login procedure for the remote system:

Login procedure is:

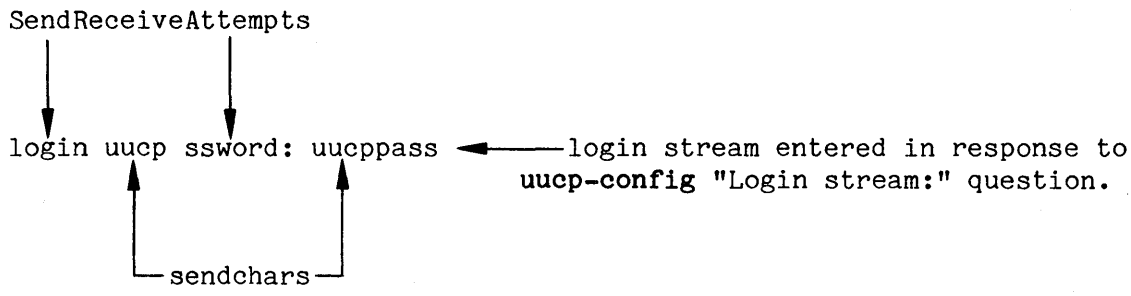
1. (receive) "login:"

2. (send) "uucp<CR><LF>"

3. (receive) "Password:"

4. (send) "uucppass<CR><LF>"

The login stream for this simple case is:



Note that "ssword", a subset of "Password" is used.

Login Stream Syntax Options

sendchar contains those characters sent to the remote system after a successful **ReceieveSendAttempt**. The transmission of the **sendchar** characters are automatically followed by the transmission of the <CR><LF> (carriage return line feed) characters.

SendReceiveAttempt
will be successful, or not successful. If it is not successful, then the login to the remote system has failed. If SendReceiveAttempt is successful then the characters in the next sendchar field are sent.

SendReceiveAttempt Discussion:

The SendReceiveAttempt field consists of one or more -send-receive pairs. The SendReceiveAttempt field transmits the send portion of the first pair to the remote system and expects the receive portion. If it is received, the SendReceiveAttempt is successful. If it is not received (either no response or a non-matching response), then the send portion of the second pair is sent and the receive portion of the second pair is expected. This process continues through all the -send-receive pairs until either a receive is recognized, or until the pairs are exhausted. If no receive is recognized, the SendReceiveAttempt is unsuccessful.

The field may begin with a receive rather than a send, in which case nothing is initially sent. The receive is first expected and the field then functions as before.

SendReceiveAttempt Syntax:

Case (1) receive1

or

Case (2) [receive1]-senda-receivea[-sendN-receiveN] ...

Case (1)

receive1 contains the characters (or some subset of the characters) the **SendReceiveAttempt** expects to receive from the remote system. (For example, "ogin:" which is a subset of the word "Login:".) If **receive1** is received, then the SendReceiveAttempt field is successful and the next sendchar is sent. If **receive1** is not received, then the SendReceiveAttempt field is not successful and the login has failed.

Case (2)

receive1 contains the characters expected by the local system. If they are received, then the SendReceiveAttempt field is successful and the next sendchar is sent. If they are not received, then the next -send-receive pair is used. If receive1 is not present, then senda is sent, and receivea is expected, and the process continues as before.

-senda-receivea
is one of one or more -send-receive pairs that are used if the local system does not receive the characters contained in the previous receive field. The characters contained in the send portion of the pair are sent and the characters in the receive portion are expected. (The "-" (minus sign) character must be used as a separator; no blanks may be used.) If these are not received, then the next -sendN-receiveN pair is used in a similar manner. If none of the pairs elicits a receive that is recognized, then the SendReceiveAttempt field is unsuccessful and the login attempt fails. (Any **send** characters sent are automatically followed by the <CR><LF> (carriage return, line feed) characters.

Special Characters

All **send** and sendchar fields may contain special characters. Some special characters are preceded by the "\" character. Refer to Table 3-1 for a list of the special characters and their function.

NOTE

When entering the special characters \d, \r, \b, or \c in the login stream with **uucp-config**, enter an additional "\" before each special character. For example, to enter the "\d" (send a <CR>) enter "\\d". This is not necessary when editing the /usr/lib/uucp/L.sys file directly.

Table 3-1
Special Characters in Login Stream Syntax

Special Character	Effect
\d	Delay 1 second
\r	Send <CR>
\b or -BREAK	Send BREAK
\c at -EOL	Don't send a <CR><LF> after sending this send
-EOT	Send ^D

For example, if **-send** is "-\d", a delay of 1 second happens before other characters are processed.

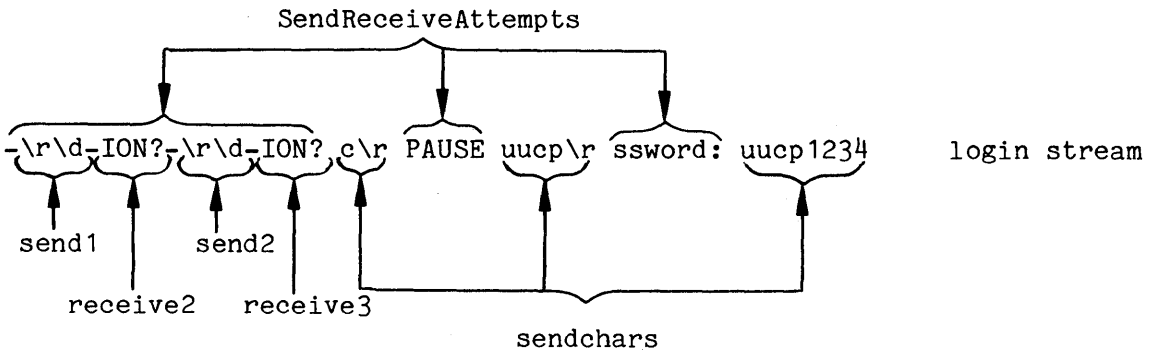
After you have entered the login stream, continue answering the **uucp-config** questions.

Example

A login procedure:

1. (send) <CR>
2. (delay 1 sec)
3. (receive) "DESTINATION?"
4. (repeat steps 1, 2 and 3 once if 3 fails)
5. (send) "c<CR>"
6. (receive) "PAUSE:<LF>login:"
7. (send) "uucp<CR>"
8. (receive) "Password:"
9. (send) "uucp1234<CR>"

The login stream for this example is:



Note that "ION?" is a subset of "DESTINATION?".

This syntax may also be found in the "UUCP Implementation Description" article in the Bell Labs Documentation section of this manual under the topic "Uucp Installation".

o.

L.sys line is: <sysname> Any ttyX 2400 HSI -EOT-ogin:-EOT-ogin: uucp

Is this OK? ???

The line placed in the /usr/lib/uucp/L.sys file describing the system is displayed for your approval. The line displayed here is typical, but may not be exactly the line that is displayed for your installation. For information on the syntax of the lines in this file, refer to the "Database Files" discussion in the UNICOM Maintenance section of this manual. Enter "y" or "n". If you enter "n" the entire cycle of questions will be repeated for this system.

p.

Updating USERFILE with
uucp,<userfile>

Is this OK? ???

The line placed in the /usr/lib/uucp/USERFILE file describing the system is displayed for your approval. For information on the syntax of the lines in this file, refer to the "Database Files" discussion in the UNICOM Maintenance section of this manual. Enter "y" or "n". If you enter "n" the entire cycle of questions will be repeated for this system.

q.

How often do you wish to poll '<sysname>'\?
Enter which days of the week, in the form "1,3,5" where 1=monday,
2=tuesday, etc. Be sure to place commas between entries
If you wish to poll all days, just type an '*' (asterisk)
Days of Week? ???

Uucp on your system polls neighbor system "sysname" for communication at regular intervals. This is done by the **cron** program that uses the database file /usr/lib/crontab. This question and the next two, determine when and how often your master system will poll neighbor system "sysname". Enter the days of the week that you wish to poll "sysname". Use the syntax described the in above message.

r.

Enter which hours you wish polls to be performed,
in the form "0,2,4,6,8,10,12,14,18,20,22" where 0=midnite, 23=11pm.
If you wish to poll every hour, just type a '*' (asterisk)

Hours? ???

Enter the hours that you wish to poll "sysname" using the syntax in the above message.

s.

This line will be entered in your "/usr/lib/crontab" file:
0 * * * * /usr/lib/uucp/uucico -r1 -ssysname

Is this OK? ???

The line displayed is the line to be entered in the crontab file. Answer "y". Refer to the on-line man page **cron** (8) for an explanation of the syntax of the crontab line. Refer to the **uucico** command in the Bell Labs Documentation Section of this manual for information on the syntax of the **uucico** command. If you answer "n" you must repeat the entire question cycle for this system.

This is the last question in the question cycle.

Files Modified

uucp-config modifies these files:

- /usr/lib/USERFILE
- /usr/lib/L-devices
- /usr/lib/L.sys
- /etc/ttys
- /usr/lib/crontab

Refer to the "Database Files" discussion in the UNICOM Maintenance section of this manual for more information on these files. The format and function of the first three files may be found in the "Uucp Implementation" article in the "Bell Labs Documentation" section of this manual. The format of the /etc/ttys file may be found in the on-line man pages ttys(5) and getty(8). The format of the /usr/lib/crontab file may be found in the on-line man page cron(8).

MODIFYING AND CORRECTING ERRORS IN THE INSTALLATION

You may wish to modify the parameters you have entered either to correct an error in your parameter entry or to change the details of your UNICOM network. There are three methods for modifying the parameters in the database files used by **uucp**.

1. Remove all the UNICOM software and re-install it, re-entering all the parameters, modifying some as necessary.

   a. Note down all the parameters in your present UNICOM installation.

   b. Log in as user "root" and type: /usr/lib/uucp/DEINSTALL. Answer "y" to the question:

      Are you sure you want to do this?

      (Refer to the "DEINSTALL Program" discussion in this section.)

   c. Check the contents of the /etc/ttys file. Use a text editor to re-enable all the ports that were disabled by your previous UNICOM installation. Refer to the "NOTE" in the DEINSTALL PROGRAM discussion in this section for the procedure.

   d. Do the entire UNICOM installation again, entering all the parameters (those changed and those unchanged). Refer to the "Software Installation Procedure" discussion in this section.

2. Use the **uucp-config** program to enter correct data into the database files and remove the old lines in the files with a text editor.

   a. Log in as "root" and type: /usr/lib/uucp/uucp-config .

   b. Answer the **uucp-config** questions, entering the modified parameters for each system that requires modification. Refer to the "Uucp-config Questions" discussion in this section. Exit the **uucp-config** program.

   c. Delete the old lines from the database files with a text editor.

      (The **uucp-config** program adds new lines to the database files, but does not remove the old lines that contain the unmodified parameters.) The following files are modified by uucp-config:

      i. /usr/lib/uucp/L.sys

ii. /usr/lib/uucp/USERFILE

iii. /usr/lib/uucp/L-devices

iv. /usr/lib/crontab

Delete those lines in these files that still have the old parameters that you have just changed with uucp-config. Refer to the "Database Files" in the UNICOM Maintenance section of this manual for further information.

