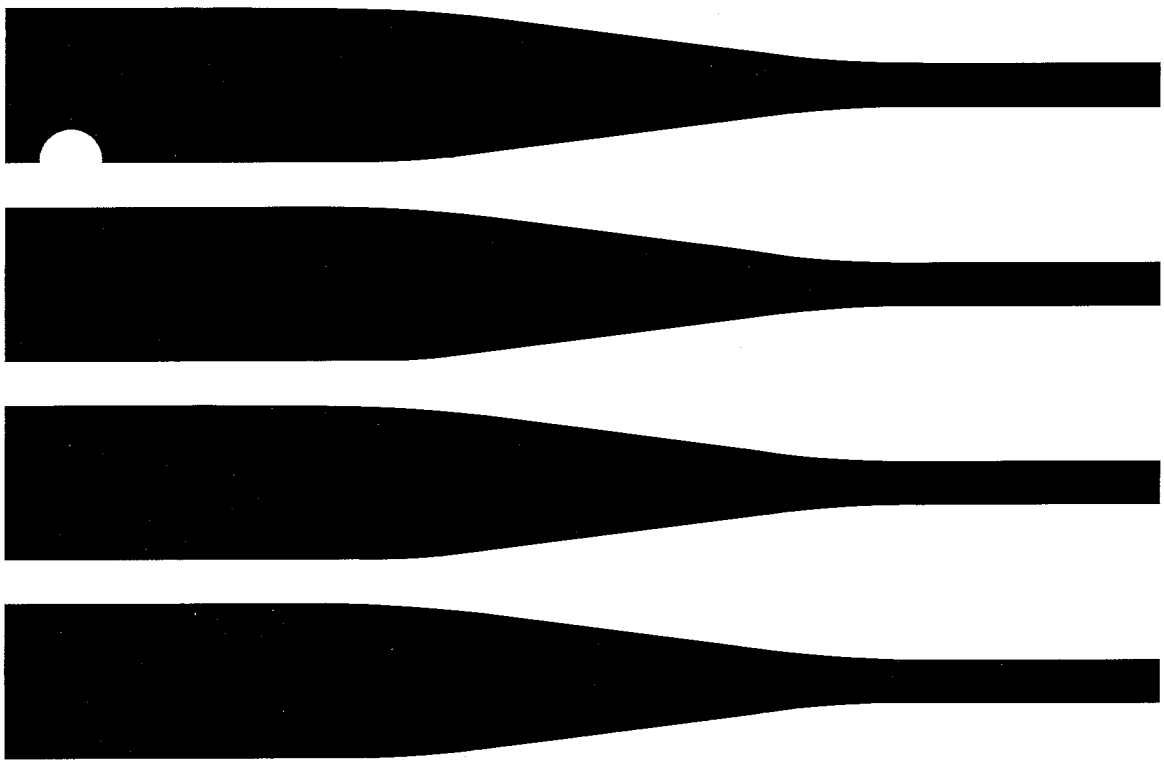
Issue 1
October 1984

AT&T 3B2 Computer
UNIX™ System V Release 2.0
Basic Networking
Utilities Guide

Select Code
305-432

Comcode
403779630



TRADEMARKS

The following is a listing of the trademarks used in this manual:

- Dimension is a registered trademark of AT&T.
- Hayes is a trademark of Hayes Microcomputer Products, Inc.
- Micom is a registered trademark of Micom Systems, Inc.
- Penril is a registered trademark of Penril Corporation, Inc.
- Rixon is a registered trademark of Rixon, Inc.
- Ventel is a registered trademark of Ven-Tel, Inc.
- UNIX is a trademark of AT&T Bell Laboratories.

NOTICE

The information in this document is subject to change without notice. AT&T Technologies assumes no responsibility for any errors that may appear in this document.

NOTE

This Utilities Guide contains descriptive information and UNIX* System manual pages for the commands that are in this optional utilities. You can keep this guide intact, the way it was delivered, or you can remove the pages from the soft cover and file them in the appropriate binders. If you decide to file the contents of this utilities in binders, read the following paragraphs.

Remove the descriptive information from the soft cover, place the provided tab separator in front of the title page, and file this material in a UTILITIES binder. Remove the UNIX System manual pages and file them in either the *3B2 Computer UNIX System V User Reference Manual* binder or with the *System Administration Utilities Guide* manual pages.

The binder in which you will file the manual pages depends on the class or subclass of the command. Notice that each manual page has a number in parenthesis following it. This number identifies the command class or subclass. For example, a command with a (1) after it is a general purpose user command, a command with a (1C) is a communication command, a command with a (1G) is a graphics command, and a command with a (1M) is classified as a system maintenance command.

File the manual pages that have either a (1), (1C), or (1G) after the command in the *User Reference Manual.* File the manual pages that have a (1M), (7), or (8) after the command with the *System Administration Utilities Guide* manual pages. It is recommended that you file the manual pages in alphabetical order.

---

* Trademark of AT&T Bell Laboratories

CONTENTS

Chapter 1

INTRODUCTION

Chapter 1

INTRODUCTION

GENERAL

This guide describes the operation, administration, and command format (syntax) of the Basic Networking Utilities. The majority of the material discussed is intended to be used by a sophisticated user who needs to know how to administer and maintain the utilities. Lower level (Novice) users may find the tutorial information (Chapter 7) helpful in understanding how the various commands are used to communicate with other machines.

GUIDE ORGANIZATION

This guide is structured so you can easily find information without having to read the entire text. The remainder of this guide is organized as follows:

- Chapter 2, "OVERVIEW," contains an overview of the Basic Networking Utilities. It discusses the hardware and software associated with the operation of the networking system.

After the hardware and software are introduced, a description of how the various programs work together to communicate with other machines is presented.

- Chapter 3, "ADMINISTRATION," discusses the administration of the Basic Networking Utilities. The discussion includes the files associated with the everyday operation of the system, supporting data base files, and administrative tasks.

- Chapter 4, "SIMPLE ADMINISTRATION," describes the Simple Administration feature of the Basic Networking Utilities (**uucpmgmt**).

- Chapter 5, "DIRECT LINKS," contains information on establishing a direct link from your AT&T 3B2 Computer to another 3B Computer. This includes some general information as well as the specifics needed to set up the Basic Networking software and establish the direct link.

- Chapter 6, "MAINTENANCE," contains information on the maintenance of the Basic Networking Utilities. This includes some suggestions on debugging and a listing of error messages.

- Chapter 7, "COMMAND DESCRIPTIONS," describes the command format (syntax) of each command in the command set. The descriptions include the purpose of the command, a discussion on the command syntax and options, and examples of using the command.

- The Appendix contains the Manual Pages that are associated with the Basic Networking Utilities.

DEFINITION OF TERMS

There may be terms used in this guide that you are not familiar with. Below is a list of such terms:

local machine	Refers to the machine on the "near" end of a communication link; normally, your 3B2 Computer.
remote machine	Refers to a machine on the "far" end of a communication link; normally, a machine that your 3B2 Computer calls.
active machine	A machine with the Basic Networking Utilities *and* the hardware required to establish communication links (i.e. Auto Dial Modem).
passive machine	A machine that has the Basic Networking Utilities but does *not* have the hardware required to establish communication links.
network	A group of machines that are set up to exchange information and resources.
node	A terminating point (machine) on a network.
UUCP	This term (all caps) is used to indicate a group of programs and files that allow "unix-to-unix copy" capability between UNIX* Systems. In general, it refers to all Basic Networking Utilities with the exception of the **cu** and **ct** programs. If "uucp" is shown in the text with bold type (**uucp**), this is referring specifically to the **uucp** program or login ID.

* Trademark of AT&T Bell Laboratories

Chapter 2

OVERVIEW

PAGE

Chapter 2

OVERVIEW

This chapter contains an overview of the Basic Networking Utilities. The hardware and software of the utilities is introduced prior to discussing how the it operates.

WHAT IS BASIC NETWORKING

The Basic Networking Utilities allow machines using the UNIX Operating System to communicate with one another. The utilities allow you to:

- Transfer files and send electronic mail to other UNIX System machines as background processes
- Interactively communicate with UNIX System machines and possibly non-UNIX System machines
- Execute commands (restrictive) on a remote machine without logging in
- Call a remote terminal and allow the user of that terminal to login on your 3B2 Computer.

The later part of this chapter discusses the variety of ways information is transferred from one machine to another. It also discusses how commands are executed remotely, and how your 3B2 Computer can be requested to call a remote terminal. But first, you should become familiar with the hardware and software associated with the Basic Networking Utilities.

WHAT KIND OF HARDWARE IS NEEDED

Before your 3B2 Computer can communicate with a remote machine, a communication link must be established to the remote machine. There are three types of hardware used to establish a communication link to another machine. The first is a direct link from a serial port on the 3B2 Computer to a serial port on the other machine. This type of connection is useful when two machines communicate with each other on a regular basis. Direct links allow data to be transferred at rates as high as 19200 bits per second (bps). Even though the RS-232 standard recommends that direct links be limited to 50 feet or less, two machines may be separated by several hundred feet provided that noise on the direct link does not become a problem. If noise becomes a problem or greater distance is needed between the two machines, the transfer rate may need to be decreased or limited distance modems placed at each end of the connection.

The Basic Networking Utilities does not contain the hardware needed to link your 3B2 Computer directly to remote machines. However, information on establishing a direct link to another machine and the parts needed can be found in Chapter 5, "DIRECT LINKS."

The second type of communication link uses the telephone network. In this type of link, the machine that establishes the connection (local machine) must have an automatic call unit (ACU). The ACU dials the specified phone number upon request from the Basic Networking Utilities. The called (remote) machine must have a telephone modem capable of answering incoming calls so other machines can contact it through the telephone network. If you wish to use the telephone network for establishing communication links, contact your AT&T Account Representative or Authorized Dealer for information on the AT&T Automatic Dial Modem.

The third type of communication link is established through a Local Area Network (LAN). In this case, the 3B2 Computer must be a node on a LAN switch. This will allow the 3B2 Computer to establish a link to any machine connected to that LAN.

If a machine can establish a link to and request communication with another machine, it is considered an "active machine." Active machines must be able to establish links using one of the three types of hardware mentioned above. A "passive machine" cannot establish a link to (call) a remote machine. However, a connection can be established *to* a passive computer if the passive computer has:

- A telephone modem that can automatically answer a call, or
- A direct link dedicated to serving incoming calls.

Note: If a machine is connected to a LAN, it is considered to be a active machine since it can call other machines on the LAN.

When a passive machine is called by an active machine, this is referred to as "polling." Polling passive machines is discussed later in more detail.

THE BASIC NETWORKING SOFTWARE

The Basic Networking Utilities is composed of software programs, daemons (background routines), and a supporting data base. The supporting data base contains support files that store information such as telephone numbers, location of the devices (hardware) used to establish links, security restrictions, etc. The software programs and a skeleton data base is supplied on the Basic Networking Utilities floppy diskette that comes with this guide. Since each 3B2 Computer will have a unique supporting data base for Basic Networking, the files that make up this data base are empty (except for comments) when they are loaded onto the hard disk. The Basic Networking Simple Administration feature is used to create unique entries in some of these files. The Simple Administration subcommands are discussed in Chapter 4, "SIMPLE ADMINISTRATION."

The Directories and Their Purpose

There are several directories that contain the programs and support files of Basic Networking. Some of these directories are unique to Basic Networking, while others are common to the UNIX Operating System and the 3B2 Computer. The directories used by Basic Networking are listed below:

/usr/bin	This directory is used by the UNIX Operating System to store executable programs and is used by Basic Networking for the same purpose.
/usr/admin/menu	This directory contains the Simple Administration subcommands for certain utilities. The subcommands for the Basic Networking Utilities are in the directory **/usr/admin/menu/packagemgmt/uucpmgmt**.
/usr/lib/uucp	This directory is the "HOME" directory for the **uucp** administrative login. It contains the files of the supporting data base and some executable programs.

/usr/spool/locks	This directory contains the lock (LCK) files for the Basic Networking hardware devices. Lock files are discussed in Chapter 3.
/usr/spool/uucp	This directory is the "spool directory" for "work" that is to be processed by the Basic Networking Utilities. It contains a tree-like structure of subdirectories associated with remote machines that your 3B2 Computer wishes to communicate with (has communicated with recently). These subdirectories are also used for administrative purposes such as storing log and status information.
/usr/spool/uucppublic	This directory is the "public" directory for UUCP transfers. The public directory is used to store files that have been sent to your 3B2 Computer. Some remote machines may be restricted to placing files in this directory, while others may have permission to place files elsewhere.

The Software Programs and Their Purpose

There are several types of programs associated with the Basic Networking Utilities. Some of these programs are used by normal users to transfer data and obtain status information, while others are used for administrative purposes, or are executed internally. The following paragraphs contain a brief description of the programs and their purpose.