3. Use a text editor to directly edit the database files.

a. Log in as user "root" and edit those database files that contain the parameters that need to be modified. Refer to the "Database Files" in the UNICOM Maintenance section of this manual for further information.

DEINSTALL PROGRAM

The UNICOM installation disk, in combination with the **uucp-config** program, creates and modifies a number of files. These files must have the proper data and syntax in order for the UNICOM commands to work. If an error is entered into any of these files, it may be easiest to fix the error by removing all the UNICOM files and modifications, and doing the installation over again. The **DEINSTALL** program removes all the files and entries to existing files of the UNICOM installation and the **uucp-config** program. **DEINSTALL** also restores the standard TNIX **mail** command when it removes the UNICOM **mail** command. **DEINSTALL** is located in the /usr/lib/uucp directory.

To DEINSTALL the UNICOM software, log in as the user "root" and enter: **/usr/lib/uucp/DEINSTALL** . **DEINSTALL** will ask:

> Are you sure you really want to do this?

Answer "y".

NOTE

> **DEINSTALL** does not remove the changes that the **uucp-config** program makes to the /etc/ttys file. If your local system was installed as a master system on a port, the port was disabled by the **uucp-config** program and will not accept a log in from a terminal.
>
> You may restore the port by editing the /etc/ttys file (owned by user "root"). The file will have a line: 0ZttyX , where X is an integer in the range 0-7 and Z is an integer or a letter. Change the first "0" to a "1" in that line and the port will be restored to normal access. After doing this, you must reboot the system.

ADDING A NEW SYSTEM TO AN EXISTING UNICOM NETWORK

Each time you add a new system to your existing UNICOM network, all the systems directly connected to the new system must recognize the new system.

Use the **uucp-config** program to add the new system to each directly connected system. Answer a complete cycle of questions for each new system added to the network. Refer to the "Using the Uucp-config Program" discussion earlier in this section.

INSTALLING THE CU COMMAND

The **cu** command requires the use of a dedicated port. A port is dedicated for **cu** use by changing the entry for the port in the /etc/ttys file. This dedicated port is the master for the **cu** command and will not accept an external login. The **cu** command cannot communicate over lines using the HSI protocol.

CU INSTALLATION PROCEDURE

1. Log in as user "root". Be sure you are the only user on the system.

2. Using a text editor, modify the line for port X in the /etc/ttys file that says

   1ZttyX

   Where X is between 1 and 7 and Z is a letter or a numeral. Change the first "1" in the line to a "0" so that it reads:

   0ZttyX

   The Z in the line determines the baud rate for the line when using the **cu** command. The baud rate on both the master and slave ports for the **cu** command must be the same. When invoking the **cu** command, use the correct baud rate with the -s option.

3. Enter: shutdown

4. Reboot the system.

TESTING THE INSTALLATION

You will want to test your UNICOM installation. Test your UNICOM installation from the master side of the master-slave pair of systems. Each link between two systems must be tested separately. If two systems are connected via links using RS-232-C protocol, you may test the hardware portion of the link separately from the software using the line. This is done with the **cu** command. Should **cu** not function between systems, a hardware problem is indicated. Otherwise, you may test the software installation by trying the **uucp** command. If the **uucp** command functions, the rest of the UNICOM commands will function. Follow these steps to test UNICOM:

1. Log in as user "root" and try the **cu** command on each master port to communicate with each slave system you have just installed. Also try the **cu** command on those ports dedicated to **cu** communication. For example, for a TNIX system the command might look like:

   ```
   cu -s 2400 -x -l /dev/tty2
   ```

   If **cu** is successful, a response of "Connected" will appear. Now log in on the remote system as though you were at a terminal connected to that system. If the login is successful your **cu** command works and you know that your hardware connections are functioning to that system.

   You can also use the **cu** command over an autodialer-to-modem telephone line connection. After entering your **cu** command, you will be connected to the autodialer rather than to the remote system. Now enter the control sequence to cause the autodialer to call the remote system. Once that telephone connection is made to the remote system, log in to the remote system. (Remember to log out from the remote system when you are finished.) If the **cu** command fails, check your hardware connections and the /etc/ttys file on both the master and slave systems. Once the **cu** command functions, go on to the steps dealing with testing the **uucp** command.

2. Send a file from the master side of the of the connection to the slave side with the **uucp** command. For example, here is a command that will copy the file /usr/spool/uucppublic/sendfile to a remote system:

   ```
   uucp -m  /usr/spool/uucppublic/sendfile  sysname!~uucp
   ```

   Sendfile will be found in the /usr/spool/uucppublic file on the slave side if the copy is successful. A small file should take only a few minutes to send between two idle systems.

3. Congratulations, your installation is successful! Go on and read the UNICOM Maintenance section of this manual.

TROUBLESHOOTING

If your **uucp** command did not work and your **cu** command did work, some of the entries in the database files you have created with **uucp-config** may be in error.

Check the entries in the files:

- /usr/lib/uucp/USERFILE
- /usr/lib/uucp/L.sys
- /usr/lib/uucp/L-devices
- /usr/lib/crontab

Here are some typical entries for these files (for both master and slave) for two 8560's connected through a cable using RS-232-C protocol on port 7, 2400 baud, using the default login stream, and polling every day on the hour:

- /usr/lib/uucp/USERFILE

```
uucp,slavename   /      (for master)

uucp,mastername   /      (for slave)
```

- /usr/lib/uucp/L.sys

```
mastername Any   Slave   2400    notty       (for slave)

slavename Any   tty7    2400    tty7   -EOT-ogin:-EOT-ogin: uucp
                                                    (for master)
```

- /usr/lib/uucp/L-devices

```
DIR   Slave   0   2400  (for slave)

DIR   tty7   0   2400   (for master)
```

- /usr/lib/crontab

This file will have a number of entries but among them will be:

```
(For slave)
5   1   *   *   *   /usr/lib/uucp/uuclean -p -m
0 * * * * /usr/lib/uucp/uucico -r1 -smastername
0 * * * * /usr/lib/uucp/uulog

(For master)
5   1   *   *   *   /usr/lib/uucp/uuclean -p -m
0 * * * * /usr/lib/uucp/uucico -r1 -sslavename
0 * * * * /usr/lib/uucp/uulog
```

Your files will have a different syntax if you have a different system configuration.

For information on the syntax of the lines in this file, refer to the "Database Files" discussion in the UNICOM Maintenance section of this manual.

If you find an error in these files, follow the steps in the "Modifying and Correcting Errors in the Installation" discussion in this section.

DEBUGGING THE LOGIN STREAM

A likely reason for failure of the **uucp** command installation from the master side is an incorrect login steam. The login stream you entered with the **uucp-config** program is placed as an entry in the L.sys file. You may monitor the progress of the login for debugging the login stream, with the following commands:

1. Log in as user "root" and enter the /usr/spool/uucp directory.

2. Enter:

```
rm STST*
rm LCK*
```

3. Enter: uucico -r1 -x5 -ssysname

sysname is the name of the remote slave system.

Step two removes anyfiles that may prevent a communication re-try to a slave system. Step three tries to contact the remote system and prints debugging messages. The debugging messages begin with "expect" and "sendthem" indicating the characters that are being sent and received in the login attempt. The message "OFN connect" will indicate a successful connection.

After getting a successful connection, repeat step two and try sending a file once more with the **uucp** command.

Section 4

UNICOM MAINTENANCE

Page

Introduction 4-1
Getting Started with UNICOM Maintenance 4-1
Maintenance Responsibilities 4-1

Checking and Restoring the UNICOM Commands 4-3
Verifying That a Problem Exists 4-3
Identifying the Problem System(s) 4-3
Restoring Uucp 4-4

Maintaining the Uucp Software on Existing Systems 4-4

Modifying Current UNICOM Systems and Adding New Systems 4-5
Adding a New System to your UNICOM Network 4-7

Security 4-7

UNICOM Administrative Commands 4-9
UUCLEAN - uucp queue directory clean-up 4-10
UULOG - uucp administrative program for reading LOGFILE information 4-11
UUPHONE - UUCP phone dialing program 4-12

Section 4

UNICOM MAINTENANCE

INTRODUCTION

This section tells you how to maintain the UNICOM software package. This section assumes that you are in charge of maintaining the UNICOM software, and that you are familiar with TNIX and the 8560 MUSDU System Users Manual.

The maintenance of UNICOM software is concerned with maintaining the **uucp** command.

Getting Started with UNICOM Maintenance

- As soon as you become responsible for UNICOM, log in as user "uucpa" (**uucp** administrator) and use the **passwd** command to create a password for the "uucpa" user. This is done for security purposes. As the "uucpa" user, you can modify files owned by the **uucp** programs. You will also need permission to log in as user "root" to perform some of the UNICOM maintenance.

- You may find it helpful to read the Bell Labs Documentation section of this manual to familiarize yourself with the working of **uucp**. In particular, read sections 8 and 9 of the "Uucp Implementation Description" article in that section.

Maintenance Responsibilities

As the person responsible for UNICOM maintenance, you will have several responsibilities:

1. Checking to see if the UNICOM communications are functioning properly. This will consist of checking the functioning of the **uucp** command.

2. Restoring the functioning of the **uucp** command if it fails.

3. Maintaining the **uucp** software on existing TNIX systems. Some database files owned by users "uucp" and "uucpa" must be deleted periodically.

4. Modifying the parameters of existing UNICOM software. For example you may wish to change the polling time of some system in your UNICOM network. You will be changing the parameters in the **uucp** database files.

5. Adding a new system to your UNICOM network. You may need to install UNICOM on a new system. You will need to modify existing systems to recognize the new system.

6. Maintaining system security. The files another system may access or execute in your system can be limited.

The remainder of this section contains the following topics:

- Checking and Restoring the UNICOM Commands Explains how to check when the **uucp** command is not working, and how to get the command working again.
- Maintaining the Uucp Software on Existing Systems Explains the housekeeping that must be done to keep UNICOM functioning.
- Modifying Current UNICOM Systems and Adding New Systems Explains how to modify the database files for your current UNICOM network and how to add a new system to your UNICOM network. Also explains the database files.
- Security Explains the administration of the file access protection incorporated in the UNICOM commands.
- UNICOM Administrative Commands Describes the **uulog**, **uuphone** and the **uuclean** commands, that are used to help maintain UNICOM.

CHECKING AND RESTORING THE UNICOM COMMANDS

VERIFYING THAT A PROBLEM EXISTS

A failure of a UNICOM command that queues its requests (**uucp**, **uux**, **mail**) is not apparent to the user when the command is issued. The command will accept the user parameters and return with a prompt. This indicates only that **uucp** has queued the request, not that the request was successfully completed. The request will be executed at a later time. The time depends on the frequency of polling, the length of the queues on both machines, the CPU use by other programs, and other factors. For these reasons, a communication request between busy machines can take considerable time before it is honored.

By contrast, the failure of the **cu** command is immediately apparent: you will not be able to log in to the remote machine.

If **uucp** communication is not taking place between two systems, eventually users will conclude that one or more of the UNICOM commands are not working and the users will notify you of this.

Sometimes users mistakenly believe that UNICOM commands are not working, when they actually are. This may happen if the user has not waited long enough for the data to transfer.

Another mistake in the use of UNICOM commands, resulting in failure of the command, is the use of file and directory protection modes. **Uucp** acts on file access as a user with "others" privileges, not the privileges of the user who issued the command. Any file or directory protection against "others" will prevent **uucp** from copying or writing to a file.

IDENTIFYING THE PROBLEM SYSTEM(S)

Communication between one pair of systems may work at the same time that communication between another pair does not. To see if communication between two systems is functioning, check the /usr/spool/uucp/LOGFILE in the master or slave system for records of transfer between the two systems. Use the **uulog** command to do this. Enter:

```
uulog -ssysname
```

Any communication attempt to the remote system **sysname** will be listed, along with the dates and times of the attempt. If the most current attempts say "OK" and "COPY SUCCEEDED", then communication with that system is working. If the lines listed say "FAILED", then communication with that system is not working. As a further verification, send a file via **uucp** from the master system. For example enter:

```
uucp -gA -m somefile sysname!~uucp
```

(The -gA option gives your request a higher priority than other users, and helps speed up the time for the request to be executed.)

RESTORING UUCP

At times, communication between systems will fail and not restart. The cause may be a temporary error in the communication link that **uucp's** error checking has detected. Although the error was temporary, communication to a system sysname may not restart because of special files left in the /usr/spool/uucp directory in both the master and slave system.

These files are named LCK..sysname, STST.sysname and LCK..ttyX, (where X is the number of the **sysname** port). To restart communication, log in as user "root" or "uucpa" and remove these files. Try the communication link again from the master system. Remember to use the -gA option with the **uucp** command.

MAINTAINING THE UUCP SOFTWARE ON EXISTING SYSTEMS

The maintenance (not the modification) of the **uucp** software consists of removing unneeded files created automatically by the software. These files are in the /usr/spool/uucp directory. A utility program, **uuclean,** is provided to help with the removal of files. The UNICOM installation procedure placed a crontab entry for the **cron** program that automatically executes **uuclean** once a day. This is the crontab line:

```
5   1   *   *   *   /usr/lib/uucp/uuclean -p -m
```

This crontab entry executes the command **uuclean -p -m** which removes all files older than 72 hours from the /usr/spool/uucp at 1:05 am every morning (if the system is on).

You may wish to be more selective about file removal. User "root" may modify this line in the crontab file. Refer to the on-line man page **cron** for information on the syntax of the crontab line. Refer to the **uuclean** command in the "Uucp Administrative Commands" discussion in this section.

The following files may require removal:

- LOGFILE, SYSLOG and files beginning with "LOG." The LOGFILE and SYSLOG files record the **uucp** transactions. These files grow quickly and without bound. They must be removed or edited periodically. This might be done automatically by having **cron** periodically execute a file containing a shell script such as:

```
mv /usr/spool/uucp/LOGFILE   /o.LOGFILE
mv /usr/spool/uucp/SYSLOG   /o.SYSLOG
```

o.LOGFILE is the old LOGFILE. You could then edit or remove the o.LOGFILE and o.SYSFILE files.

The "LOG." files are temporary and are merged with LOGFILE by the **uulog** command.

- Files beginning with "TM", "STST", and "LCK" may require removal. For more information refer to topic 9, "Uucp Administration" in the article "Uucp Implementation Description" in the Bell Labs Documentation section. The crontab line included by the UNICOM installation removes these files periodically.

- The standard queue files beginning with "D.", "C.", and "X.", may become useless if they are attempting communication that can not succeed. Reasons such as bad phone numbers, and login changes can cause communication failure -- for example, an "X." file requesting a command execution that is not permitted on a system. The crontab line included by the UNICOM installation removes these files periodically.

MODIFYING CURRENT UNICOM SYSTEMS AND ADDING NEW SYSTEMS

You can modify the parameters in the database files used by **uucp** to determine its operation. These parameters are the ones you entered during the interactive installation of UNICOM. Refer to the "Installation Disk" and "Uucpconfig Questions" discussions in the Installation section of this manual for the parameters you may change. For example, some of the parameters each UNICOM installation uses are port numbers, system names, telephone numbers, and file access permission for the directly connected remote systems in the UNICOM network. Refer to the "Modifying and Correcting Errors in the Installation" discussion in the Installation section of this manual for methods used to modify these parameters.

If you decide to modify the database files directly with a text editor, refer to the "Database File" discussion in this section and to the "Uucp Installation" topic in the "Uucp Implementation Description" article in the Bell Labs Documentation section for information about the files.

Uux Remote Execution Permission

The **uux** command allows a remote system to request your local system to execute commands. You may wish to add to the standard commands (**uucp** and **lp1r** by default) that the remote system may execute on your system. The commands another system may execute are limited to those listed in the /usr/lib/uucp/L-xcmds file. The interactive installation of UNICOM installs the file but does not permit you to modify it. You must modify it directly

with a text editor.

Log in as user "uucpa" or "root", and add the names of additional commands as separate lines in the file. The default file contains the following lines:

```
rmail
lp1r
uusend
fsend
fget
rnews
uucp
```

(The **rmail**, **uusend**, **fsend**, **fget**, and the **rnews** commands are necessary utilities, not used directly by a user.) To add the command **fgrep** to your system's repetoire of executable commands, add the line "fgrep" to the end of the file:

```
rmail
lp1r
uusend
fsend
fget
rnews
uucp
fgrep
```

DATABASE FILES

The following paragraphs describe briefly the purpose of the database files used by the **uucp** programs. You may modify these files to change the parameters used by **uucp**.

For information on the syntax of the following files refer to the "Uucp Installation" discussion in the "Uucp Implementation Discussion" article in the Bell Labs Documentation section of this manual.

- /usr/lib/uucp/USERFILE -- This file contains parameters that can restrict user access to files via **uucp**. These restrictions are in addition to those that result from the "uucp" user having only "others" access permission.

- /usr/lib/uucp/L.sys -- The parameters in this file are the names of the remote systems, the times of polling, the type and speed of the link, the log in procedure for the remote system, and the phone number of the remote system, if any.

- /usr/lib/uucp/L-devices -- The parameters in this file describe the type, speed and port number for the connections to remote systems.

- /usr/lib/uucp/L-xcmds -- This file lists the commands that remote systems can cause to be executed on the local system. The syntax of this file may be found in the "Uux Remote Execution Permission" discussion earlier in this section.

- /usr/lib/crontab -- This file is used by the **cron** program to execute commands periodically. The **uucp** installation places lines in this file that cause periodic polling of each remote system and periodic maintenance functions. At least the following three lines will be placed in the **crontab** file by the **uucp** installation:

```
5  1  *  *  *  /usr/lib/uucp/uuclean -p -m
0  *  *  *  *  /usr/lib/uucico -r1 -ssysname
0  *  *  *  *  /usr/lib/uulog
```

  Other lines may be added, depending upon your system configuration.

- /etc/ttys -- This file determines the login status and baud rate of the ports on your 8560 system.

SUPPORTING ADDITIONAL AUTO-DIALERS

Only two autodialer models are supported by the **uucp** command. You may wish to modify UNICOM software to allow **uucp** to use additional autodialers. The **/usr/lib/uucp/uuphone** program, included in the installation disk, interprets the line in the **L.sys** file that specifies the autodialer information. The C language source code, **/usr/lib/uucp/uuphone.c** (with comments), is included during the installation. To support additional autodialers, modify this source code, compile the code, and name the executable code /usr/lib/uucp/uuphone and **uucp** will recognize the new autodialer.

ADDING A NEW SYSTEM TO YOUR UNICOM NETWORK

Refer to the "Adding a New System to an Existing UNICOM Network" discussion in the Installation section of this manual for information on how to add a system to an existing UNICOM network.

SECURITY

The security issues for UNICOM are the read and write access permissions for the **uucp** and **uux** commands. The following paragraphs discuss some of these issues.

The UNICOM installation gives the "uucp" user the read and write permission of "others", and access to files owned by 'uucp" and "uucpa". This may be changed in the /etc/passwd file by user "root". If "uucp" is given the same user number as another user in this file, then the "uucp" communication

programs can read and write all the files of that user.

The /usr/lib/uucp/USERFILE can be used to add additional restrictions on the directories that may be read and written to by "uucp". Refer to the "Database Files" discussion in this section for more information.

The "uucpa" user name should have a password. This protects the "uucp" files from modification and the data contained in files owned by "uucp" from unauthorized access. (For example, some of the database files contain login information to remote systems.) You may wish to give the "uucp" user a password to prevent unauthorized use of **uucp** programs on your system. (Logging in as user "uucp" only starts the **uucico** program, not an interactive shell.) Remote systems will have to know this password, however.

The **uux** command has other systems execute commands. The /usr/lib/uucp/L-xcmds file lists those commands that may be executed on your system by remote systems. Do not permit commands to be executed that may breach the security of your system. For example, permitting the command **chmod** could create problems.

UNICOM ADMINISTRATIVE COMMANDS

UNICOM includes three commands that are useful for the administration of **uucp**. These commands are found in the /usr/lib/uucp directory. You must be in this directory to use these commands.

- **uulog** reports on the transactions of the **uucp** and **uux** commands. It tells what happened, when, and between whom. It also merges temporary files (those with names beginning with "LOG.") with the main /usr/spool/uucp/LOGFILE.

- **uuclean** is a utility used to remove unused files from the /usr/spool/uucp/ directory.

- **uuphone** is not used directly by the administrator. It is used by **uucp** to control autodialers when communicating over the phone lines. The standard **uuphone** program supports two models of autodialer. You may modify **uuphone** to permit **uucp** to use other autodialers.

UUCLEAN - UUCP QUEUE DIRECTORY CLEAN-UP

uuclean [-m] [-ntime] [-ppre] ...

DESCRIPTION

Uuclean scans the queue directory (/usr/spool/uucp/) for files with the specified prefix, and deletes all those files that are older than the specified number of hours.

OPTIONS

-ppre Scans for files with **pre** as the file prefix. Up to 10 **-p** arguments may be specified. **-p** without any **pre** deletes all files older than the specified time.

-ntime Files whose age is more than **time** hours are deleted if the prefix test is satisfied. (Default time is 72 hours.)

-m Sends mail to the owner of the file when the file is deleted.

This program is typically started by **cron**.

For more information, see the **uucp** command description in the Command Dictionary section of this manual.

NOTES

To prevent **uucico** from writing to files at the same time as they are being removed, the **uuclean** command should not be run at any time when **uucp** (**uucico**) may be activated.