User Programs

cu: Connects your 3B2 Computer to a remote machine and allows you to be logged in on both machines at the same time. This allows you to transfer files or execute commands on either machine without dropping the link.

ct: Connects your 3B2 Computer to a remote terminal and allows the user of the remote terminal to login. The user of a remote

terminal may call into the 3B2 Computer and request that the 3B2 Computer call the remote terminal back. In this case, the 3B2 Computer drops the initial link so that the modem will be available when it is called back.

uucp: Performs all of the preliminary work in queuing file transfers to remote machines. It creates "work" files which contain the instructions for transferring the queued file(s). Depending on the options specified, it may make a copy of the file to be transferred in the spool directory. These files are called "data" files. Once the "work" and "data" files have been created, **uucp** calls the **uucico** daemon which in turn attempts to contact the remote machine.

uuto: This program works very similar to the **uucp** program. In fact, it calls the **uucp** program to create "work" and "data" files. The main difference between **uuto** and **uucp** is the way the transferred files are placed on the remote machine. With **uucp**, you can specify a pathname on the remote machine where you want the files to be placed. With **uuto**, all transferred files are placed in the **uucppublic** directory under **/usr/spool/uucppublic/receive** (refer to the **uuto** Manual Page, located in the Appendix).

uupick: When files are transferred to a machine using **uuto**, **uupick** can be used to retrieve the files placed under **/usr/spool/uucppublic/receive**.

uux: This program creates "work" files, "data" files, and "execute" files for executing commands on a remote machine. The "work" file contains the same information as those created by **uucp** and **uuto**. The "execute" files contain the command string to be executed on the remote machine and a list of the "data" files. The "data" files are those files required for the command execution.

uustat: This program displays status information for requested transfers (**uucp**, **uuto**, or **uux**). It also provides you with a means of controlling queued transfers.

Administrative Programs

uulog: This program displays the contents of a specified machine's log file. Individual log files are created for each remote machine your 3B2 Computer communicates with using the **uucp**, **uuto**, and **uux** programs.

uucleanup: This program has several functions that are all associated with the cleanup of the spool directory. It is normally executed out of a shell script called **uudemon.cleanu** which is started by **cron**.

Uutry: This program is a shell script that is used to test call processing capabilities with a moderate amount of debugging. It invokes the **uucico** daemon to establish the communication link between your 3B2 Computer and the specified machine.

uucheck: This program checks for the presence of Basic Networking directories, programs, and support files. It is also capable of checking certain parts of the **Permissions** file.

Internal Programs

uugetty: This program is very similar to the **getty** program except it permits a line (port) to be used in both directions. **Uugetty** allows users to login on your 3B2 Computer and, if the line is not in use, it will allow **uucico**, **cu**, or **ct** to use it for dialing out. If one of these programs attempts to dial out when the line is busy, **uugetty** will deny the requester permission and echo a message indicating that the device is unavailable. **Uugetty** is executed as a function of the **init** program.

The UUCP Daemons and Their Purpose

There are three daemons that are part of the Basic Networking Utilities. These daemons are routines that run as background processes to handle file transfers and command executions.

uucico: This daemon is referred to as the transport program for UUCP requests. It selects the device used for the link, establishes the link to the remote machine, performs the required login sequence, performs permission checks, transfers "data" and "execute" files, logs results, and notifies specified users of transfer completions via **mail**. When the local **uucico** daemon calls a remote machine, it "talks" to the **uucico** daemon on the remote machine during the session. The **uucico** daemon is executed by several methods. It is started by the **uucp**, **uuto**, and **uux** programs to contact the remote machine after all the required "data", "work", and/or "execute" files have been created. It is also started by the **uusched** and **Uutry** programs.

uuxqt: This daemon is the execution program for remote execution requests. It searches the spool directory for "execute" files (X.) that have been sent from a remote machine. When an X. file is found, **uuxqt** opens it to get the list of data files that are required for the execution. It then checks to see if the required data files are available and accessible. If the files are present and can be accessed, **uuxqt** checks the **Permissions** file to verify that it has permission to execute the requested command. The **uuxqt** daemon is executed out of the **uudemon.hour** shell script which is started by **cron**.

uusched: This daemon schedules the queued work in the spool directory. Before starting the **uucico** daemon, **uusched** randomizes the order in which remote machines will be called. **Uusched** is executed out of a shell script called **uudemon.hour** which is started by **cron**.

The Supporting Data Base Files and Their Purpose

As mentioned earlier, several of the Basic Networking programs require information contained in support files. These support files are located in the **/usr/lib/uucp** directory. The **cu**, **ct**, **uucico**, and **uuxqt** programs require supporting information from the following files:

Devices This file contains information concerning the location and line speed of the automatic call unit, direct links, and possibly network devices.

Dialers This file contains character strings required to negotiate with network devices (automatic calling device) in the establishment of connections to remote computers (non 801-type dialers).

Systems This file contains information needed by the **uucico** daemon (and possibly the **cu** program) to establish a link to a remote machine. It contains information such as the name of the remote machine, the name of the connecting device associated with the remote machine, when the machine can be reached, telephone number, login id, password, etc.

Dialcodes This file contains dial-code abbreviations that may be used in the phone number field of **Systems** file entries.

Permissions This file defines the level of access that is granted to machines when they attempt to transfer files or remotely execute commands on your 3B2 Computer.

There are several other files that may be considered part of the supporting data base, but these files are not directly related to the process of establishing a link and transferring files. For this reason, discussion of these files is reserved for Chapter 3, "ADMINISTRATION."

HOW BASIC NETWORKING OPERATES

This section briefly describes the operation of the Basic Networking Utilities. There are five programs that allow your 3B2 Computer to communicate with remote machines. The following paragraphs briefly describe what happens when you execute these programs.

ct - Connect a Terminal

The **ct** program instructs your 3B2 Computer to initiate a call to a remote terminal and issue a **getty** to that remote terminal. The command line of the **ct** command must contain the telephone number of the remote terminal. Of course, the remote terminal must be attached to a modem that will automatically answer the call.

When the **ct** command line is issued, the **ct** program will search for an automatic dialer in the **Devices** file with a transfer rate that matches what was specified in the command line. If no transfer rate was specified, it defaults to 1200 bps. When **ct** finds the dialer to be used, it attempts to dial the phone number specified in the command line. If no dialer is available, **ct** asks if it should wait for an available dialer and if so, how many minutes should it wait. An option is available to override this dialogue.

When the modem at the remote terminal answers the call from your 3B2 Computer, it is issued a **getty** (login) process. At this point, the user at the remote terminal may attempt to login.

The user at a remote terminal may call your 3B2 Computer, login, and request that the 3B2 Computer call the remote terminal back using the **ct** command. If this scenario is used, the remote user issues a **ct** command and the link from the remote terminal is dropped. After **ct** finds an available dialer in the **Devices** file, it will call the remote terminal back.

cu - Call a UNIX System

The **cu** command enables you to call another machine and login as a remote user. The telephone number or node name of the remote machine is required in the command line. If the phone number is specified, it is passed on to the automatic dial modem. If a system name is specified, the phone number is obtained from the associated **Systems** file entry. If an automatic dial modem is not used to establish the connection, the line (port) associated with the direct link to the remote machine can be specified in the command line.

If an automatic dial modem is used, the **cu** program will search for an automatic dialer in the **Devices** file with a transfer rate that matches what was specified in the command line. If no speed is specified, the first dialer listed (if available) is used regardless of its transfer rate. After the link is established and you have successfully completed the login process, you will be logged in on both computers. This will allow you to execute commands on either computer and/or transfer ASCII coded files from one computer to another. After you terminate the connection, you will still be logged in on your 3B2 Computer (calling computer). This command can only be executed by an active computer.

uucp - UNIX-to-UNIX System Copy

The **uucp** command will allow you to transfer file(s) to a remote computer without knowing any details of the connection. All that you are required to know is the name of the remote computer and possibly the login ID of the remote user to whom the file(s) is being sent. The details of the connection are kept in the **Systems** file.

When you enter a **uucp** command, the **uucp** program creates a "work" file and possibly a "data" file for the requested transfer. The "work" file contains information required for transferring the file(s). The "data" file is simply a copy of the specified source file. After these files are created in the spool directory, the **uucico** daemon is started.

The **uucico** daemon attempts to establish a connection to the remote machine that is to receive the file(s). It first gathers the

information required for establishing a link to the remote machine from the **Systems** file. This is how **uucico** knows what type of device to use in establishing the link. Then, **uucico** searches the **Devices** file looking for the devices that match the requirements listed in the **Systems** file. After **uucico** finds an available device, it attempts to establish the link and login on the remote machine.

When **uucico** logs in on the remote machine, it starts the **uucico** daemon on the remote machine. The two **uucico** daemons then negotiate the line protocol to be used in the file transfer(s). The local **uucico** daemon then transfers the file(s) to the remote machine and the remote **uucico** places the file in the specified path name(s) on the remote machine. After your 3B2 Computer completes the transfer(s), the remote machine may send files that are queued for your 3B2 Computer. The remote machine can be denied permission to transfer these files with an entry in the **Permissions** file. If this is done, the remote machine must establish a link to your 3B2 Computer to perform the transfers.

If the remote machine or the device selected to make the connection to the remote machine is unavailable, the request remains queued in the spool directory. Each hour (default), **uudemon.hour** is started by **cron** which in turn starts the **uusched** daemon. When the **uusched** daemon starts, it searches the spool directory for the remaining "work" files, generates the random order in which these requests are to be processed, and then starts the transfer process (**uucico**) described in the previous paragraphs.

The transfer process described generally applies to an active machine. An active machine (one with calling hardware and Basic Networking software) can be set up to "poll" a passive machine. A passive machine can queue file transfers (because it has Basic Networking software) but, it cannot call the remote machine because it does not have the required hardware. The **Poll** file (**/usr/lib/uucp/Poll**) contains a list of machines that are to be polled in this manner. For additional information, refer to the discussion on the **Poll** file and **uudemon.poll** in Chapter 3, "ADMINISTRATION."

uuto - Public UNIX-to-UNIX System Copy

The **uuto** program uses the **uucp** program to build "work" files and "data" files in the spool directory for requested transfers. The difference is that the **uuto** command will not allow you to specify a path name as a destination for the file. The **uuto** command automatically puts the file in a directory under **/usr/spool/uucppublic/receive**. Once the transfer is complete, mail is sent to the appropriate user indicating that a file has arrived and was placed in the public area. That user can then use the **uupick** command to retrieve that file. The **uupick** command will search the public area for files destined to the user and allow the user to interactively delete, print, or move the file to a named directory.

uux - UNIX-to-UNIX System Execution

The **uux** command allows commands to be executed on a remote machine. It gathers files from various computers and executes the specified command on these files and sends the standard output to a file on the specified computer. This can be useful when some of the required resources (commands and/or files) are not present on your 3B2 Computer. Remote mail is implemented using the **uux** program, but its execution is embedded in the standard **mail** command. For security reason, many machines will limit the list of commands that can be executed via **uux** to the default, receipt of mail.

When the **uux** command is issued, the **uux** program creates an "execute" (X.) file which contains the names of the files required for execution, your login name, the destination of the standard output, and the command to be executed. **Uux** also creates "work" (C.) files that are used to gather the files required for execution. These files are then sent to the remote machine, along with the "execute" file, by the **uucico** daemon and placed in the remote spool directory.

Periodically, the **uuxqt** daemon on the remote machine is started to search for X. files in the spool directory. Upon finding an X. file, the **uuxqt** daemon checks to see if all the required data files are available and accessible. It then checks the **Permissions** file to verify that the command(s) listed can be performed. After execution, **uuxqt** sends the standard output to a file on the specified computer.

Chapter 3

ADMINISTRATION

PAGE

Chapter 3

ADMINISTRATION

This chapter discusses the files and tasks associated with the operation of the Basic Networking Utilities. The amount of effort required to administer the Basic Networking Utilities depends on the amount of "traffic" that enters or leaves your 3B2 Computer. For an average computer, little if any intervention with the automatic cleanup functions is required. A computer with a large amount of traffic may require more attention as problems arise.

By now, you have probably realized that the "UUCP facilities" make up the bulk of the Basic Networking Utilities. The UUCP facilities could generally be defined as all of the programs and support files in Basic Networking with the exception of the **ct** and **cu** programs.

ADMINISTRATIVE FILES

TM - temporary data file

These data files are created under the spool directory (i.e. /usr/spool/uucp/XXXX) when receiving a file from another machine. The directory "XXXX" has the same name as the remote machine that is sending the file. The temporary data file names have the format:

TM.*pid.ddd*

Where:

pid	is a process-id,
ddd	is a sequential three digit number starting at zero.