FILES

/usr/lib/uucp - directory with commands used by uuclean internally
/usr/lib/uucp/spool - queue directory

UULOG - UUCP ADMINISTRATIVE PROGRAM FOR READING LOGFILE INFORMATION

uulog [-ssys] ... [-uuser] ...

DESCRIPTION

Uulog maintains a summary log of **uucp** and **uux** transactions in the file /usr/spool/uucp/LOGFILE by gathering information from partial log files named /usr/spool/uucp/LOG.*.?. (The partial log files will be created only if the **LOGFILE** is being used by another process.) **Uulog** removes the partial log files.

OPTIONS

-ssys Prints information about work involving system **sys**.

-uuser Prints information about work done for the specified **user**.

FILES

/usr/spool/uucp - queue directory

For more information see the article "Uucp Implementation Description" in the Bell Labs Documentation Section in this Manual.

UUPHONE - UUCP PHONE DIALING PROGRAM

uuphone [-xN] system dialer line phone#

DESCRIPTION

Uuphone is used by the **uucp** program when communicating with other systems via autodialer.

OPTIONS

-xN N = 0..9. This option provides debugging information. The larger the integer (0..9), the more information is provided.

system The remote system name for logging purposes.

dialer The third field from the /usr/lib/uucp/L-devices file, Three possibilities are recognized: "VADIC", "HAYES", or a device number.

line The device to which standard input and standard output is directed.

phone# The phone number string.

This program assumes that its standard input and its standard output have come from the autodialer.

NOTES

This program is invoked by the **uucico** program (a program invoked by the user program **uucp**) when communicating with another system via an autodialer. **uuphone** currently supports the Hayes Smartmodem (300 baud version), and the Racal-Vadic 3451 with autodial option. The **uuphone** program recognizes the words "HAYES" and "VADIC" in the **dialer** field to determine the format of the protocol with the autodialer. You must modify and recompile the C language source code in /usr/lib/uucp/uuphone.c to permit **uuphone** to recognize additional autodialers and to create the proper format for these dialers. Rename the new version of the **uuphone** program to /usr/lib/uucp/uuphone. (The C compiler is a software option available in the 8560 MUSDU Native Programming Package.)

For more information on the **uucp** command, refer to the Command Dictionary section of this manual.

FILES

/usr/lib/uucp/uuphone.c - C language source for uuphone
/usr/lib/uucp/uuphone - object code for uuphone

Section 5

TECHNICAL NOTES

Page

Note 1 -- Cabling Connections 5-1
Cable Connections 5-1
8560 HSI to 8560 HSI 5-2
8560 RS-232 to UNIX RS-232 or 8560 RS-232 5-2
Modem to 8560 and Auto-Dialer-Modem to 8560 5-2

Note 2 -- UNICOM and 8560 System Performance 5-3

Note 3 - Notes on Uucp 5-4

Section 5

TECHNICAL NOTES

This section contains the following technical notes:

1. Cabling Connections -- explains the pin connections for cables you must use between systems communicating with UNICOM software.

2. UNICOM and 8560 System Performance -- explains the effect of UNICOM on the performance of the 8560 system.

3. Notes on Uucp -- a collection of various comments on the **uucp** system of programs.

NOTE 1 -- CABLING CONNECTIONS

TNIX systems may be connected with cables using HSI protocol, RS 232-C protocol, or via phone lines (using RS 232-C protocol) with the use of an autodialer-modem on the master system and a modem with autoanswer on the slave system.

UNIX and TNIX systems may be connected with cables using RS 232-C protocol or via phone lines (using RS 232-C protocol) with an autodialer-modem and a modem with autoanswer. Any one TNIX or UNIX system may be connected to several other TNIX or UNIX systems. Each system may use one or more of the three methods of connection (HSI, RS-232, phone lines).

CABLE CONNECTIONS

The cables you use to connect your 8560 to other systems must have the correct pin connections. All connections are done through the HSI ports on the rear panel of the 8560. Each pin on the 8560 port must be connected through a cable to the correct pin on the port of the communicating system. You may use either standard RS-232-C or HSI cable in combination with a pin adapter.

You will need to make the adapters to connect the correct pins on the 8560 HSI connectors to the correct pins on the cables. Except for the modem adapter, the adapters may be placed either at the master or slave side of the connection. The following lists explain the pin adapters for each of the different combinations of communication links:

8560 HSI to 8560 HSI

8560		8560
1	◄──►	1
2	◄──►	3
3	◄──►	2
5	◄──►	20
7	◄──►	7
11	◄──►	12
12	◄──►	11
13	◄──►	25
20	◄──►	5
25	◄──►	13

8560 RS-232 to UNIX RS-232 or 8560 RS-232

System		System
1	◄──►	1
2	◄──►	3
3	◄──►	2
7	◄──►	7

(Also connect pins 5, 8 and 20 together on each system.)

5	5
|	|
8	8
|	|
20	20

Modem to 8560 and Auto-Dialer-Modem to 8560

Modem Auto-dialer		8560
1	◄──►	1
2	◄──►	3
3	◄──►	2
20	◄──►	5
4	◄──►	6
7	◄──►	7
8	◄──►	20

NOTE 2 -- UNICOM AND 8560 SYSTEM PERFORMANCE

The **uucp** program uses considerable CPU resources during execution. **uucp** execution is initiated not only by user request, but by the **cron** program. (The **cron** program is controlled by the /usr/lib/crontab file.) Therefore, a significant load on the CPU resources by **uucp** may occur at unpredictable times.

If the effect of **uucp** on system performance is greater than desired during normal working hours, the system manager may modify the the second field in the /usr/lib/uucp/L.sys file so that **uucp** file transmission occurs only during off hours. Refer to the L.sys file description in the "Uucp Implementation Description" article in the Bell Labs Documentation section of this manual.

NOTE 3 - NOTES ON UUCP

UUCP

```
$ uucp file othersys!~uucp/file
```

This command simply transfers a file to the system 'othersys'. It **cannot** route files through multiple systems like 'rmail' does. It simply deposits the file in the desired directory on the remote system. The destination address consists of the name of the remote system, a '!', and an extended file specification. The notation '~user' will be expanded to the HOME directory of the named user. Thus, ~uucp expands (on most systems) to /usr/spool/uucppublic.

uucp creates one or two files in /usr/spool/uucp:

C. othersysnXXXX
D. othersysBXXXX

(XXXX is a unique ID)
The second file is present only if the first file must be copied from its current location, or if it came on the standard input.

UUX

```
$ uux  'othersys!lpr -r' < file
```

This command arranges to execute the program 'lpr' on the remote system, feeding it the standard input from 'file'. Only commands which the remote system has specifically allowed to be executed are legal.

uux creates three files in /usr/spool/uucp:

C. othersysAYYYY
D. othersysBYYYY
D. localsysXYYYY

(YYYY is a unique ID)

the first file contains commands to uucp to transfer the other two files over to the remote system. The second file contains the data which was input to uux and which will be input to the desired program. The third file contains the commands which will execute the desired program on the remote system.

UUXQT

This is the UUCP remote executor. It scans the spool directory for files of the form:

X.othersysYYYY

and executes them on the local system. Thus, execution is done by copying a file of this form onto a remote system. UUXQT will only execute commands which have been explicitly allowed in the L-xcmds file.

L.sys

This is the main UUCP date file. It contains one line for each 'neighbor' system of the local site. These lines have the form:

sitename time-to-call device speed phone login-info

e.g.

testy Any tty2 2400 HSI -EOT-login:-EOT-login: uucp

Take care that the separators between fields are SPACES, not TABS. Note however, that the phone number can be replaced with the word HSI to enable HSI communications.

Here is a brief description of each field:

sysname	names of remote system
time-to-call	times of day when remote may be called. Days can be (Su Mo Tu We Th Fr Sa), or Wk for any week-day or Any for any day. time is specified as a range (e.g. 0800-1700)
device	ACU or hard-wired device to use for call. 8560 UUCP does not currently support ACU's.
speed	line speed (same as in other places)
phone	phone number to call, 'HSI' for HSI ports
LOGIN	This field gives uucico information on how to log in to the remote system. Refer to the "Login Stream Syntax" discussion in the Installation section of this manual for the format of this field.

L-devices

This file provides information about devices which are available for UUCP use. e.g.

```
DIR tty2 0 2400
```

This is a DIRECT line, tty2, running at 2400 baud, and with no associated call unit (autodialer). These fields are discussed in more detail under the topic "Uucp Installation" in the Bell Labs Documentation Section of this manual.

USERFILE:

This file contains information which is only used by the SLAVE to determine if the calling system has permission to access files below certain directories.

The format is

```
login-name,sysname path-name [path-name]
```

e.g.

```
uucp,happy /usr/spool/uucppublic
uucp,grumpy  /
uucp3,   /
```

In the above example, the system 'happy', if it uses the 'uucp' login, can only access files in the spool and public directories. In contrast, the system 'grumpy', while also loging in as 'uucp' can access any files. Any system logging in as uucp3 can access any files. Note that the system is supplied with only a 'uucp' login. Other, more restricted, logins need to be created by hand.

Section 6 Bell Labs Documentation

Page

Introduction 6-1

UNICOM - UNIX Differences in Bell Labs Documentation 6-2
A Dial-up Network of UNIX Systems 6-2
Uucp Implementation Description 6-3

A Dial-up Network of UNIX Systems 6-5
1. Purpose 6-6
2. Design Goals 6-6
3. Processing 6-8
4. Present Uses 6-11
5. Performance 6-12
6. Further Goals 6-12
7. Lessons 6-13

Uucp Implementation Description 6-15
Introduction 6-15
1. Uucp -- UNIX to UNIX File Copy 6-16
2. Uux -- UNIX to UNIX Execution 6-18
3. Uucico -- Copy In, Copy Out 6-20
4. Uuxqt -- Uucp Command Execution 6-23
5. Uulog -- Uucp Log Inquiry 6-23
6. Uuclean -- Uucp Spool Directory Cleanup 6-24
7. Security 6-24
8. Uucp Installation 6-25

Files Required For Execution 6-27
L-devices 6-27
USERFILE 6-28
L.sys 6-29

9. Administration 6-31
TM -- Temporary Data Files 6-31
LOG -- Log Entry Files 6-31
STST -- System Starts Files 6-32
LCK -- Lock Files 6-32
ERRLOG -- uucp System Error File 6-32
Shell Files 6-33
Login Entry 6-33
File Modes 6-33

Section 6

BELL LABS DOCUMENTATION

INTRODUCTION

The UNICOM **uucp** and **uux** user commands (along with the associated maintenance and internal commands) implement computer networking among UNIX systems. The ported UNICOM commands are compatible with UNIX networks and allow TNIX--UNIX networking.

This section includes two articles that describe the use, maintenance, and functioning of these UNIX commands.

The articles in this section were written by the authors of the UNIX **uucp** system of programs. The UNICOM programs are almost identical in functioning and maintenance to the programs developed by Bell Labs.

These articles may be used as a definitive source of information about how TNIX UNICOM commands function.

UNICOM - UNIX DIFFERENCES IN BELL LABS DOCUMENTATION

The following list explains the differences between the UNICOM and UNIX implementation of the **uucp** system of commands as described in the articles in this section:

A DIAL-UP NETWORK OF UNIX SYSTEMS

2. Design Goals

- The rates of data transmission do not apply to UNICOM.
- The sequence count software in the UNICOM software has not been impelmented.

3. Processing

- The dial-codes file has not been implemented in UNICOM.
- Figures 1 and 2 are not included with the article.

5. Performance

The rates of data transmission do not apply to UNICOM.

UUCP IMPLEMENTATION DESCRIPTION

1. Uucp-UNIX to UNIX File Copy

- The "-n" and the "-e" **uucp** options listed may not function for UNICOM.
- The file uucp.h does not exist in UNICOM.

2. Uux-UNIX to UNIX Execution

- The file uucp.h does not exist in UNICOM.
- The file /usr/lib/uucp/L-xcmds is used by UNICOM to determine what commands **uuxqt** may execute.

3. Uucico-Copy In, Copy Out

- The UNICOM L-dialcodes file has not been implemented.
- The "call back required" mechanism for UNICOM has not been implemented.

7. Security

The call back and conversation sequence options for UNICOM have not been implemented.

8. Uucp Installation

- UNICOM does not provide the source code for the **uucp** system of commands.
- UNICOM has no source library directory.
- UNICOM has no uucp.h or makefile.
- The L-dialcodes file has not been implemented for UNICOM.
- The call back in USERFILE has not been implemented for UNICOM.

9. Administration

The SQFILE has not been implemented for UNICOM.

A Dial-Up Network of UNIX® Systems

D. A. Nowitz

M. E. Lesk

Bell Laboratories
Murray Hill, New Jersey 07974

ABSTRACT

A network of over eighty UNIX† computer systems has been established using the telephone system as its primary communication medium. The network was designed to meet the growing demands for software distribution and exchange. Some advantages of our design are:

- *The startup cost is low. A system needs only a dial-up port, but systems with automatic calling units have much more flexibility.*
- *No operating system changes are required to install or use the system.*
- *The communication is basically over dial-up lines, however, hardwired communication lines can be used to increase speed.*
- *The command for sending/receiving files is simple to use.*

Keywords: networks, communications, software distribution, software maintenance.

August 18, 1978

†UNIX is a Trademark of Bell Laboratories.

A Dial-Up Network of UNIX® Systems

D. A. Nowitz

M. E. Lesk

Bell Laboratories
Murray Hill, New Jersey 07974

1. Purpose

The widespread use of the UNIX† system[1] within Bell Laboratories has produced problems of software distribution and maintenance. A conventional mechanism was set up to distribute the operating system and associated programs from a central site to the various users. However this mechanism alone does not meet all software distribution needs. Remote sites generate much software and must transmit it to other sites. Some UNIX systems are themselves central sites for redistribution of a particular specialized utility, such as the Switching Control Center System. Other sites have particular, often long-distance needs for software exchange; switching research, for example, is carried on in New Jersey, Illinois, Ohio, and Colorado. In addition, general purpose utility programs are written at all UNIX system sites. The UNIX system is modified and enhanced by many people in many places and it would be very constricting to deliver new software in a one-way stream without any alternative for the user sites to respond with changes of their own.

Straightforward software distribution is only part of the problem. A large project may exceed the capacity of a single computer and several machines may be used by the one group of people. It then becomes necessary for them to pass messages, data and other information back and forth between computers.

Several groups with similar problems, both inside and outside of Bell Laboratories, have constructed networks built of hardwired connections only.[2,3] Our network, however, uses both dial-up and hardwired connections so that service can be provided to as many sites as possible.

2. Design Goals

Although some of our machines are connected directly, others can only communicate over low-speed dial-up lines. Since the dial-up lines are often unavailable and file transfers may take considerable time, we spool all work and transmit in the background. We also had to adapt to a community of systems which are independently operated and resistant to suggestions that they should all buy particular hardware or install particular operating system modifications. Therefore, we make minimal demands on the local sites in the network. Our implementation requires no operating system changes; in fact, the transfer programs look like any other user entering the system through the normal dial-up login ports, and obeying all local protection rules.

† **UNIX is a Trademark of Bell Laboratories.**

We distinguish "active" and "passive" systems on the network. Active systems have an automatic calling unit or a hardwired line to another system, and can initiate a connection. Passive systems do not have the hardware to initiate a connection. However, an active system can be assigned the job of calling passive systems and executing work found there; this makes a passive system the functional equivalent of an active system, except for an additional delay while it waites to be polled. Also, people frequently log into active systems and request copying from one passive system to another. This requires two telephone calls, but even so, it is faster than mailing tapes.

Where convenient, we use hardwired communication lines. These permit much faster transmission and multiplexing of the communications link. Dial-up connections are made at either 300 or 1200 baud; hardwired connections are asynchronous up to 9600 baud and might run even faster on special-purpose communications hardware.[4,5] Thus, systems typically join our network first as passive systems and when they find the service more inportant, they acquire automatic calling units and become active systems; eventually, they may install highspeed links to particular machines with which they handle a great deal of traffic. At no point, however, must users change their programs or procedures.

The basic operation of the network is very simple. Each participating system has a spool directory, in which work to be done (files to be moved, or commands to be executed remotely) is stored. A standard program, *uucico*, performs all transfers. This program starts by identifying a particular communication channel to a remote system with which it will hold a conversation. *Uucico* then selects a device and establishes the connection, logs onto the remote machine and starts the *uucico* program on the remote machine. Once two of these programs are connected, they first agree on a line protocol, and then start exchanging work. Each program in turn, beginning with the calling (active system) program, transmits everything it needs, and then asks the other what it wants done. Eventually neither has any more work, and both exit.

In this way, all services are available from all sites, passive sites, however, must wait until called. A variety of protocols may be used; this conforms to the real, non-standard world. As long as the caller and called programs have a protocol in common, they can communicate. Furthermore, each caller knows the hours when each destination system should be called. If a destination is unavailable, the data intended for it remain in the spool directory until the destination machine can be reached.

The implementation of this Bell Laboratories network between independent sites, all of which store proprietary programs and data, illustratives the pervasive need for security and administrative controls over file access. Each site, in configuring its programs and system files, limits and monitors transmission. In order to access a file a user needs access permission for the machine that contains the file and access permission for the file itself. This is achieved by first requiring the user to use his password to log into his local machine and then his local machine logs into the remote machine whose files are to be accessed. In addition, records are kept identifying all files that are moved into and out of the local system, and how the requestor of such accesses identified himself. Some sites may arrange to permit users only to call up and request work to be done; the calling users are then called back before the work is actually done. It is then possible to verify that the request is legitimate from the standpoint of the target system, as well as the originating system. Furthermore, because of the call-back, no site can masquerade as another even if it knows all the necessary passwords.

Each machine can optionally maintain a sequence count for conversations with other machines and require a verification of the count at the start of each conversation. Thus, even if call back is not in use, a successful masquerade requires the calling party to present the correct sequence number. A would-be impersonator must not just steal the correct phone number, user name, and password, but also the sequence count, and must call in sufficiently promptly to precede the next legitimate request from either side. Even a successful masquerade will be detected on the next correct conversation.

3. Processing

The user has two commands which set up communications, *uucp* to set up file copying, and *uux* to set up command execution where some of the required resources (system and/or files) are not on the local machine. Each of these commands will put work and data files into the spool directory for execution by *uucp* daemons. Figure 1 shows the major blocks of the file transfer process.

File Copy

The *uucico* program is used to perform all communications between the two systems. It performs the following functions:

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

Uucico may be started in several ways;

a. by a system daemon,

b. by one of the *uucp* or *uux* programs,

c. by a remote system.

Scan For Work

The file names in the spool directory are constructed to allow the daemon programs *(uucico, uuxqt)* to determine the files they should look at, the remote machines they should call and the order in which the files for a particular remote machine should be processed.

Call Remote System

The call is made using information from several files which reside in the uucp program directory. At the start of the call process, a lock is set on the system being called so that another call will not be attempted at the same time.

The system name is found in a "systems" file. The information contained for each system is:

1. system name,
2. times to call the system (days-of-week and times-of-day),
3. device or device type to be used for call,
4. line speed,
5. phone number,
6. login information (multiple fields).

The time field is checked against the present time to see if the call should be made. The *phone number* may contain abbreviations (e.g. "nyc", "boston") which get translated into dial sequences using a "dial-codes" file. This permits the same "phone number" to be stored at every site, despite local variations in telephone services and dialing conventions.

A "devices" file is scanned using fields 3. and 4. from the "systems" file to find an available device for the connection. The program will try all devices which satisfy 3. and 4. until a connection is made, or no more devices can be tried. If a non-multiplexable device is successfully opened, a lock file is created so that another copy of *uucico* will not try to use it. If the connection is complete, the *login information* is used to log into the remote system. Then a command is sent to the remote system to start the *uucico* program. The conversation between the two *uucico* programs begins with a handshake started by the called, *SLAVE*, system. The *SLAVE sends a message to let the MASTER* know it is ready to receive the system identification and conversation sequence number. The response from the *MASTER* is verified by the *SLAVE* and if acceptable, protocol selection begins.

Line Protocol Selection

The remote system sends a message

P*proto-list*

where *proto-list* is a string of characters, each representing a line protocol. The calling program checks the proto-list for a letter corresponding to an available line protocol and returns a *use-protocol* message. The *use-protocol* message is

U*code*

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

Greg Chesson designed and implemented the standard line protocol used by the uucp transmission program. Other protocols may be added by individual installations.

Work Processing

During processing, one program is the *MASTER* and the other is *SLAVE*. Initially, the calling program is the *MASTER*. These roles may switch one or more times during the conversation.

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

S send a file,

R receive a file,

C copy complete,

H hangup.

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

The send and receive replies are based on permission to access the requested file/directory. After each file is copied into the spool directory of the receiving system, a copy-complete message is sent by the receiver of the file. The message *CY* will be sent if the UNIX *cp* command, used to copy from the spool directory, is successful. Otherwise, a *CN* message is sent. The requests and results are logged on both systems, and, if requested, mail is sent to the user reporting completion (or the user can request status information from the log program at any time).

The hangup response is determined by the *SLAVE* program by a work scan of the spool directory. If work for the remote system exists in the *SLAVE's* spool directory, a *HY* response is sent.

A sample conversation is shown in Figure 2.

Conversation Termination

When a *HY* message is received by the *MASTER* it is echoed back to the *SLAVE* and the protocols are turned off. Each program sends a final "OO" message to the other.

4. Present Uses

One application of this software is remote mail. Normally, a UNIX system user writes "mail dan" to send mail to user "dan". By writing "Mail usg!dan" the mail is sent to user "dan" on system "usg".