After the entire file is received, the **TM.** file is moved to the path name specified in the command line. If the file was sent via the **uuto** program, the file is automatically moved to the public area. If processing is abnormally terminated, the **TM.** file may remain in the "XXXX" directory. These files should be periodically removed.

LCK - lock file

Lock files are created in the **/usr/spool/locks** directory for each device in use. Lock files prevent duplicate conversations and multiple attempts to use the same calling device. The file name has the format

LCK.*str*

where *str* is either a device or computer name. The files may be left in the spool directory if runs abort (usually on computer crashes). The lock files will be ignored (reused) after the parent process is no longer active.

Work (C.) File

Work files are created in a spool directory when work (transfers or remote command executions) has been queued for a remote computer. Their names have the format:

C. *sysnxxxx*

where *sys* is the name of the remote computer, *n* is the ASCII character representing the grade (priority) of the work, and *xxxx* is the four digit job sequence number assigned by UUCP.

Work files contain the following information:

- Full path name of the file to be sent or requested
- Full path name of the destination or ~user/file name
- User login name
- List of options
- Name of associated data file in the spool directory. If the **-c** or **-p** option was specified, a dummy name (D.0) is used
- Mode bits of the source file
- Remote user's login name to be notified upon completion of the transfer.

Data (D.) File

Data files are created when it is specified in the command line to copy the source file to the spool directory. Their names have the following format:

D. *sysnxxxx*

where *sys* is the name of the remote computer, *n* is the character representing the grade (priority) of the work, and *xxxx* is the four-digit job sequence number assigned by **uucp**. The four digit job sequence number may be followed by a subjob number which is used when there are several **D.** files created for a work (**C.**) file.

Execute (X.) File

Execute files are created in the spool directory prior to remote command executions. Their names have the following format:

X. *sysnxxxx*

where *sys* is the name of the remote computer, *n* is the character representing the grade (priority) of the work, and *xxxx* is the four digit sequence number assigned by UUCP.

Execute files contain the following information:

- Requester's login and computer name
- Name of file(s) required for execution
- Input to be used as the standard input to the command string
- Computer and file name to receive standard output from the command execution

- Command string
- Option lines for return status requests.

Machine Log Files

Log files are created for each remote machine with which your 3B2 Computer communicates. Each machine may have four log files, one for **uucico**, **uuxqt uux**, and/or **uucp** requests depending on the type of communication that has taken place. The log files are kept in the directory **/usr/spool/uucp/.Log**. Each day, these log files are combined and stored in the directory **/usr/spool/uucp/.Old** when **uudemon.cleanu** is executed. The combined files are kept three days before they are removed. If space is a problem, the administrator may consider reducing the number of days the files are kept by modifying the **uudemon.cleanu** shell file.

SUPPORTING DATA BASE

The data base that supports the Basic Networking Utilities is composed of several support files. These support files contain information required by the **uucico** and **uuxqt** daemons during file transfers or remote command executions. All of the support files are located in the **/usr/lib/uucp** directory.

Devices File

The **Devices** file (**/usr/lib/uucp/Devices**) contains the information for all of the devices that may be used to establish a link to a remote machine. It contains information for both automatic call units, direct links, and network connections. Although provisions are made for several types of devices, only the AT&T Automatic Dial Modem and Direct Links are supported by AT&T. The management of this file is supported by the Simple Administration subcommand **devicemgmt**.

This file works very closely with the **Dialers**, **Systems**, and **Dialcodes** files. It may be beneficial to become vaguely familiar with these files before attempting to gain an understanding of the **Devices** file.

Each entry in the **Devices** file has the following format:

Type Line Line2 Class Dialer-Token-Pairs

where each field (separated by a space) is defined in the following paragraphs.

Type: This field may contain one of four keywords:

Direct	This keyword indicates a Direct Link to another computer (for **cu** connections only).
ACU	This keyword indicates that the link to a remote computer is made through an automatic call unit (Automatic Dial Modem). This modem may be connected either directly to the 3B2 Computer or indirectly through a Local Area Network (LAN) switch.
Network	This keyword indicates that the link is established through a LAN switch where *Network* is replaced with either **micom** or **develcon**. These two LAN switches are the only ones that contain caller scripts in the **Dialers** file (discussed a little later). Other switches may be used if caller scripts are constructed and placed in the **Dialers** file.
System-Name	This keyword indicates a direct link to a particular machine where *System-Name* is replaced by the name of the particular computer. This naming scheme is used to convey the fact that the line associated with this **Devices** entry is for a particular machine.

The keyword used in the Type field is matched against the third field of **Systems** file entries as shown below:

Devices: **ACU** contty - 1200 penril

Systems: eagle Any **ACU** 1200 3-2-5-1 ogin: nuucp ssword: Oakgrass

Line: This field contains the device name of the line (port) associated with the **Devices** entry. For instance, if the Automatic Dial Modem for a particular entry was attached to the /dev/contty line, the device name would be contty.

Line2: If the ACU keyword was used in the Type field and the ACU is an 801 type dialer, this field would contain the device name of the 801 dialer. It should be noted that 801 type ACUs do not contain a modem. Therefore, a separate modem is required and would be connected to a different line (defined in the Line field). This means that one line would be allocated to the modem and another to the dialer. Since the 3B2 Computer will not normally use this type of configuration, this field is ignored, but must contain a pseudo entry as a placeholder (use a "-" as a placeholder).

Class: If an ACU keyword is used, this may be just the speed of the device. It may contain a letter and speed (e.g. C1200, D1200) to differentiate between classes of dialers (Centrex or Dimension PBX). This is necessary because many larger offices may have more than one type of telephone network. One network may be dedicated to serving only internal office communications while the other handles the external communications. Therefore, it is necessary to distinguish which line(s) should be used for internal communications and which should be used for external communications. The same distinction must be made in the **Systems** file because a match is made against the fourth field of **Systems** file entries as shown below:

Devices: ACU contty - **D1200** penril

Systems: eagle Any ACU **D1200** 3-2-5-1 ogin: nuucp ssword: Oakgrass

Some devices can be used at any speed, so the keyword "Any" may be used in the Class field. If "Any" is used, the line will match any

speed requested in a **Systems** entry. If this field is "Any" and the **Systems** Class field is "Any", the speed defaults to 1200 bps.

Dialer-Token-Pairs: This field contains pairs of dialers and tokens. The "dialer" portion may be an automatic dial modem, LAN switch, or "direct" for Direct Link devices. The "token" portion may be supplied immediately following the "dialer" or if not present, it is taken from the **Systems** file. This field has the format:

dialer-token dialer-token

where the last pair may or may not be present, depending on the associated device (dialer). In most cases, the last pair will contain only a "dialer" and the "token" is retrieved from the Phone field of the **Systems** entry.

The Dialer-Token-Pairs (DTP) field may be structured four different ways, depending on the device associated with the entry:

1. If a direct link is established to a particular computer, the DTP field of the associated entry would contain the keyword "direct." This is true for both types of direct link entries, Direct and System-Name (refer to discussion on the Type field).

2. If an automatic dialing modem is connected directly to a 3B2 Computer port, the DTP field of the associated **Devices** entry will only have one pair. This pair would normally be the name of the modem. This name is used to match the particular **Devices** entry with an entry in the **Dialers** file. Therefore, this "dialer" must match the first field of a **Dialers** file entry as shown below:

   Devices: ACU contty - 1200 **ventel**

   Dialers: **ventel** =&-% "" \r\p\r\c $ <K\T%%\r>\c ONLINE!

   Notice that only the "dialer" (**ventel**) is present in the DTP field of the **Devices** entry. This means that the "token" to be passed on to the dialer (in this case the phone number) is taken from the Phone field of a **Systems** file entry.

3. If an automatic dialing modem is connected to a LAN, the 3B2 Computer must first access the switch and the switch will make the connection to the automatic dialing modem. This type of entry would have two pairs. The "dialer" portion of each pair (fifth and seventh fields of entry) is used to match entries in the **Dialers** file as shown below:

   Devices: ACU tty14 - 1200 **develcon** vent **ventel**

   Dialers: **ventel** =&-% "" \r\p\r\c $ <K\T%%\r>\c ONLINE!
   Dialers: **develcon** "" "" \pr\ps\c est:\007 \E\D\e \007

   In the first pair, develcon is the "dialer" and vent is the "token" that is passed to the Develcon switch to tell it which device (ventel modem) to connect to the 3B2 Computer. This token would be unique for each LAN switch since each switch may be set up differently. Once the ventel modem has been connected, the second pair is accessed where ventel is the "dialer" and the "token" is retrieved from the **Systems** file.

4. If a machine to which you wish to communicate is on the same local network switch as your 3B2 Computer, the 3B2 Computer must first access the switch and the switch can make the connection to the other machine. In this type of entry, there is only one pair. The "dialer" portion is used to match a **Dialers** entry as shown below:

   Devices: develcon tty13 - 1200 **develcon** \D

   Dialers: **develcon** "" "" \pr\ps\c est:\007 \E\D\e \007

   As shown, the "token" is left blank which indicates that it is retrieved from the **Systems** file. The **Systems** file entry for this particular machine will contain the token in the Phone field which is normally reserved for the phone number of the machine (refer to "Systems File", Phone field). This type of DTP contains an escape character (\D) which ensures that the contents of the Phone field will not be interpreted as a valid entry in the **Dialcodes** file.

There are two escape characters that may appear at the end of a DTP field:

\T Indicates that the Phone (token) field should be translated using the **Dialcodes** file. This escape character is normally placed in the **Dialers** file for each caller script associated with an automatic dial modem (penril, ventel, etc.). Therefore, the translation will not take place until the caller script is accessed.

\D Indicates that the Phone (token) field should not be translated using the **Dialcodes** file. If no escape character is specified at the end of a **Devices** entry, the \D is assumed (default). A \D is also used in the **Dialers** file with entries associated with network switches (develcon and micom).

Dialers File

The **Dialers** file (**/usr/lib/uucp/Dialers**) is used to specify the initial handshaking that must take place on a line before it can be made available for transferring data. This initial handshaking is usually a sequence of ASCII strings that are transmitted and expected, and is often used to dial a phone number using an ASCII dialer (such as the Automatic Dial Modem). As shown in the above examples, the fifth field in a **Devices** file entry is used as an index into the **Dialers** file. Here an attempt is made to match the **Devices** field with the first field of each **Dialers** entry. In addition, each odd numbered **Devices** field starting with the seventh position is used as an index into the **Dialers** file. The management of this file is not supported by Simple Administration. Therefore, changes must be made using one of the editors (**ed**, **vi**).

If the match succeeds, the **Dialers** entry is interpreted to perform the dialer negotiations. The first field matches the fifth and additional odd numbered fields in the **Devices** file. The second field is used as a translate string: the first of each pair of characters is mapped to the second character in the pair. This is usually used to translate = and - into whatever the dialer requires for "wait for dialtone" and "pause."

The remaining fields are "expect-send" strings. Below are some character strings distributed with the Basic Networking Utilities in the **Dialers** file.

```
penril =W-P "" \d > s\p9\c )-W\p\r\ds\p9\c-) y\c : \E\TP > 9\c OK
ventel =&-% "" \r\p\r\c $ <K\T%%\r>\c ONLINE!
hayes =,-, "" \dAT\r\c OK\r \EATDT\T\r\c CONNECT
rixon =&-% "" \d\r\r\c $ s9\c )-W\r\ds9\c-) s\c : \T\r\c $ 9\c LINE
vadiac =K-K "" \005\p *-\005\p-*\005\p-* D\p BER? \E\T\e \r\c LINE
develcon "" "" \pr\ps\c est:\007 \E\D\e \007
micom "" "" \s\c NAME? \D\r\c GO
direct
```

The meaning of some of the escape characters (those beginning with "\") used in the **Dialers** file are listed below:

\p	pause (approximately ¼ to ½ second)
\d	delay (approximately 2 seconds)
\D	phone number or token without **Dialcodes** translation
\T	phone number or token with **Dialcodes** translation
\K	insert a BREAK
\E	enable echo checking (for slow devices)
\e	disable echo checking
\r	carriage return
\c	no new-line
\n	send new-line
\nnn	send octal number.

Additional escape characters that may be used are listed in the section discussing the **Systems** file.