The primary uses of our network to data have been in software maintenance. Relatively few of the bytes passed between systems are intended for people to read. Instead, new programs (or new versions of programs) are sent to users, and potential bugs are returned to authors. Aaron Cohen has implemented a "stockroom" which allows remote users to call in and request software. He keeps a "stock list" available programs, and new bug fixes and utilities are added regularly. In this way, users can always obtain the latest version of anything without bothering the authors of the programs. Although the stock list is maintained on a particular system, the items in the stockroom may be warehoused in many places; typically each program is distributed from the home site of its author. Where necessary, uucp does remote-to-remote copies.

We also routinely retrieve test cases from other systems to determine whether errors on remote systems are caused by local misconfigurations or old versions of software, or whether they are bugs that must be fixed at the home site. This helps identify errors rapidly. For one set of test programs maintained by us, over 70% of the bugs reported from remote sites were due to old software, and were fixed merely by ditributing the current version.

Another application of the network for software maintenance is to compare files on two different machines. A very useful utility on one machine has been Doug McIlroy's "diff" program which compares two text files and indicates the differences, line by line, between them.[6] Only lines which are not identical are printed. Similarly, the program "uudiff" compares files (or directories) on two machines. One of these directories may be on a passive system. The "uudiff" program is set up to work similarly to the inter-system mail, but it is slightly more complicated.

To avoid moving large numbers of usually identical files, *uudiff* computes file checksums on each side, and only moves files that are different for detailed comparison. For large files, this process can be iterated; checksums can be computed for each line, and only those lines that are different actually moved.

The "uux" command has been useful for providing remote output. There are some machines which do not have hard-copy devices, but which are connected over 9600 baud communication lines to machines with printers. The *uux* command allows the formatting of the printout on the local machine and printing on the remote machine using standard UNIX command programs.

5. Performance

Throughput, of course, is primarily dependent on transmission speed. The table below shows the real throughput of characters on communication links of different speeds. These numbers represent actual data transferred; they do not include bytes used by the line protocol for data validation such as checksums and messages. At the higher speeds, contention for the processors on both ends prevents the network from driving the line full speed. If desired, operating system modifications can be installed that permit full use of even very fast links.

Nominal speed	Characters/sec.
300 baud	27
1200 baud	100-110
9600 baud	200-850

In addition to the transfer time, there is some overhead for making the connection and logging in ranging from 15 seconds to 1 minute. Even at 300 baud, however, a typical 5,000 byte source program can be transferred in four minutes instead of the 2 days that might be required to mail a tape.

Traffic between systems is variable. Between two closely related systems, we observed 20 files moved and 5 remote commands executed in a typical day. A more normal traffic out of a single system would be around a dozen files per day.

The total number of sites at present in the main network is 82, which includes most of the Bell Laboratories full-size machines which run the UNIX operating system. Geographically, the machines range from Andover, Massachusetts to Denver, Colorado.

Uucp has also been used to set up another network which connects a group of systems in operational sites with the home site. The two networks touch at one Bell Labs computer.

6. Further Goals

Eventually, we would like to develop a full system of remote software maintenance. Conventional maintenance (a support group which mails tapes) has many well-known disadvantages.[7] There are distribution errors and delays, resulting in old software running at remote sites and old bugs continually reappearing. These difficulties are aggravated when there are 100 different small systems, instead of a few large ones.

The availability of file transfer on a network of compatible operating systems makes it possible just to send programs directly to the end user who wants them. This avoids the bottleneck of negotiation and packaging in the central support group. The "stockroom" serves this function for new utilities and fixes to old utilities. However, it is still likely that distributions will not be sent and installed as often as needed. Users are justifiably suspicious of the "latest version" that has just arrived; all too often it features the "latest bug." What is needed is to address both problems simultaneously:

1. Send distributions whenever programs change.
2. Have sufficient quality control so that users will install them.

To do this, we recommend systematic regression testing both on the distributing and receiving systems. Acceptance testing on the receiving systems can be automated and permits the local system to ensure that its essential work can continue despite the constant installation of changes sent from elsewhere. The work of writing the test sequences should be recovered in lower counseling and distribution costs.

Some slow-speed network services are also being implemented. We now have intersystem "mail" and "diff", plus the many implied commands represented by "uux." However, we still need intersystem "write" (real-time inter-user communication) and "who" (list of people logged in on different systems). A slow-speed network of this sort may be very useful for speeding up counseling and education, even if not fast enough for the distributed data base applications that attract many users to networks. Effective use of remote execution over slow-speed lines, however, must await the general installation of multiplexable channels so that long file transfers do not lock out short inquiries.

7. Lessons

The following is a summary of the lessons we learned in building these programs.

1. By starting your network in a way that requires no hardware or major operating system changes, you can get going quickly.
2. Support will follow use. Since the network existed and was being used, system maintainers were easily persuaded to help keep it operating, including purchasing additional hardware to speed traffic.
3. Make the network commands look like local commands. Our users have a resistance to learning anything new: all the inter-system commands look very similar to standard UNIX system commands so that little training cost is involved.
4. An initial error was not coordinating enough with existing communications projects: thus, the first version of this network was restricted to dial-up, since it did not support the various hardware links between systems. This has been fixed in the current system.

Acknowledgements

We thank G. L. Chesson for his design and implementation of the packet driver and protocol, and A. S. Cohen, J. Lions, and P. F. Long for their suggestions and assistance.

References

1. D. M. Ritchie and K. Thompson, "The UNIX Time-Sharing System," *Bell Sys. Tech. J.* 57(6) pp. 1905-1929 (1978).
2. T. A. Dolotta, R. C. Haight, and J. R. Mashey, "UNIX Time-Sharing System: The Programmer's Workbench," *Bell Sys. Tech. J.* 57(6) pp. 2177-2200 (1978).
3. G. L. Chesson, "The Network UNIX System," *Operating Systems Review* 9(5) pp. 60-66 (1975). Also in *Proc. 5th-Symp. on Operating Systems Principles.*
4. A. G. Fraser, "Spider—An Experimental Data Communications System," *Proc. IEEE Conf. on Communications*, p. 21F (June 1974). IEEE Cat. No. 74CH0859-9-CSCB.
5. A. G. Fraser, "A Virtual Channel Network," *Datamation*, pp. 51-56 (February 1975).
6. J. W. Hunt and M. D. McIlroy, "An algorithm for Differential File Comparison," Comp. Sci. Tech. Rep. No. 41, Bell Laboratories, Murray Hill, New Jersey (June 1976).
7. F. P. Brooks, Jr., *The Mythical Man-Month*, Addison-Wesley, Reading, Mass. (1975).

Uucp Implementation Description

D. A. Nowitz
Bell Laboratories
Murray Hill, New Jersey 07974

ABSTRACT

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

Introduction

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

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

There are four primary programs:

uucp — builds *work files* and gathers *data files* in the spool directory for data transmission.

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

uucico — executes the work files for data transmission.

uuxqt — executes the scripts for UNIX command execution.

There are a couple of administrative programs:

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

uuclean — removes old files from the spool directory.

† **UNIX is a Trademark of Bell Laboratories.**

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

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

1. Uucp—UNIX to UNIX File Copy

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

uucp [optiion] ... source... destination

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

Uucp has several options:

-d	Make directories when necessary for copying the file.
-c	Don't copy source files to the spool directory, but use the specified source when the actual transfer takes place.
-esys	Send this job to system *sys* to execute. (Note that this will only work when the system *sys* allows *uuxqt* to execute a *uucp* command. See the "Uuxqt" and "Security" sections.)
-gletter	Put *letter* in as the grade in the name of the work file. (This can be used to change the order of work for a particular machine.)
-m	Send mail to the requester on completion of the work.
-nsec	Notify *user* on the remote machine that a file has been sent.

There are several options available for debugging:

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

The destination may be a directory name, in which case the file name is taken from the last part of the source's name. If the directory exists, it must be writable by everybody. (Note that if the destination is a directory name and the "-d" option is specified to create the directory, the directory name must be followed by "/".) The source name may contain special shell characters such as "?*[]". These will be expanded on the appropriate system.

The command

uucp *.c usg!/usr/dan

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

The source and/or destination names may also contain a ~*user* prefix. This translates to the login directory of *user* on the specified system. File names beginning with "~/" translate into the public directory (usually /usr/spool/uucppubic) on the remote system. For names with partial path-names, the current directory is prepended to the file names. File names with ../ are not permitted for security reasons.

The command
uucp usg!~dan/*.h ~dan

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

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

1. Copy source to destination on local system.
2. Receive files from other systems.
3. Send files to a remote system.
4. Send files from remote systems to another remote system.
5. Receive files from remote systems when the source contains special shell characters as mentioned above.
6. Request that the *uucp* command be executed by a remote system.

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

Type 1—Local Copy

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

Type 2—Receive Files

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

1. R
2. The full path-name of the source or a ~something/path-name. The ~*something* part will be expanded on a remote system.
3. The full path-name of the destination file. If the ~*something* notation is used, it will be immediately expanded.
4. The user's login name.
5. A "-" followed by an option list. The options -m and -d may appear.

Type 3—Send Files

Each source file is copied into a *data file* in the spool directory. (A "-c" option on the *uucp* command will prevent the *data file* from being made. In this case, the file will be transmitted from the indicated source.) The fields for these entries are given below.

1. S
2. The full-path name of the source file.
3. The full-path name of the destination or ~something/file-name.
4. The user's login name.
5. A "-" followed by an option list. The options -d, -m, and -n may appear.
6. The name of the *data file* in the spool directory. A dummy name, "D.0" is used when the -c option is specified.
7. The file mode bits of the source file in octal print format (e.g., 0666).
8. The user on the remote system to be notified upon completion of the file copy when the "-n" option is specified.

Type 4 and Type 5—Remote uucp Required

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

Type 6—Remote Execution

This occurs when the "-e" option is used. In this case, the *uux* facility is used to create and send the request. This requires that the remote *uuxqt* program allows the *uucp* command.

2. Uux—UNIX To UNIX Execution

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

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

where the command-string is made up of one or more arguments. All special shell characters such as "< >¦~" must be quoted either by quoting the entire command-string or quoting the character as a separate argument. Within the command-string, the command and file names may contain a *system-name!* prefix. All arguments that do not contain a "!" will not be treated as files. (They will not be copied to the execution machine.) An argument that contains a "!" but is not to be treated as a file at the present time, can be escaped by using "()" around the argument. (Note that the "()" symbols must usually be escaped with a "\" symbol.) The "-" is used to indicate that the standard input for *command-string* should be inherited from the standard input of the *uux* command. The following options are available for debugging:

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

-*xnum* *Num* is a level number between 1 and 9; higher numbers give more debugging output.

The command

pr abc ¦ uux - usg!lpr

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

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

For required files that are not on the execution machine, *uux* will generate receive command files (type 2 above). These command-files will be put on the execution machine for execution by the *uucico* program.

The *execute file* contains a script that will be processed by the *uuxqt* program. It is made up of several lines, each of which contains an identification character and one or more arguments. The lines are described below.

User Line

U user system

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

Required File Line

F file-name real-name

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

Standard Input Line

I file-name

The standard input is either specified by a "<" in the command-string or inherited from the standard input of the *uux* command if the "-" option is used. If a standard input is not specified, "/dev/null" is used. (Note that if there is a standard input specified, it will also appear in an "F" line.)

Standard Output Line

O file-name system-name

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

Command Line

C command [arguments]...

The arguments are those specified in the command-string. The standard input and standard output will not appear on this line. All *required files* will be moved to the execution directory (usually /usr/lib/uucp/.XQTDIR) and the UNIX command is executed using the shell specified in the *uucp.h* header file. In addition, a shell "PATH" statement is prepended to the command line as specified in the *uuxqt* program. (Note that a check is made to see that the command is allowed as specified in the *uuxqt* program.) After execution, the standard output is copied or sent to the proper place.

3. Uucico—Copy In, Copy Out

The *uucico* program will perform several major functions:

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

Uucico may be started in several ways:

a. by a system demon specified in a crontab entry,

b. by one of the *uucp, uux, uuxqt* or *uucico* programs,

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

d. by a remote system. (The uucico program should be specified as the "shell" field in the "/etc/passwd" file for the logins used by remote systems to access uucp.)

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

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

Uucico is generally started by another program. There are several options used for execution:

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

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

The following options are used primarily for debugging:

-d*dir* Use directory *dir* for the spool directory.

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

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

Scan For Work

The names of the work related files in the spool directory have format

type . system-name grade number

where

type is an upper case letter (*C*—copy command file, *D*—data file, *X*—execute file),

system-name is the remote system,

grade is a character,

number is a four digit, zero padded sequence number.

The file

C.res45n0031

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

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

Call Remote System

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

The *L.sys* file contains information required to make the remote connection:

[1] system name,

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

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

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

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

[6] login information (zero or more fields).

The time field is checked against the present time to see if the call should be made. The *phone number* may contain abbreviations (e.g., mh, py, boston) that get translated into dial sequences using the *L-dialcodes* file.

The *L-devices* file is scanned using fields [3] and [4] from the *L.sys* file to find an available device for the call. The program will try each devices that satisfy [3] and [4] until a call is made, or no more devices can be tried. If a device is successfully opened, a lock file is created. If the call is completed, the *login information* (field [6] of *L.sys*) is used to login.

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

Line Protocol Selection

The remote system sends a message.

P*proto-list*

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

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

U*code*

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

Work Processing

The *MASTER* program does a work search similar to the one used in the "Scan For Work" section. (The *MASTER* has been specified by the "-r1" uucico option.) Each message used during the work processing is specified by the first character of the message:

S send a file,

R receive a file,

C copy complete,

X execute a *uucp* command,

H hangup.

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

The send and receive replies are based on permission to access the requested file/directory using the *USERFILE* and read/write permissions of the file/directory. After each file is copied into the spool directory of the receiving system, a copy-complete message is sent by the receiver of the file. The message *CY* will be sent if the file has successfully been moved from the spool directory to the destination. Otherwise, a *CN* message is sent. (In this case, the file is put in the public directory, usually /usr/spool/uucppublic, and the requester is notified by mail.) The requests and results are logged on both systems.

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

Conversation Termination

When a *HY* message is received by the *MASTER* it is echoed back to the *SLAVE* and the protocols are turned off. Each program sends a final "OO" message to the other. The original *SLAVE* program will clean up and terminate. The *MASTER* will proceed to call other systems unless a "-s" option was specified.

4. Uuxqt—Uucp Command Execution

The *uuxqt* program is used to execute scripts generated by *uux*. The *uuxqt* program may be started by either the *uucico* or *uux* programs or a demon specified by a *crontab* entry. The program scans the spool directory for *execute files* (prefix "X."). Each one is checked to see if all the required files are available and if so, the command line is verified and executed.

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

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

5. Uulog—Uucp Log Inquiry

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

The *uulog* program may also be used to request the output of *LOGFILE* entries. The request is specified by the use of the options:

-s*sys* Print entries where *sys* is the remote system name.

-u*user* Print entries for user *user*.

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

6. Uuclean—Uucp Spool Directory Cleanup

This program is typically started by the uucp daily demon. Its function is to remove files from the spool directory that are more than 3 days old. These are usually files for work that can not be completed. The requester of this work is notified that the files have been deleted.

There are several options:

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

-m Send mail to the owner of each file being removed. (Note that most files put into the spool directory will be owned by the owner of the uucp programs since the setuid bit will be set on these programs. This mail is sometimes useful for administration.)

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

-p*pre* Examine files with prefix *pre* for deletion. (Up to 10 of these options may be specified.)

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

7. Security

NOTE

The uucp system, left unrestricted, will let any outside user execute any commands and copy out/in any file that is readable/writable by a uucp login user. It is up to the individual sites to be aware of this and apply the protections that they feel are necessary.

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

- The login for uucp does not get a standard shell. Instead, the *uucico* program is started so that all work is done through *uucico*.
- The owner of the uucp programs should be an administrative login. It should not be one of the logins used for remote system access to uucp.
- A pathcheck is done on file names that are to be sent or received. The *USERFILE* supplies the information for these checks. The *USERFILE* can also be set up to require call-back for certain login-ids. (See the "Files Required For Execution" section for the file description.)
- A conversation sequence count can be set up so that the called system can be more confident-of the caller's identity.
- The *uuxqt* program comes with a list of commands that it will execute. A "PATH" shell statement is prepended to the command line as specified in the *uuxqt* program. The installer may modify the list or remove the restrictions as desired.
- The *L.sys* file should be owned by the uucp administrative login and have mode 0400 to protect the phone numbers and login information for remote sites.
- The programs *uucp, uucico, uux, uuxqt, uulog,* and *uuclean* should be owned by the uucp administrative login, have the setuid bit set, and have only execute permissions.

8. Uucp Installation

It is assumed that the *login name* used by a remote computer to call into a local computer is not the same as the login name of a normal user or the uucp administrative login. However, several remote computers may use the same login name.

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

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

There are several directories used by the uucp system:

lib (/usr/src/cmd/uucp)—This directory contains the uucp system source files.

program (/usr/lib/uucp)—This is the directory used for some of the executable system programs and the system files. Some of the programs reside in "/usr/bin".

spool (/usr/spool/uucp)—This is the uucp system spool directory.

xqtdir (/usr/lib/uucp/.XQTDIR)—This directory is used during execution of the *uux* scripts.

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

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

uucp.h Modification

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

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

makefile Modification

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

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

Compile the System

The command

make install

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

make

will compile the entire system.

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

Files Required For Execution

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

L-devices

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

type line call-unit speed

where

type	is a device type such as ACU or DIR. The field can also be used to specify particular ACUs for some calls by using a suffix on the ACU field, e.g., ACU3. This name should be used in *L.sys*.
line	is the device for the line (e.g., cu10).
call-unit	is the automatic call unit associated with *line* (e.g., cua0). Hardwired lines have a number "0" in this field.
speed	is the line speed.

The line

ACU cu10 cua0 300

would be set up for a system that has device "/dev/cu10" wired to call-unit "/dev/cua0" for use at 300 baud.

L-dialcodes

This file contains the dialcode abbreviations used in the *L.sys* file (e.g., py. mh. boston). The entry format is

abb dial-seq

where

abb	is the abbreviation,
dial-seq	is the dial sequence to call that location.

The line

py 165-

would be set up so that entry py7777 would send 165-7777 to the dial-unit.

USERFILE

This file contains user accessibility information. It specifies four types of constraint:

1. which files can be accessed by a normal user of the local machine,
2. which files can be accessed from a remote computer,
3. which login name is used by a particular remote computer,
4. whether a remote computer should be called back in order to confirm its identity.

Each line in the file has the format

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

where

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

The constraints are implemented as follows:

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

3. When a remote computer logs in, the login name that it uses *must* appear in the *USERFILE*. There may be several lines with the same login name but one of them must either have the name of the remote system or must contain a *null* system name.
4. If the line matched in (3.) contains a "c", the remote machine is called back before any transactions take place.

The line

u,m /usr/xyz

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

The line

dan, /usr/dan

allows the ordinary user *dan* to issue commands for files whose name starts with "/usr/dan". (Note that this type restriction is seldom used.)

The lines

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

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

The lines

root, /
. /usr

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

L.sys

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

system name

The name of the remote system.

time

This is a string that indicates the days-of-week and times-of-day when the system should be called (e.g., MoTuTh0800-1730).

The day portion may be a list containing some of

Su Mo Tu We Th Fr Sa

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

The time should be a range of times (e.g., 0800-1230). If no time portion is specified, any time of day is assumed to be okay for the call. Note that a time range that spans 0000 is permitted, for example, 0800-0600 means all times are ok other than times between 6 and 8 am.

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

device

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

class

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

phone

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

login

The login information is given as a series of fields and subfields in the format

[expect send]...

where *expect* is the string expected to be read and *send* is the string to be sent when the *expect* string is received.

The expect field may be made up of subfields of the form

expect[-send-expect]...

where the *send* is sent if the prior *expect* is *not* successfully read and the *expect* following the *send* is the next expected string. (e.g., login--login will expect *login*; if it gets it, the program will go on to the next field; if it does not get *login*, it will send *null* followed by a new line, then expect *login* again.)

There are two special names available to be sent during the login sequence. The string *EOT* will send an EOT character and the string *BREAK* will try to send a *BREAK* character. (The *BREAK* character is simulated using line speed changes and null characters and may not work on all devices and/or systems.) A number from 1 to 9 may follow the *BREAK* for example, *BREAK1* will send 1 null character instead of the default of 3. Note that *BREAK1* usually works best for 300/1200 baud lines.

A typical entry in the L.sys file would be

sys Any ACU 300 mh7654 login uucp ssword: word

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

9. Administration

This section indicates some events and files that must be administered for the uucp system. Some administration can be accomplished by *shell files* initiated by *crontab* entries. Others will require namual intervention. Some sample *shell files* are given toward the end of this section.

SQFILE—Sequence Check File

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

TM—Temporary Data Files

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

TM.pid.ddd

where *pid* is a process-id and *ddd* is a sequential three digit number starting at zero. After the entire file is received, the *TM* file is moved/copied to the requested destination. If processing is abnormally terminated the file will remain in the spool directory. The leftover files should be periodically removed; the *uuclean* program is useful in this regard. The command

program/uuclean -pTM

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

LOG—Log Entry Files

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

uulog

will accomplish the merge. Options are available to print some or all the log entries after the files are merged. The *LOGFILE* should be removed periodically.

The *LOG.* files are created initially with mode 0222. If the program that creates the file terminates normally, it changes the mode to 0666. Aborted runs may leave the files with mode 0222 and the *uulog* program will not read or remove them. To remove them, either use *rm, uuclean*, or change the mode to 0666 and let *uulog* merge them into the *LOGFILE*.

STST—System Starts Files

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

STST.sys

where *sys* is the remote system name.