The penril entry in the **Dialers** file is executed as follows. First, the phone number argument is translated, replacing any "=" with a (pause). The handshake given by the remainder of the line works as follows:

""	Wait for nothing.
\d	Delay for 2 seconds.
>	Wait for a ">."
s\p9\c	Send an "s", pause for ½ second, send a 9, send no terminating new-line
)-W\p\r\ds\p9\c-)	Wait for a ")." If it is not received, process the string between the "-" characters as follows. Send a "W", pause, send a carriage-return, delay, send an "s", send a "9", without a new-line, and then wait for the ")."
y\c	Send a "y."
:	Wait for a ":."
\E\TP	Enable echo checking. (From this point on, whenever a character is transmitted, it will wait for the character to be received before doing anything else.) Then, send the phone number followed by a pause character (P). The \T means take the phone number passed as an argument and apply the **Dialcodes** translation and the modem function translation specified by field number 2 of this entry.
>	Wait for a ">."
9\c	Send a "9" without a new-line.
OK	Waiting for the string "OK."

Systems File

The **Systems** file (**/usr/lib/uucp/Systems**) contains the information needed by the **uucico** daemon to establish a communication link to a remote machine. Each entry in the file represents a machine that can be called by the 3B2 Computer. In addition, Basic Networking can be configured to prevent any machine that does not appear in this file from logging in on your 3B2 Computer (refer to "**remote.unknown**"). More than one entry may be present for a particular machine. The additional entries represent alternate communication paths that will be tried in sequential order. The management of this file is supported by the Simple Administration subcommand **systemmgmt**.

Each entry in the **Systems** file has the following format:

System-Name Time Type Class Phone Login

where each field is defined in the following paragraphs.

System-name: This field contains the node name of the remote machine.

Time: This field is a string that indicates the day-of-week and time-of-day when the remote machine can be called. The day portion may be a list containing some of the following:

Su Mo Tu We Th Fr Sa

Wk for any week-day

Any for any day

Never for a passive arrangement with the remote machine. In this case, the 3B2 Computer will never initiate a call to the remote machine. The call must be initiated by the remote machine. The 3B2 computer is in a passive mode in respect to the remote machine. (See discussion of **Permissions** file.)

The time should be a range of times such as 0800-1230. If no time portion is specified, any time of day is assumed to be allowed for the call. Note that a time range that spans 0000 is permitted. For example, 0800-0600 means all times are allowed other that times between 6 a.m. and 8 a.m. An optional subfields available to specify the minimum time (in minutes) before a retry, following a failed attempt. The subfield separator is a semicolon (;). For example, Any;9 is interpreted as call any time; but, wait at least 9 minutes before retrying after a failure occurs.

Type: This field contains the device type that should be used to establish the communication link to the remote machine. The **Devices** file is searched for the device type listed and the device found is used to establish the connection (if available). The following keywords may appear in this field:

ACU	This keyword indicates that the link to a remote computer is made through an automatic call unit (automatic dial modem). This modem may be connected either directly to the 3B2 Computer or indirectly through a Local Area Network (LAN) switch.
Network	This keyword indicates that the link is established through a LAN switch where *Network* is replaced with either **micom** or **develcon**. These two switches are the only ones that contain caller scripts in the **Dialers** file (discussed a little later). Other switches may be used if caller scripts are constructed and placed in the **Dialers** file.
System-Name	This keyword indicates a direct link to a particular machine where *System-Name* is replaced by the name of the particular computer (should be same as field one).

The keyword used in this field is matched against the first field of **Devices** file entries as shown below:

Systems: eagle Any **ACU** D1200 3-2-5-1 ogin: nuucp ssword: Oakgrass

Devices: **ACU** contty - D1200 penril

Class: This field is used to indicate the transfer speed of the device used in establishing the communication link. It may contain a letter and speed (e.g. C1200, D1200) to differentiate between classes of dialers (refer to the discussion on the "Devices File," Class field). Some devices can be used at any speed, so the keyword "Any" may be used. This field must match the Class field in the associated **Devices** entry as shown below:

Systems: eagle Any ACU **D1200** 3-2-5-1 ogin: nuucp ssword: Oakgrass

Devices: ACU contty - **D1200** penril

Phone: This field is used to provide the phone number (token) of the remote machine for automatic dialers (LAN switches). The phone number is made up of an optional alphabetic abbreviation and a numeric part. The abbreviation must be one that is listed in the **Dialcodes** file. In this string, an equal sign (=) tells the ACU to wait for a secondary dial tone before dialing the remaining digits. A dash in the string (-) instructs the ACU to pause four seconds before dialing the next digit.

If your 3B2 Computer is connected to a LAN switch, you may access other machines that are connected to that switch. The **Systems** entries for these machines will not have a phone number in the Phone field. Instead, this field will contain the "token" that must be passed on to the switch so it will know which machine the 3B2 Computer wishes to communicate with. The associated **Devices** entry should have a \D at the end of the entry to ensure that this field is not translated using the **Dialcodes** file.

Login: This field contains the login information given as a series of fields and subfields of the format:

expect send

where expect is the string that is received and send is the string that is 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. For example, with "login--login", UUCP will expect "login." If UUCP gets "login," it will go on to the next field. If it does not get login, it will send nothing followed by a new line, then look for login again. If no characters are initially expected from the remote machine, the characters "" (null string) should be used in the first expect field. Note that all send fields will be sent followed by a new-line unless the send string is terminated with a \c.

There are several escape character that cause specific actions when they are a part of a string sent during the login sequence. The following escape characters are useful in UUCP communications:

\N	Send a null character.
\b	Send a backspace character.
\c	If at the end of a string, suppress the new-line that is normally sent. Ignored otherwise.
\d	Delay two seconds before sending or reading more characters.
\p	Pause for approximately ¼ to ½ second.

\n	Send a new-line character.
\r	Send a carriage-return.
\s	Send a space character.
\t	Send a tab character.
\\	Send a \ character.
EOT	Send EOT character (actually EOT new-line is sent twice).
BREAK	Send a break character.
\ddd	Collapse the octal digits (ddd) into a single character and send that character.

Dialcodes File

The **Dialcodes** file (**/usr/lib/uucp/Dialcodes**) contains the dial-code abbreviations used in the Phone field of the **Systems** file. Each entry has the format

abb dial-seq

where abb is the abbreviation used in the **Systems** file (Phone field) and dial-seq is the dial sequence that is passed to the dialer when that particular **Systems** entry is accessed.

The entry

jt 9=847-

would be set up to work with a Phone field in the **Systems** file such as jt7867. When the entry containing jt7867 is encountered, the sequence 9=847-7867 would be sent to the dialer.

The management of this file is not supported by Simple Administration. Therefore, changes must be made using one of the editors (**ed** or **vi**).

Permissions File

The **Permissions** file (**/usr/lib/uucp/Permissions**) is used to specify the permissions that remote machines have with respect to login, file access, and command execution. Options are provided for restricting the ability to request files and the ability to receive files queued by the local site. In addition, an option is available to specify the commands that a remote site can execute on the local machine. The management of this file is not supported by Simple Administration. Therefore, changes must be made using one of the editors (**ed**, **vi**).

How Entries are Structured

Each entry is a logical line with physical lines terminated with a \ to indicate continuation. Entries are made up of "white space" delimited options. Each option is a name/value pair. These are constructed by an option name followed by a "=" and the value. Note that no white space is allowed within an option assignment.

Comment lines begin with a "#" and they occupy the entire line up to a newline character. Blank lines are ignored (even within multi-line entries).

There are two types of **Permissions** entries:

LOGNAME Specifies permissions that take effect when a remote machine logs in on (calls) your 3B2 Computer.

MACHINE Specifies permissions that take effect when your 3B2 Computer logs in on (calls) a remote machine.

LOGNAME entries will contain a LOGNAME option and MACHINE entries will contain a MACHINE option.

Considerations

The following items should be considered when using the **Permissions** file to restrict the level of access granted to remote machines:

1. All login IDs used by remote machines to login for UUCP type communications must appear in one and only one LOGNAME entry.

2. Any site that is called whose name does not appear in a MACHINE entry, will have the following default permissions/restrictions:

   - Local send and receive requests will be executed.
   - The remote machine can send files to your 3B2 Computers **/usr/spool/uucppublic** directory.
   - The commands sent by remote machine for execution on your 3B2 Computer must be one of the default commands; usually **rmail**.

Options

This section provides the details of each option, specifying how they are used and their default values.

REQUEST

When a remote machine calls your 3B2 Computer and requests to receive a file, this request can be granted or denied. The REQUEST option specifies whether the remote machine can request to set up file transfers from your 3B2 Computer. The string

REQUEST=yes

specifies that the remote machine can request to transfer files from your 3B2 Computer. The string

REQUEST=no

specifies that the remote machine cannot request to receive files from your 3B2 Computer. The "no" string is the default value. It will be used if the REQUEST option is not specified. The REQUEST option can appear in either a LOGNAME (remote calls you) entry or a MACHINE (you call remote) entry.

SENDFILES

When a remote machine calls your 3B2 Computer and completes its work, it may attempt to take work your 3B2 Computer has queued for it. The SENDFILES option specifies whether your 3B2 Computer can send the work queued for the remote machine. The string

SENDFILES=yes

specifies that the 3B2 Computer may send the work that is queued for the remote machine as long as it logged in as one of the names in the LOGNAME option. This string is mandatory if the 3B2 Computer is in a 'passive mode' with respect to the remote machine. The string

SENDFILES=call

specifies that files queued in your 3B2 Computer will only be sent when the 3B2 Computer calls the remote machine. The call value is the default for the SENDFILE option. This option is only significant in LOGNAME entries since MACHINE entries apply when calls are made out to remote machines. If the option is used with a MACHINE entry, it will be ignored.

READ and WRITE

These options specify the various parts of the file system that **uucico** can read from or write to. The READ and WRITE options can be used with either MACHINE or LOGNAME entries.

The default for both the READ and WRITE options is the **uucppublic** directory as shown in the following strings:

READ=/usr/spool/uucppublic WRITE=/usr/spool/uucppublic

The strings

READ=/ WRITE=/

specify permission to access any file that can be accessed by a local user with "other" permissions.

The value of these entries is a colon separated list of path names. The READ option is for requesting files, and the WRITE option for depositing files. One of the values must be the prefix of any full path name of a file coming in or going out. To grant permission to deposit files in /usr/news as well as the public directory, the following values would be used with the WRITE option:

WRITE=/usr/spool/uucppublic:/usr/news

It should be pointed out that if the READ and WRITE options are used, all path names must be specified because the path names are not added to the default list. For instance, if the /usr/news path name was the only one specified in a WRITE option, permission to deposit files in the public directory would be denied.

NOREAD and NOWRITE

The NOREAD and NOWRITE options specify exceptions to the READ and WRITE options or defaults. The strings

READ=/ NOREAD=/etc WRITE=/usr/spool/uucppublic

would permit reading any file except those in the **/etc** directory (and its subdirectories - remember, these are prefixes) and writing only to the default **/usr/spool/uucppublic** directory. NOWRITE works in the same manner as the NOREAD option. The NOREAD and NOWRITE can be used in both LOGNAME and MACHINE entries.

CALLBACK

The CALLBACK option is used in LOGNAME entries to specify that no transaction will take place until the calling system is called back. The string

CALLBACK=yes

specifies that your 3B2 Computer must call the remote machine back before any file transfers will take place.

The default for the COMMAND option is

CALLBACK=no

The CALLBACK option is very rarely used. Note that if two sites have this option set for each other, a conversation will never get started.

COMMANDS

Warning: The COMMANDS option can be hazardous to the security of your system. Use it with extreme care.

The **uux** program will generate remote execution requests and queue them to be transferred to the remote machine. Files and a command are sent to the target machine for remote execution. The COMMANDS option can be used in MACHINE entries to specify the commands that a remote machine can execute on your 3B2 Computer. The string

COMMANDS=rmail

indicates the default commands that a remote machine can execute on your 3B2 Computer. If a command string is used in a MACHINE entry, the default commands are overridden. For instance, the entry

```
MACHINE=owl:raven:hawk:dove \
COMMANDS=rmail:rnews:lp
```

overrides the COMMAND default such that the command list for machines owl, raven, hawk, and dove now consists of **rmail**, **rnews** and **lp**.

In addition to the names as specified above, there can be full path names of commands. For example,

```
COMMANDS=rmail:/usr/lbin/rnews:/usr/local/lp
```

specifies that command **rmail** uses the default path. The default paths for the 3B2 Computer are **/bin**, **/usr/bin**, and **/usr/lbin**. When the remote machine specifies **rnews** or **/usr/lbin/rnews** for the command to be executed, **/usr/lbin/rnews** will be executed regardless of the default path. Likewise, **/usr/local/lp** is the **lp** command that will be executed.

Including the "ALL" value in the list means that any command from the remote machine(s) specified in the entry will be executed. If you use this value, you give the remote machine full access to your 3B2 Computer.

The string

```
COMMANDS=/usr/lbin/rnews:ALL:/usr/local/lp
```

illustrates two points. The ALL value can appear anywhere in the string. And, the path names specified for rnews and lp will be used (instead of the default) if the requested command does not contain the full path names for **rnews** or **lp**.

The VALIDATE option should be used with the COMMANDS option whenever potentially dangerous commands like **cat** and **uucp** are specified with the COMMANDS option. Any command that reads or writes files is potentially dangerous to local security when executed by the UUCP remote execution daemon (**uuxqt**).

VALIDATE

The VALIDATE option is used in conjunction with the COMMANDS option when specifying potentially dangerous commands. It is used to provide a certain degree of verification of the caller's identity. The use of the VALIDATE option requires that privileged machines have a unique login/password for UUCP transactions. An important aspect of this validation is that the login/password associated with this entry be protected. If an outsider gets that information, that particular VALIDATE option can no longer be considered secure.

A great deal of consideration should be given to providing a remote machine with a privileged login and password for UUCP transactions. Giving a remote machine a special login and password with file access and remote execution capability is like giving anyone on that machine a normal login and password on your 3B2 Computer. Therefore, if you cannot trust someone on the remote machine, do not provide that machine with a privileged login and password.

The LOGNAME entry

LOGNAME=uucpfriend VALIDATE=eagle:owl:hawk

specifies that if one of the remote machines that claims to be eagle, owl, or hawk logs in on your 3B2 Computer, it must have used the login uucpfriend. As can be seen, if an outsider gets the uucpfriend login/password, masquerading is trivial.

But what does this have to do with the COMMANDS option, which only appears in MACHINE entries? It links the MACHINE entry (and COMMANDS option) with a LOGNAME entry associated with a privileged login. This link is needed because the execution daemon is not running while the remote machine is logged in. In fact, it is an asynchronous process with no knowledge of what machine sent the execution request. Therefore, the real question is how does your 3B2 Computer know where the execution files came from?

Each remote machine has its own "spool" directory on your 3B2 Computer. These spool directories have write permission given only to the UUCP programs. The execution files from the remote machine are put in its spool directory after being transferred to your 3B2 Computer. When the **uuxqt** daemon runs, it can use the spool directory name to find the MACHINE entry in the **Permissions** file and get the COMMANDS list, or if the machine name does not appear in the **Permissions** file, the default list will be used.

The following example shows the relationship between the MACHINE and LOGNAME entries:

```
MACHINE=eagle:owl:hawk REQUEST=yes \
COMMANDS=ALL \
READ=/  WRITE=/

LOGNAME=uucpz VALIDATE=eagle:owl:hawk \
REQUEST=yes SENDFILES=yes \
READ=/  WRITE=/
```

These entries provide unlimited read, write, and command execution for the remote machines eagle, owl, and hawk. The ALL value in the COMMANDS option means that any command can be executed by either of these machines. Using the ALL value gives the remote machine unlimited access to your 3B2 Computer. In fact, files that are only readable or writable by user "uucp" (like **Systems** or **Devices**) can be accessed using commands like **ed**. This means a user on one of the privileged machines can write in the **Systems** file as well as read it!

In the first entry, you must make the assumption that when you want to call one of the machines listed, you are really calling either eagle, owl, or hawk. Therefore, any files put into one of the eagle, owl, or hawk spool directories is put there by one of those machines. If a remote machine logs in and says that it is one of these three machines, its execution files will also be put in the privileged spool directory. You therefore have to validate that the machine has the privileged login "uucpz."

MACHINE Entry for "Other" Systems

You may want to specify different option values for the machines your 3B2 Computer calls that are not mentioned in specific MACHINE entries. This may occur when there are many machines calling in, and the command set changes from time to time. The name "OTHER" for the machine name is used for this entry as shown below:

```
MACHINE=OTHER \
COMMANDS=rmail:rnews:/usr/lbin/Photo:/usr/lbin/xp
```

All other options available for the MACHINE entry may also be set for the machines that are not mentioned in other MACHINE entries.

Combining MACHINE and LOGNAME Entries

It is possible to combine MACHINE and LOGNAME entries into a single entry where the common options are the same. For example, the two entries

```
MACHINE=eagle:owl:hawk REQUEST=yes \
  READ=/  WRITE=/

LOGNAME=uucpz REQUEST=yes SENDFILES=yes \
  READ=/  WRITE=/
```

share the same REQUEST, READ, AND WRITE options. These two entries can be merged into one entry as shown below:

```
MACHINE=eagle:owl:hawk REQUEST=yes \
LOGNAME=uucpz SENDFILES=yes \
  READ=/  WRITE=/
```

Sample Permissions Files

Example 1

This first example represents the most restrictive access to your computer.

```
LOGNAME=nuucp
```

It states that login "nuucp" has all the default permissions/restrictions:

- The remote machine can only send files to **uucppublic**.
- The remote machine cannot request to receive files (REQUEST option).
- No files that are queued for the remote machine will be transferred during the current session (SENDFILES option).
- The only commands that can be executed are the defaults.

This entry alone is sufficient to start communications with remote machines, permitting files to be transferred only to the **/usr/spool/uucppublic** directory.

Example 2

The next example is for remote machines that log in, but have fewer restrictions. The login and password corresponding to this entry should not be distributed to the general public; it is usually reserved for closely coupled systems where the **Systems** file information can be tightly controlled.

```
LOGNAME=uucpz REQUEST=yes SENDFILES=yes \
  READ=/  WRITE=/
```

This entry places the following permissions/restrictions on a machine that logs in as "uucpz"

- Files can be requested from your 3B2 Computer (REQUEST option).
- Files can be transferred to any directory or any file that is writable by user "other." That is a file/directory that is writable by a local user with neither owner nor group permissions (WRITE option).
- Any files readable by user "other" can be requested (READ option).
- Any requests queued for the remote machine will be executed during the current session. These are files destined for the machine that has called in (SENDFILES option).
- The commands sent for execution on the local machine must be in the default set.

Example 3

The two previous examples showed entries that referred to remote machines when they log into your 3B2 Computer. This example is an entry used when calling remote machines.

```
MACHINE=eagle:owl:hawk:raven \
  REQUEST=yes READ=/  WRITE=/
```

When calling any of the systems given in the MACHINE list, the following permissions prevail:

- The remote machine can both request and send files (REQUEST option).
- The source or destination of the files on the local machine can be anywhere in the file system.
- The only commands that will be executed for the remote machine are those in the default set.

Any site that is called that does not have its name in a machine entry will have the default permissions as stated in Example 1, with the exception that files queued for that machine will be sent (the SENDFILES option is only interpreted in the LOGNAME entry).

Poll File

The **Poll** file (**/usr/lib/uucp/Poll**) contains information for polling specified Machines. Each entry in the **Poll** file contains the name of the remote machine to call, followed by a TAB character, and finally the hours the machine should be called. The entry

eagle 0 4 8 12 16 20

will provide polling of machine eagle every four hours.

> ***Note:*** It should be understood that **uudemon.poll** does not actually perform the poll, it merely sets up a polling work (C.) file in the spool directory that will be seen by the scheduler, started by **uudemon.hour**. Refer to the discussion on **uudemon.poll**.

Maxuuxqts File

The **Maxuuxqts** (**/usr/lib/uucp/Maxuuxqts**) file contains an ASCII number to limit the number of simultaneous **uuxqt** programs running. This file is delivered with a default entry of one. This may be changed to meet local needs. If there is a lot of traffic from **mail**, it may be advisable to increase this number to reduce wait time. However, keep in mind that the load on the system increases with the number of **uuxqt** programs running.

Maxuuscheds File

The **Maxuuscheds** (**/usr/lib/uucp/Maxuuscheds**) file contains an ASCII number to limit the number of simultaneous **uusched** programs running. Each **uusched** running will have one **uucico** associated with it; limiting the number will directly affect the load on the system. The limit should be less than the number of outgoing lines used by UUCP (a smaller number is often desirable). This file is delivered with a default entry of one. Again, this may be changed to meet the needs of the local system. However, keep in mind that the load on the system increases with the number of **uusched** programs running.

remote.unknown

The **remote.unknown** program (**/usr/lib/uucp/remote.unknown**) is a shell file that is executed when a remote site that is not in the **Systems** file calls in to start a conversation. The shell script will append the name and time information to the file **/usr/spool/uucp/.Admin/Foreign**. Since it is a shell, it can be easily modified. For example, it can be set up to send mail to the administrator. The contents of this file, as delivered, is as follows:

```
FOREIGN=/usr/spool/uucp/.Admin/Foreign
echo "`date`: call from system $1" >>$FOREIGN
```

If you want to permit unknown machines to converse, you may change the mode of the **remote.unknown** file to unexecutable (444).

ADMINISTRATIVE TASKS

There is a minimum amount of maintenance that must be applied to your 3B2 Computer to keep the files updated, to ensure that the network is running properly, and to track down line problems. When more than one remote machine is involved, the job becomes more difficult because there are more files to update and because users are much less patient when failures occur between machines that are under local control. The **uustat** program provides you with information about the latest attempts to contact various machines and the age and number of jobs in the queue for remote machines. The following sections describe the routine administrative tasks that must be performed by someone acting as the UUCP administrator or are automatically performed by the UUCP daemons (demons).

The biggest problem in a dialup network like UUCP is dealing with the backlog of jobs that cannot be transmitted to other machines. The following cleanup activities should be routinely performed.

Cleanup of Undeliverable Jobs

The **uustat** program should be invoked regularly to provide information about the status of connections to various machines and the size and age of the queued requests. The **uudemon.admin** shell should be started by **cron** at least once per day. This will send the administrator the current status. Of particular interest are the age (in days) of the oldest request in each queue, the number of times a failure has occurred when attempting to reach that machine, and the reason for failure. In addition, the age of the oldest execution request (X. file) is also given.

The **uudemon.cleanu** shell file is set up to remove any jobs that have been queued for several days and cannot be sent. Leftover data (D.) and work (C.) files are removed after seven days, and execute (X.) files are removed after two days. It also provides feedback to the user indicating when jobs are not being accomplished and when these jobs are being deleted.

Cleanup of the Public Area

In order to keep the local file system from overflowing when files are sent to the public area, the **uudemon.cleanu** procedure is set up with a **find** command to remove any files that are older than seven days and directories that are empty. This interval may need to be shortened by changing the **uudemon.cleanu** shell file if there is not sufficient space to devote to the public area.

Since the spool directory is very dynamic, it may grow large before transfers take place. Therefore, it is a good idea to reorganize its structure. The best way to do this on your 3B2 Computer is to put some code in the **/etc/rc.d** file to execute whenever the system is booted. The following is an example of such code:

```
#       Clean up /usr/spool/uucp
#       Most cleanup is now done by uudemon.cleanu
#       so just copy out and back.
#
echo " UUCP SPOOL DIRECTORIES CLEANUP STARTED"
#
cd /usr/spool/uucp
mkdir ../nuucp
chown uucp ../nuucp
chgrp uucp ../nuucp
find . -printlcpio -pdml ../nuucp
cd ..
mv uucp ouucp
mv nuucp uucp
rm -rf ouucp
#
chown uucp /dev/cu[al]*
chgrp uucp /dev/cu[al]*
chmod 0666 /dev/cul*
chmod 0222 /dev/cua*
echo " UUCP SPOOL DIRECTORIES CLEANUP FINISHED"
```

Compaction of Log Files

This version of Basic Networking has individual log files for each machine and each program (e.g. machine eagle has a logfile for **uucico** requests and a logfile for **uuxqt** execution requests). The **uulog** program gives the user access to the information in these files by machine name. These files are combined and stored in directory **/usr/lib/uucp/.Old** whenever **uudemon.cleanu** is executed. This shell script saves files that are two days old. The two days can be easily changed by changing the appropriate line in the **uudemon.cleanu** shell. If space is a problem, the administrator might consider reducing the number of days the files are kept.

Cleanup of sulog and cron log

The **/usr/adm/sulog** and **/usr/lib/cron/log** files are both indirectly related to UUCP transactions. The **sulog** file contains a history of the **su** command usage. Since the uudemon entries in the **/usr/lib/cron/root** file each use the **su** command, the **sulog** could become rather large over a period of time. The **sulog** should be periodically purged to keep the file at a reasonable size.

Similarly, a history of all processes spawned by **/etc/cron** are recorded in **/usr/lib/cron/log**. The cron **log** file will also become large over a period of time and should be periodically purged to limit its size. The *3B2 System Administration Utilities Guide* contains information on the purging of these files.

UUCP AND CRON

The **cron** daemon is a tool that proves to be very useful in the administration of UNIX Systems. When the 3B2 Computer is in run state 2 (multi-user), **cron** scans the **/usr/spool/cron/crontab/root** file every minute for entries that contain "work" scheduled to be executed at that time. It is recommended that the UUCP administrator make use of **cron** to aid in the administration of the Basic Networking Utilities.