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

LCK—Lock Files

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

LCK..str

where *str* is either a device or system name. The files may be left in the spool directory if runs abort (usually only on system crashes). They will be ignored (reused) after 1.5 hours. When runs abort and calls are desired before the time limit, the lock files should be removed.

ERRLOG—uucp System Error File

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

Shell Files

The *uucp* program will spool work and attempt to start the *uucico* program, but *uucico* will not always be able to execute the request immediately. Therefore, the *uucico* program should be periodically started. The command to start *uucico* can be put in a "shell" file with a command to merge *LOG.* files and started by a crontab entry on an hourly basis. The file could contain the commands

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

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

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

program/uuclean -pTM -pC. -pD.
program/uuclean -pST -pLCK -n12

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

A daily or weekly shell should also be created to remove or save old *LOGFILE*s. A shell like

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

can be used.

Login Entry

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

File Modes

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

Section 7

INDEX

A

adding a new system, 3-29
administrative commands, 4-9
applications, 1-3, 6-11
autodialer, 3-3, 3-16, 4-7

B

baud rates, 3-3
Bell Labs Documentation, 6-1

C

cabling connections, 5-1
call UNIX, 2-2
checking UNICOM, 4-3
command dictionary, 2-1
commands 1-7, 2-1
configuration, 1-4
crontab, 3-11, 3-24, 4-4, 4-6
cu, 2-2
cu command over an autodialer, 3-31

D

database files, 4-6, 6-27
debugging the login stream, 3-33
dedicated port, 3-30
DEINSTALL program, 3-28
direct connection, 1-5
errors, 4-3
errors in installation, 3-8
executing commands on remote systems, 2-11
features, 1-2
file removal, 4-4
files modified by uucp-config, 3-25

H

HAYES autodialer, 3-16
how UNICOM works, 1-5, 5-4--6-6
HSI, 3-3
HSI or RS-232-C?, 3-3

I

indirect connection, 1-5
installation 3-1, 3-7, 6-25

installation disk, 3-8
installation tasks, 3-2
installing cu, 3-30
introduction, 1-1

L

L-devices, 4-6, 5-6, 6-27
L-xcmds, 4-5
L.sys, 4-6, 5-6
list of commands, 1-7
local system, 1-5
LOGFILE, 4-4, 6-31
login procedure, 3-1, 3-18
login stream, 3-18
login stream syntax, 3-19

M

mail, 2-5
maintenance, 4-1, 4-4
maintenance duties, 4-1
master, 1-5, 3-4, 3-15
modem, 3-3
modifying the installation, 3-26, 3-7

N

neighbor system, 3-15
network, 1-4, 3-2

O

overview, 1-1

P

poll, 1-6
polling, 1-6,3-24
public files, 2-8

Q

queue, 1-5

R

remote command execution, 2-11
remote system, 1-5
requirements, 1-6
restoring uucp, 4-4
RS-232-C, 3-3

S

security, 4-7
slave, 1-5, 3-4, 3-15
software installation, 3-8
special login stream characters, 3-21
SYSLOG, 4-4
system connections, 3-3
system crash, 3-3
system name, 3-9, 3-4

T

technical notes, 5-1
terminology, 3-1
testing the installation, 3-31
throughput, 1-6
troubleshooting, 3-32

U

UNICOM commands, 2-1
UNIX to UNIX copy, 2-7
user name, 3-1
USERFILE, 4-6, 3-15, 6-28
uucico, 1-7, 3-24, 6-18
uuclean, 3-11, 4-4, 4-10, 6-24
uucp, 2-7, 5-4, 6-16
uucp-config, 3-12
uucp-config flowchart, 3-13
uucp-config questions, 3-13
uucppublic directory, 2-8
uulog, 4-3, 4-11, 6-23
uuname, 2-10
uuphone, 4-7, 4-12
uux, 4-5, 2-11, 5-4, 6-18
uuxqt, 5-5, 6-19

V

VADIC autodialer, 3-18, 3-16

MANUAL CHANGE INFORMATION

At Tektronix, we continually strive to keep up with latest electronic developments by adding circuit and component improvements to our instruments as soon as they are developed and tested.

Sometimes, due to printing and shipping requirements, we can't get these changes immediately into printed manuals. Hence, your manual may contain new change information on following pages.

A single change may affect several sections. Since the change information sheets are carried in the manual until all changes are permanently entered, some duplication may occur. If no such change pages appear following this page, your manual is correct as printed.

Tektronix®
COMMITTED TO EXCELLENCE

MANUAL CHANGE INFORMATION

Date: 11/22/83 Change Reference: C1/1183

Product: 8560: UNICOM Users Manual Manual Part No.: 070-4536-00

DESCRIPTION Product Group 61

CHANGE INSTRUCTIONS

THIS IS A PAGE REPLACEMENT CHANGE.

1. To ensure that information is incorporated in the proper sequence, enter Manual Change Information beginning with the earliest changes.

2. Several editions of this manual may exist. Before entering this change, be sure that replacement page information relates to information in your manual.

3. To implement this change:

   a. Remove old pages: Title page, 2-1 thru 2-4

   b. Insert new pages: Title page, 2-1 thru 2-4

4. You may wish to retain this Manual Change Information sheet at the back of your manual as a record of the change.

Part No. 070-4536-00
Product Group 61

8560 Series
Multi-User Software
Development
Unit

TNIX

UNICOM

Users Manual

This manual supports the following TEKTRONIX product:

8560U05

This manual supports a software module that is compatible with:

TNIX Version 1 (8560)
TNIX Version 2 (8560 Series)

Please check for change information at the rear of this manual

First Printing FEB 1983
Revised DEC 1983

Tektronix®
COMMITTED TO EXCELLENCE

ABOUT WARRANTY AND SUPPORT FOR THIS PRODUCT

This product is provided by Tektronix as category C software.

NOTE

Licensed Software for which the software support is specifed as Category C is furnished without warranty of any kind, and without any representation regarding quality, performance, or suitability.

TEKTRONIX SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Tektronix' liability for damages, if any, whether based upon contract, negligence, strict liability in tort, warranty, or any other basis, shall not exceed the fee paid by the Customer for the Licensed Software.

Category C software is provided on an "as is" basis. Any software services, if available, will be provided at the then current charges.

Section 2

COMMAND DICTIONARY

INTRODUCTION

This section explains the user commands included in the UNICOM software package. (Commands that are used by the system manager for the maintenance of UNICOM are explained in the UNICOM Maintenance section of this manual.) The following commands are explained in this section:

- **cu** (Call UNIX) -- Allows you to access a directly connected system in real-time through your own system. Only one user at a time may use this command through each intersystem link.

- **mail** -- Allows you to send mail to directly and indirectly connected systems. This command functions almost identically to the **mail** command that is provided as a standard TNIX command.

- **uucp** (UNIX to UNIX Copy) -- Allows you to copy files between directly connected systems.

- **uuname** (Name Remote Systems) -- Lists the remote systems connected directly to your system, and the name of your local system.

- **uux** (UNIX to UNIX Execution) -- Allows you to cause other directly connected systems to execute commands.

CU -- CALL UNIX

SYNTAX

cu [-txd] [-e] [-o][-m] [-s speed] [-l port]

DESCRIPTION

Cu calls up another UNIX or TNIX system.

OPTIONS

-t -- used to dial out to a terminal.

-x -- specifies XON-XOF flagging on input from the remote system. Use **-x** when calling a TNIX system. If files sent via **cu** are scrambled, try reversing the setting of this flag and sending the files again.

-d -- specifies DTR flagging on input from the remote system.

-e -- specifies even parity to the remote system.

-o -- specifies odd parity to the remote system.

-m -- specifies mark parity to the remote system.

-s speed -- gives the transmission speed (300, 600, 1200, 2400, 4800, or 9600 baud); The default value is 2400 baud.

-l port -- specifies the communication line device. The default value is:

-l /dev/tty3

Ask your system manager which port (/dev/tty1 -- /dev/tty7) to use for communication with a particular system. There must not be a login on the port being used.

After the **cu** command is used, the local system will indicate a connection to the remote system. You may then log in to the other system as though your terminal were being used directly with that system. You must have a user name and a password for the remote system.

In addition to the standard system commands, you may enter commands that begin with the "~" character and that have the following effects:

~@ (NULL)
Causes a 'break' to be sent to the remote system.

~. Terminate the connection. This does not log you out from the remote system.

~^D Terminate the connection. This does not log you out from the remote system. (^D = CTRL D)

~<file Copy the contents of **file** to the remote system, as though typed at the terminal. Terminated with <LF> character.

~! Invoke an interactive shell on the local system.

~!cmd ...
Execute the command on the local system (via **sh -c**).

~$cmd ...
Execute the command locally and send its output to the remote system. Terminated with <CR,LF> characters. To exit the local system shell enter logout.

~%take fromfile [tofile]
Copy file "fromfile" (on the remote system) to file "tofile" on the local system. If "tofile" is omitted, the "fromfile" name is used both places.

For some remote systems a ^D is needed to end file reception before the system prompt returns. For example:

```
~%take filea fileb ^D
```

The use of **~%take** requires the existence of the **echo** and **tee** commands on the remote system. Also, **stty tabs** mode is required on the remote system if tabs are to be copied without expansion.

~%put fromfile [tofile]
copy file "fromfile" (on the local system) to file "tofile" on remote system. If "tofile" is omitted, the "fromfile" name is used both places.

For some remote systems a ^D is needed to end file transmission before the system prompt returns. For example:

```
~%put filea fileb ^D
```

The use of **~%put** requires the **stty** and **cat** commands on the remote system. It also requires that the current erase and kill characters on the remote system be identical to the current ones on the local system.

~~... Send the line "~..." for execution on the remote system. You may use this command to cause a directly connected remote system to execute the **cu** command. This will permit your local system to use the **cu** command to communicate with indirectly connected systems.

~>[>][:]8560file
0 or more lines of text from the remote system to be written to '8560file'.
~>

This sequence places or appends text directly into a file (8560file) on the local system. If ":" is used, the diversion is **silent,** i.e., it is written only to the file. If ":" is omitted, output is written both to the file and to the standard output. The trailing "~>" terminates the diversion.

DIAGNOSTICS

Exit code is zero for normal exit, nonzero (various values) otherwise.

NOTES

File copy has no error checking.

The actual baud rate of data transmission between your terminal and the remote system is slower than the baud rate of the line. This occurs because the **cu** program looks for special characters in all data sent and received.

The **cu** command may use a port on a master system that is configured for **uucp** use. The conditions for this use:

- The port must use RS-232 protocol.
- You must be logged in as user "uucpa" to use the **cu** command on this port.
- The **uucp** command and the **cu** command will compete for use of the port; therefore do not to allow the two commands to use the port simultaneously.

When calling out from TNIX, the connection must be made prior to setting characteristics for the port, because they are lost when the communication is finished.

TNIX is not able to detect and halt transmission to a full terminal input buffer, so copying long files to a busy TNIX system is not recommended.