As delivered, the Basic Networking Utilities contain four entries in the **root** crontab file. Each one of these entries execute shell scripts which are used for various administrative purposes. These shell scripts can be easily modified to meet the needs of your system.

uudemon.admin

The **uudemon.admin** shell script mails status information to the UUCP administrative login (**uucp**) using **uustat** commands with the -p and -q options. Refer to the **uustat** manual page for interpretation of these options.

The **uudemon.admin** shell script should be executed daily by an entry in the root crontab file. The default **root** crontab entry for **uudemon.admin** is as follows:

```
48 8,12,16 * * * /bin/su uucp -c " /usr/lib/uucp/uudemon.admin" > /dev/null
```

uudemon.cleanu

The **uudemon.cleanu** shell script cleans up the Basic Networking log files and directories. Archived log files are updated so that no log information over three days old is kept. Log files for individual machines are taken from the **/usr/spool/uucp/.Log** directory, merged, and placed in the **/usr/spool/uucp/.Old** directory along with the older log information. Files and directories that are no longer needed in the spool directories are removed. After clean up is performed, the UUCP administrative login (**uucp**) is mailed a summary of the status information gathered during the current day.

The **uudemon.cleanu** shell script should be executed by an entry in the **root** crontab file. It can be run daily, weekly, or whenever, depending on the amount of UUCP traffic which enters and leaves your 3B2 Computer. The default root crontab entry for **uudemon.cleanu** is as follows:

```
45 23 * * * ulimit 5000; /bin/su uucp -c " /usr/lib/uucp/uudemon.cleanu" > /dev/null 2>&1
```

If log files get very large, the ulimit may need to be increased.

uudemon.hour

The **uudemon.hour** shell script is used to call UUCP programs on an hourly basis. The **uusched** program is called to search the spool directory for work files (C.) that have not been processed and schedule these files for transfer to a remote machine. The **uuxqt** daemon is called to search the spool directory for execute files (X.) that have been transferred to your 3B2 Computer and were not processed at the time they were transferred.

The **uudemon.hour** shell script should be executed by an entry in the root crontab file. If the amount of traffic leaving and entering your 3B2 Computer is large, it may be started once or twice an hour. If it is small, it may be started once every four hours or so. The default root crontab entry for **uudemon.hour** is as follows:

```
41,11 * * * * " /usr/lib/uucp/uudemon.hour > /dev/null
```

uudemon.poll

The **uudemon.poll** shell script is used to poll the remote machines listed in the **Poll** file (**/usr/lib/uucp/Poll**). It creates work files (C.) for machines according to the entries listed in the **Poll** file. It should be set up to run once an hour just prior to **uudemon.hour** so that the work files will be present when **uudemon.hour** is called.

The **uudemon.poll** script should be executed by an entry in the root crontab file. The exact times it runs should be dependent on the scheduling of **uudemon.hour**. The default root crontab entry for **uudemon.poll** is as follows:

```
1,30 * * * * " /usr/lib/uucp/uudemon.poll > /dev/null
```

Note how **uudemon.poll** is scheduled to run eleven minutes before **uudemon.hour** runs.

INITTAB ENTRIES

The **/etc/inittab** file contains information for the processes to be spawned on the 3B2 Computer devices, including the ports. This file should normally be managed by the uucpmgmt Simple Administration subcommand **portmgmt**. Ports that are used by Basic Networking are normally bidirectional ports. Bidirectional ports can be used to receive incoming calls, as well as place outgoing calls. The **uugetty** program is used in place of **getty** for those bidirectional ports associated with Basic Networking. For additional information on the **inittab** entries associated with Basic Networking, refer to the Simple Administration subcommand **portmgmt** and the **uugetty** manual page (Appendix A).

UUCP LOGINS AND PASSWORDS

There are two login IDs associated with the Basic Networking Utilities: one is the UUCP administrative login **uucp**, and the other is an access login (**nuucp**) used by remote computers to access your 3B2 Computer. These logins should not be changed from their default settings of **uucp** and **nuucp**.

The **uucp** administrative login is the owner of all the UUCP object and spooled data files. The following is a sample entry in the **/etc/passwd** file for the administrative login:

```
uucp:zAvLCKp:5:1:UUCP.Admin:/usr/lib/uucp:
```

The **nuucp** access login allows remote machines to login on your 3B2 Computer. The following is a sample entry in the /etc/passwd file for the access login:

```
nuucp:zaaAA:6:1:UUCP.Admin:/usr/spool/uucppublic:/usr/lib/uucp/uucico
```

Note that the standard shell is not given to the **nuucp** login. The shell that **nuucp** receives is the **uucico** daemon which controls the conversation when a remote machine logs into your machine.

The assigning of passwords for the **uucp** and **nuucp** logins is left up to the administrator. The passwords should be at least six to eight characters with a rich alphabet. Only the first eight characters of the passwords are significant. If the password for the access login is changed for security reasons, make certain that the remote machines that are a part of your network are properly notified of the change.

Chapter 4

SIMPLE ADMINISTRATION

PAGE

Chapter 4

SIMPLE ADMINISTRATION

INTRODUCTION

The Basic Networking Utilities Management (uucpmgmt) package is a set of Simple Administration scripts used to maintain certain support files of the Basic Networking Utilities. The support files provide information that enables your 3B2 Computer to contact remote machines for the purpose of sending and receiving mail and files, as well as executing commands on remote machines. There are four **uucpmgmt** subcommands that add, delete, list, and modify the contents of certain Basic Networking Utilities support files.

The support files managed by **uucpmgmt** are listed below:

- **Systems** (/usr/lib/uucp/Systems): This file contains the names and calling information for the communication program, permitting it to call and converse with remote machines. This file is managed using the **systemmgmt** subcommand.

- **Devices** (/usr/lib/uucp/Devices): This file contains a list of the devices which may be used to call remote machines. Device type, speed, port name (i.e. /dev/contty) are some of the information contained in the **Devices** file. This file is managed using the **devicemgmt** subcommand.

- **inittab** (etc/inittab): Information in this file controls the direction of traffic on the ports used by the Basic Networking Utilities. Ports can be designated as incoming, outgoing, or bidirectional. This file is managed using the **portmgmt** subcommand.

- **Poll** (/usr/lib/uucp/Poll): This file contains a list of the machines to be polled by your 3B2 Computer, and the times they are to be polled. This file is managed using the **pollmgmt** subcommand.

FUNCTIONALITY

The **uucpmgmt** menu is located under the **packagemgmt** menu of simple administration (sysadm). In order for **uucpmgmt** to appear in the **packagemgmt** menu, the Basic Networking Utilities must first be loaded from the floppy disk. For each subcommand, you can select the operation to be performed by entering either the associated number or the name of the subcommand.

The subcommands are organized as a series of questions and answers. Answers take the form of either an entry from a list, a y/n response, and a name or a number depending on the question. At any point, the user can type an "q" to quit the operation. When indicated as a possible default choice, entering a carriage return (<CR>) selects the default entry. If the selection list includes a "?", entering a "?" outputs the associated help message and then returns the user to the question.

Common Functionality

The following paragraphs discuss the common functions of each **uucpmgmt** subcommand.

Required Files Test: Common to the list, delete, and modify operations is a test for the existence of the file(s). If that file does not exist, the user is advised about the appropriate operations to be performed. After the advice message has been displayed, the operation is terminated, returning the user to the **uucpmgmt** menu.

List Operation: Upon entry to the list operation, a summary list of all the names known to the Basic Networking Utilities is presented. For the **systemmgmt** subcommand, the summary information is the list of systems in the **Systems** file. For **devicemgmt**, it is the list of ports; for **pollmgmt**, the systems being polled; and for **portmgmt**, the list of ports listed in the **Devices** file. You are then asked to enter the name you want to see in detail. Only names that appear on the summary list are permissible responses. When a valid name is entered, the line(s) for that name is presented. You are then asked if you would like to see another entry. If an "n" response is given, the operation is exited, returning you to the **uucpmgmt** menu. A "y" response again presents the summary list and prompts you to input the appropriate name for the detailed output.

Delete Operation: For the delete operation, you are presented the current list of available names from which deletions can be made. After you are asked to specify which name should be deleted, the line(s) corresponding to the name is displayed and you are then prompted whether you want to delete the entry(ies). An "n" response removes the line(s), a "y" response does not. This is followed by a question, asking if you would like to delete another entry. A "y" response again displays the current list of names from which deletions can be made. An "n" response returns you to the **uucpmgmt** menu.

Add Operation: The add operation prompts you for data required to make the individual entries. After all the data has been input, the generated line(s) is presented, followed by a request asking if you want to add the entry. A "y" response adds the line(s) to the file.

An "n" response takes you back to the **uucpmgmt** menu.

Modify Operation: The only modify operation is in the **portmgmt** subcommand. It is later described in the **portmgmt** section.

SUBCOMMANDS

This section contains a detailed description of each of the four subcommands; **systemmgmt**, **devicemgmt**, **portmgmt**, and **pollmgmt**.

systemmgmt

This subcommand manages the **Systems** file (/usr/lib/uucp/Systems). It permits four operations; add, delete, call, and list.

add: For the add operation, the **Systems** file is checked for read and write permission; failure causes the operation to terminate with a message. After a brief introduction to the subcommand, you are prompted for the fields needed to make entries in the file. The contents of these fields are listed below:

- Node name of system you want to call (e.g. eagle)
- Type of device used to originate call (e.g. acu)
- Speed at which call is placed (e.g. 1200)
- Phone number of machine or token used to access connection device through a switch (e.g. 9=847-7867)
- Type of equipment you are dialing into (e.g. dialup)

- Login ID at remote machine (e.g. nuucp)
- Password entry at remote machine. (e.g. Oakgrass).

After the above information has been supplied, an entry for the **Systems** file is constructed and the contents of the entry are displayed on the screen for verification. You are then prompted to indicate whether or not to add the entry to the **Systems** file. After y/n response is received, you are asked if another entry is to be made to the **Systems** file. Again, an "n" response returns you to the **uucpmgmt** menu and a "y" response takes you back to the beginning of the add option.

delete: As described in the "Common Functionality" section.

list: As described in the "Common Functionality" section.

call: This option is used to call a remote machine that has been properly entered in the **Systems** file. It is a test to ensure that all necessary entries have been entered into the **Systems** file correctly.

devicemgmt

This subcommand manages the **Devices** file (/usr/lib/uucp/Devices). It permits three operations; add, delete, and list.

add: The **Devices** file is checked for read and write permission. Failure returns you to the **uucpmgmt** menu. You are prompted for the name of a device (port) used to make the call (e.g. contty). The name entered is checked against the current **Devices** file and if it exists, you are asked if other entries are to be added. For each entry, you are prompted for the following additional data:

- Device type (e.g. penril, ventel, develcon, micom)
- Speed (e.g. 1200).

Note: If one of the modems is selected, the procedure will automatically generate two entries, one at 1200 bps and one at 300 bps without prompting for speed.

After all of the requested data has been provided, one or more entries are created for the **Devices** file and the contents of the entries are displayed on the screen for verification. You are then prompted to indicate whether or not to add the entry to the **Devices** file. After a y/n response is received, you are asked if another entry is to be made to the **Devices** file. Again, an "n" response returns you to the **uucpmgmt** menu and a "y" response takes you back to the beginning of the add operation.

delete: As described in the "Common Functionality" section.

list: As described in the "Common Functionality" section.

pollmgmt

This subcommand manages the **Poll** file (/usr/lib/uucp/Poll). It permits three operations: add, delete, and list.

add: You are prompted to enter the name of the system to be polled. If the system name already exists in the **Poll** file, a message advising that no duplicates are permitted is output. You have the opportunity to add another system or to terminate the operation. If the system name does not appear in the **Systems** file, you are warned and asked whether or not to continue. If a "y" response is received, the process continues. If an "n" response is receive, you are asked about making other poll entries.

If a potential system name is entered, you are asked for a space separated list of hours for polling. Each hour entered must be an integer in the range of 0 through 23. Invalid responses will result in a message describing the list of hours. The default value for this question causes the specified system to be polled every hour.

Upon successful input of the hours list, the **Poll** entry (consisting of the system name followed by the polling hours) is displayed on the screen for verification. You are then prompted to indicate whether or not to add the entry to the **Poll** file. After a y/n response is received, you are asked if another entry is to be made to the **Poll** file. Again, an "n" response returns you to the **uucpmgmt** menu and a "y" a response takes you back to the beginning of the add option.

delete: As described in the "Common Functionality" section.

list: As described in the "Common Functionality" section.

portmgmt

This subcommand manages the **/etc/inittab** file. It permits two operations; modify and list. The only ports from inittab that are accessible are those that appear in the **Devices** file.

modify: For a valid port, the direction of the port is printed. You are prompted for the desired direction (incoming, outgoing, or bidirectional) and speed (default is current speed). This data is used to modify the **inittab** entry and the modified entry is displayed on the screen for verification. You are prompted to indicate whether or not the modified entry is correct. After a y/n response is received, you are asked if another inittab entry is to be modified. Again, an "n" response returns you to the **uucpmgmt** menu and a "y" response takes you back to the beginning of the modify option.

If the direction is set to bidirectional, this sets up the **inittab** entry up to respawn **uugetty** on the port. If incoming is specified, the entry is set up to respawn **getty** on the port, and for outgoing, respawn is turned off.

list: As described in the "Common Functionality" section.

ASSUMPTIONS

The following assumptions are made by **uucpmgmt** upon execution:

- **/usr/lib/uucp/Systems** is readable only by **uid**=**uucp**. It is not writable.
- **/usr/lib/uucp/Devices** is readable by all. It is not writable.
- **/usr/lib/uucp/Poll** is readable by all. It may also be writable.
- **/etc/inittab** is usually owned by **root** and readable by all. It should not be writable.

Chapter 5

DIRECT LINKS

PAGE

Chapter 5

DIRECT LINKS

GENERAL

This chapter discusses how to directly link

- two 3B2 Computers
- a 3B2 Computer to a 3B5 Computer
- a 3B2 Computer to a 3B20 Computer.

Direct links would be beneficial only when:

- It is not possible to link the machines together through a Local Area Network (LAN).
- The two machines transfer large amounts of data on a regular basis
- The two machines are located no more than several hundred cable feet apart.

The amount of cable used to link two machines is dependent on the environment in which the cable is run. The standard for RS-232 connections is 50 feet or less with transmission rates as high as

19200 bits per second (bps). As the cable length is increased, noise on the lines may become a problem which means that the transmission rate must be decreased or limited distance modems be placed on each end of the line. Normally, you should not use more than 1000 cable feet to connect the two machines. This link should operate comfortably at 9600 bps in a clean (noise free) environment.

The configuration used to hardwire two 3B Computers together uses two 8-wire cables (available in lengths of 7, 14, 25, and 50 feet) and two RS-232 connectors. The ordering information is provided in Figure 5-1.

DESCRIPTION	COMCODE NUMBER
7-Foot Shielded Cable	403-60-09-68
14-Foot Shielded Cable	403-60-09-76
25-Foot Shielded Cable	403-60-09-84
50-Foot Shielded Cable	403-60-09-92
ACU/Modem Connector	232-21-25-005
Terminal/Printer Connector	232-22-25-006

Figure 5-1. Part Numbers for Hardware Used in Directs Links

If the link is established using the parts listed in Figure 5-1, the machines could not be separated by more than 100 cable-feet (two

50-foot cables connected together) because the longest cable available is 50 feet.

If the two machines are separated by more than 100 cable-feet, a null-modem cable must be constructed as follows:

Pin 1 to 1
Pin 2 to 3
Pin 3 to 2
Strap pin 4 to 5 in the same plug
Pin 6 to 20
Pin 7 to 7
Pin 8 to 20
Pin 20 to 6
Pin 20 to 8.

HOW THE DIRECT LINK IS CONNECTED

3B2 Computer to 3B2 Computer Direct Link

Using the parts listed in Figure 5-1, the establishment of a direct link between two 3B2 Computers is very simple. The following steps will guide you in establishing a direct link between two 3B2 Computers (also, refer to Figure 5-2).

1. Connect one end of the first shielded cable to the selected port on your 3B2 Computer. Be sure to attach the ground connector.

2. Connect the other end of the first shielded cable to the ACU/Modem Adapter. (If it is desired to connect the two computers over a distance greater than 100 feet, substitute the ACU/Modem Adapter with a Terminal/Printer Adapter attached to a null modem cable of the appropriate length.)

3. Connect the Terminal/Printer Adapter to the ACU/Modem Adapter.

4. Connect the second shielded cable to the Terminal/Printer Adapter.

5. Connect the other end of the second shielded cable to the appropriate port on the remote 3B2 Computer.

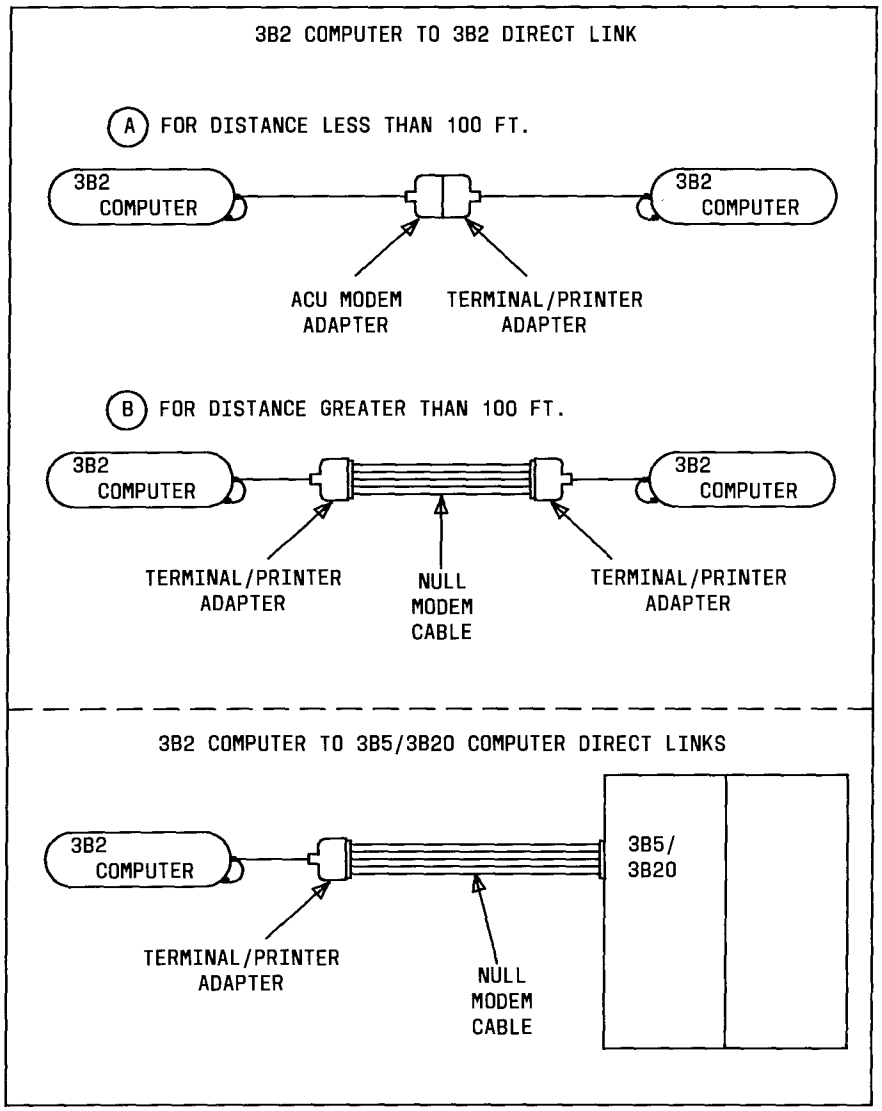


Figure 5-2. Examples of Direct Links

3B2 Computer to 3B5 Computer or 3B2 Computer to 3B20 Computer Direct Link

Establishing a direct link between a 3B2 Computer and a 3B5 Computer or between a 3B2 Computer and a 3B20 Computer is simple. The following steps will guide you in establishing a direct link between a 3B2 Computer and a 3B5 Computer or between a 3B2 and a 3B20 Computer (also, refer to Figure 5-2).

Note: The same parts are used for either link.

1. Connect one end of a shielded cable to the selected port on your 3B2 Computer. Be sure to attached the ground connector.

2. Connect the other end of the shielded cable to a Terminal/Printer Adapter.

3. Connect a null modem cable to appropriate port on the remote 3B5 Computer or 3B20 Computer.

BASIC NETWORKING SOFTWARE AND DIRECT LINKS

Ideally, systems that have a direct link should run common and current releases of the UNIX System to have the full set of capabilities available. (Bidirectional ports which is supported by the **uugetty** program was introduced with UNIX System V Release 2.0 Version 1.) However, lack of commonality does not prevent utilization of the Basis Networking feature. This section describes the software files that must be modified on your 3B2 Computer in order to accommodate a direct link connection. You may want to consult the documentation provided with your machine if your linking directly to a remote machine other than a 3B2 Computer.

The following support files must be updated to reflect the presence of a Direct Link:

- **/usr/lib/uucp/Devices**
- **/etc/inittab**
- **/usr/lib/uucp/Systems**.

All additions/modifications to the above files can be accomplished using the **uucpmgmt** Simple Administration subcommands.

Making Entries into the Devices File

The **Devices** file contains the information concerning the location (line) and transmission rate of the link. Entries can be added to the **Devices** file using the uucpmgmt **devicemgmt** subcommand. The operation to be performed under **devicemgmt** is the add operation. The add operation prompts you for the following information:

- Port name (for /dev/tty21 - **tty21**)
- Device type to call on (**direct**)
- Speed at which you want to call (**9600** or **19200**).

To access the **devicemgmt** subcommand and add entries in the **Devices** file, enter:

```
# sysadm devicemgmt<CR>

Running subcommand 'devicemgmt' from menu 'uucpmgmt',
BASIC NETWORKING UTILITIES MANAGEMENT

Note: After a brief introduction to the devicemgmt
subcommand, the procedure interactively prompts you for
the information listed above.
```

Making Changes to the /etc/inittab File

The differences between the two versions of Basic Networking (UUCP) are reflected in the **/etc/inittab** file. The newest version allows for bidirectional login capability, as well as communication by respawning **uugetty** instead of **getty**. This means that if two machines (both using **uugetty**) were connected via a direct link, either of these machines could request communication with the other. This would not be true if only one machine was capable of respawning **uugetty**.

If the direct link is connecting your 3B2 Computer with a machine that has the new version of Basic Networking, the **/etc/inittab** files on both machines should be set up to allow "bidirectional" traffic on the associated lines. This means that the lines used must respawn **uugetty** on each end of the link. This would allow either machine to request communication with (call) the other.

If the direct link is connecting your 3B2 Computer with a machine that does not have the new version of Basic Networking, the **/etc/inittab** file would be set up differently on each system. The **inittab** file on each machine would be set up to allow either "incoming" or "outgoing" traffic on its line. If one machine allows incoming traffic, the other must allow only outgoing traffic. A **uugetty** could not be used on either machine in this case.

A machine's **inittab** entry would be respawning **getty** for "incoming" traffic, or have respawn turned off for "outgoing" traffic. In order for this type of link to work, one of the machines must be set up to "poll" the other. If the remote machine is allowing only incoming traffic, you must set up your 3B2 Computer to poll the remote machine (**uucpmgmt** subcommand **pollmgmt**). If the remote machine is allowing only outgoing traffic on the link, the remote machine must poll your 3B2 Computer.

Entries in the **/etc/inittab** file can be changed using the **uucpmgmt portmgmt** subcommand. The operation to be performed under **portmgmt** is the modify operation. The modify operation prompts you for the following information:

- Port name you want to modify (for /dev/tty21 - **tty21**)
- Direction of traffic on port (**bidirectional**, **incoming**, or **outgoing**)
- Transmission speed of the link (**9600** or **19200**).

The procedure will display the ports that are currently dedicated for use by UUCP (listed in **Devices** file). The port name to be modified must be one that is listed.

To access the **portmgmt** subcommand and modify entries in the **/etc/inittab** file, enter:

```
# sysadm portmgmt<CR>

Running subcommand 'portmgmt' from menu 'uucpmgmt',
BASIC NETWORKING UTILITIES MANAGEMENT

Note: After a brief introduction to the portmgmt
subcommand, the procedure interactively prompts you for
the information listed above.
```

Making Entries into the Systems File

An entry must be made into the **Systems** file for the machine associated with the direct link. This can be done using the **systemmgmt** subcommand and the add operation. The add operation will prompt you for the following information:

- Node name of system
- Type of device to call on (**direct**)
- Transmission speed of link (**9600** or **19200**)
- Device port used with link (for /dev/tty21 - **tty21**)
- Login ID used to login on system (**nuucp**)
- Password used by above login.

In order for the direct link to operate properly at high speeds, you must insert pauses (\p) between the characters being sent out for the login ID and the password. For instance, instead of nuucp, you should enter n\pu\pu\pc\p\pp when prompted for the login ID. Do *not* select the default by pressing RETURN. The same applies for the password assigned.

To access the **systemmgmt** subcommand and add entries in the **Systems** file, enter:

```
# sysadm systemmgmt<CR>

Running subcommand 'systemmgmt' from menu 'uucpmgmt',
BASIC NETWORKING UTILITIES MANAGEMENT

Note: After a brief introduction to the systemmgmt
subcommand, the procedure interactively prompts you for
the information listed above.
```

Upon completion of the add operation, a new entry is added to the **Systems** file for the remote machine and an additional entry was created for the **Devices** file. When you use **devicemgmt** to create an entry for the link, it creates an entry similar to the one shown below.

```
Direct tty21 - 9600 direct
```

As discussed in Chapter 3, this type of entry is used only with the **cu** command. The UUCP programs require that the first field of a **Devices** entry be *System-Name* when associated with a direct link to another machine. This is why the add operation of **systemmgmt** wants to know the device port that will be used in the link to the remote machine. After it receives the port name, it creates a second entry for that port similar to the following:

```
Direct tty21 - 9600 direct
eagle tty21 - 9600 direct
```

The first entry will be used whenever the **cu** command is used to call "eagle", and the second entry will be used by **uucico** to call "eagle."

Chapter 6

MAINTENANCE

PAGE

Chapter 6

MAINTENANCE

This chapter discusses the maintenance of the Basic Networking Utilities. It discusses some of the common problems associated with the utilities and how to use the **Uutry** program for debugging. Also, included is a list of error messages and their probable causes.

COMMON PROBLEMS

Out of Space

The file system used to spool incoming or outgoing jobs can run out of space and prevent jobs from being sent or received. Not being able to receive jobs is the worse of the two conditions. When file space does become available, the 3B2 Computer will be flooded with the backlog of traffic. The shell script **uudemon.cleanu** should keep the spool directory (**/usr/spool/uucp**) from becoming very large. This script should be started by **cron** once a day.

Faulty Automatic Call Units and Modems

The automatic dial modems and/or incoming modems occasionally cause problems that make it difficult to contact other computers or receive files. These problems are usually readily identifiable since the status files accessed by **uustat** give counts and reasons for contact failure. If a bad line is suspected, the **cu** command may be useful in trying to call another computer using the suspected line.

Administrative Problems

Sometimes it can be very difficult to keep your **Systems** file up to date. This is because of changing telephone numbers, login IDs, and passwords on remote computers. This can be a very costly problem since the automatic dial modem will be tied up calling a computer that cannot be reached. Be sure to contact the administrators of remote machines whenever you change your telephone number, login, or password and request that they show you the same consideration.

DEBUGGING

In order to verify that a computer on the network can be contacted, the **uucico** daemon can be invoked directly from a terminal. A shell script, **Uutry**, is provided for this purpose in the **/usr/lib/uucp** directory. This directory is normally not listed in most users' command search path. It must therefore be moved to an appropriate place if it is to be used by all users. If the person using the **uucp** login will be the only one using **Uutry**, it can be left where it is, since the HOME directory for the **uucp** login is **/usr/lib/uucp**.

The command line

```
Uutry eagle
```

will start the transfer daemon (**uucico**) with a moderate amount of debugging output. **Uutry** redirects the output into a temporary file (**/tmp/eagle**) and executes a **tail -f** command on the file. This way,

you can hit a " BREAK" to get back to the shell, and come back later to look at the output in **/tmp/eagle**.

If **Uutry** does not isolate the problem, you can attempt to transfer a file while watching the debugging output as follows:

```
uucp -r some-file eagle!~/some-name
```

The **-r** option will queue a job, but will not start the transfer daemon. Now proceed as before using **Uutry**. If any of these steps fail, support personnel may be needed to diagnose the problem. It will be much easier to diagnose the problem if the debugging output saved in the temporary file is available.

ERROR MESSAGES

This section lists the error messages associated with Basic Networking Utilities. There are two types of error messages. ASSERT errors are recorded in the **/usr/spool/uucp/.Admin/errors** file. STATUS errors are recorded in individual machine files found in the **/usr/spool/uucp/.Status** directory.

ASSERT Error Messages

When a process is aborted, ASSERT error messages are recorded in **/usr/spool/uucp/.Admin/errors**. These messages include the file name, sccsid, line number, and the text listed below. In most cases, these errors are the result of file system problems. The " errno" (when present) should be used to investigate the problem. If " errno" is present in a message, it is shown as () in the following list.

CAN'T OPEN — An open() or fopen() failed.

CAN'T WRITE — A write(), fwrite(), fprint(), etc. failed.

CAN'T READ	A read(), fgets(), etc. failed.
CAN'T CREATE	A create() call failed.
CAN'T ALLOCATE	A dynamic allocation failed.
CAN'T LOCK	An attempt to make a LCK (lock) file failed. In some cases, this is a fatal error.
CAN'T STAT	A stat() call failed.
CAN'T CHMOD	A chmod() call failed.
CAN'T LINK	A link() call failed.
CAN'T CHDIR	A chdir() call failed.
CAN'T UNLINK	A unlink() call failed.
WRONG ROLE	This is an internal logic problem.
CAN'T MOVE TO CORRUPTDIR	An attempt to move some bad C. or X. files to the **/usr/spool/uucp/.Corrupt** directory failed. The directory is probably missing or has wrong modes or owner.
CAN'T CLOSE	A close() or fclose() call failed.
FILE EXISTS	The creation of a C. or D. file is attempted, but the file exists. This occurs when there is a problem with the sequence file access. Usually indicates a software error.

No uucp server	A tcp/ip call is attempted, but there is no server for UUCP.
BAD UID	The uid cannot be found in the **/etc/passwd** file. The file system is in trouble, or the **/etc/passwd** file is inconsistent.
BAD LOGIN_UID	Same as previous.
ULIMIT TOO SMALL	The ulimit for the current user process is too small. File transfers may fail, so transfer is not attempted.
BAD LINE	There is a bad line in the **Devices** file; there are not enough arguments on one or more lines.
FSTAT FAILED IN EWRDATA	There is something wrong with the ethernet media.
SYSLST OVERFLOW	An internal table in gename.c overflowed. A big/strange request was attempted. Contact your AT&T Account Representative or Authorized Dealer.
TOO MANY SAVED C FILES	Same as previous.
RETURN FROM fixline ioctl	An ioctl, which should never fail, failed. There is a system driver problem.
BAD SPEED	A bad line speed appears in the **Devices**/**Systems** files (Class field).

PERMISSIONS file: BAD OPTION	There is a bad line or option in the **Permissions** file. Fix it immediately!
PKCGET READ	The remote machine probably hung up. No action need be taken.
PKXSTART	The remote machine aborted in a non-recoverable way. This can generally be ignored.
SYSTAT OPEN FAIL	There is a problem with the modes of **/usr/lib/uucp/.Status**, or there is a file with bad modes in the directory.
TOO MANY LOCKS	There is an internal problem! Contact your AT&T Account Representative or Authorized Dealer.
XMV ERROR	There is a problem with some file or directory. It is likely the spool directory, since the modes of the destinations were suppose to be checked before this process was attempted.
CAN'T FORK	An attempt to fork and exec failed. The current job should not be lost, but will be attempted later (**uuxqt**). No action need be taken.

Status Error Messages

Status error messages are messages that are stored in the **/usr/spool/uucp/.Status** directory. This directory contains a separate file for each remote machine that your 3B2 Computer attempts to communicate with. These individual machine files contain status information on the attempted communication, whether it was successful or not. What follows is a list of the most common error messages that may appear in these files.

OK	Things are OK.
NO DEVICES AVAILABLE	There is currently no device available for the call. Check to see that there is a valid device in the **Devices** file for the particular system. Check the **Systems** file for the device to be used to call the system.
WRONG TIME TO CALL	A call was placed to the system at a time other than what is specified in the **Systems** file.
TALKING	Self explanatory.
LOGIN FAILED	The login for the given machine failed. It could be a wrong login/password, wrong number, a very slow machine, or failure in getting through the Dialer-Token-Pairs script.
CONVERSATION FAILED	The conversation failed after successful startup. This usually means that one side went down, the program aborted, or the line (link) was dropped.

DIAL FAILED	The remote machine never answered. It could be a bad dialer or the wrong phone number.
BAD LOGIN/MACHINE COMBINATION	The machine called us with a login/machine name that does not agree with the **Permissions** file. This could be an attempt to masquerade!
DEVICE LOCKED	The calling device to be used is currently locked and in use by another process.
ASSERT ERROR	An ASSERT error occurred. Check the **/usr/spool/uucp/.Admin/errors** file for the error message and refer to the section *ASSERT Error Messages*.
SYSTEM NOT IN Systems	The system is not in the **Systems** file.
CAN'T ACCESS DEVICE	The device tried does not exist or the modes are wrong. Check the appropriate entries in the **Systems** and **Devices** files.
DEVICE FAILED	The open of the device failed.
WRONG MACHINE NAME	The called machine is reporting a different name than expected.
CALLBACK REQUIRED	The called machine requires that it calls your 3B2 Computer.

REMOTE HAS A LCK FILE FOR ME	The remote site has a LCK file for your 3B2 Computer. They could be trying to call your machine. If they have an older version of Basic Networking, the process that was talking to your machine may have failed leaving the LCK file. If they have the new version of Basic Networking, and they are not communicating with your 3B2 Computer, then the process that has a LCK file is hung.
REMOTE DOES NOT KNOW ME	The remote machine does not have the node name of your 3B2 Computer in its **Systems** file.
REMOTE REJECT AFTER LOGIN	The login used by your 3B2 Computer to login does not agree with what the remote machine was expecting.
REMOTE REJECT, UNKNOWN MESSAGE	The remote machine rejected the communication with your 3B2 Computer for an unknown reason. The remote machine may not be running a standard version of Basic Networking.
STARTUP FAILED	Login succeeded, but initial handshake failed.
CALLER SCRIPT FAILED	This is usually the same as " DIAL FAILED." However, if it occurs often, suspect the caller script in the **Dailers** file. Use **Uutry** to check.

Chapter 7

COMMAND DESCRIPTIONS

PAGE

Chapter 7

COMMAND DESCRIPTIONS

COMMAND SUMMARY

This chapter describes the commands of the Basic Networking Utilities. The commands are grouped into two categories:

- User Commands
- Administrative Commands.

In addition to the command descriptions, the format and options of the commands are discussed. Then, a sample usage of each command is presented to give you an idea of how the command is used.

User commands are invoked during attempts to communicate with remote computers or obtain status information. A summary of these commands is shown in Figure 7-1.

COMMAND	DESCRIPTION
ct	Calls and connects a remote terminal to your 3B2 Computer.
cu	Calls a remote computer and allows you to login on the called computer without logging off the 3B2 Computer.
uucp	Queues a file transfer to the specified pathname on a remote computer.
uuto	Queues a file transfer to the public directory on a remote computer.
uupick	Searches the spool directory for files destined to the person initiating the command.
uux	Sets up a remote command execution to be executed on a specified computer.
uulog	Displays information stored in a specific computers log file.
uustat	Provides you with status information on queued transfers.
uuname	Provides you with the names of the computers which are part of the network.

Figure 7-1. User Commands

Administrative commands are used to perform administrative tasks on the UUCP facility. Some are invoked automatically via **cron** while others may be invoked manually if you are logged in as **root** or **uucp**. A summary of the administrative commands is shown in Figure 7-2.

COMMAND	DESCRIPTION
uucheck	Checks for the presence of the required UUCP files and directories. Also checks for errors in the **Permissions** file.
uucleanup	Removes specific files from the spool directory that are more than a specified number of days old.
Uutry	Invokes the transfer daemon (**uucico**) to call a specified machine with a moderate amount of debugging.

Figure 7-2. Administrative Commands

HOW COMMANDS ARE DESCRIBED

A common format is used to describe each of the Basic Networking commands. The format is as follows:

- **General:** The purpose of the command is defined. This includes any unique or special information about the command.
- **Command Format:** The basic command line format (syntax) is defined and the various arguments and options are discussed.
- **Sample Command:** Example command line entries and responses are presented to show you how to use the command.

In the command format discussions, the following symbology and conventions are used to define the command syntax:

- The basic command is shown in bold type.
  For example: **command**
- Arguments that you must supply to the command are shown in an italic type.
  For example: **command** *argument*
- Command options and arguments that do not have to be supplied are enclosed in brackets [].
  For example: **command** [*arguments*]
- The pipe symbol (|) is used to separate arguments when one of several forms of an argument can be used.
  For example: **command** [*argument1* | *argument2*]

In the sample command discussions, user inputs and 3B2 Computer response examples are shown as follows.

This style of type is used to show system generated responses displayed on your screen.

This style of bold type is used to show inputs entered from your keyboard that are displayed on your screen.

These bracket symbols, < > identify inputs from the keyboard that are not displayed on your screen, such as: <CR> carriage return, <CTRL d> control d, <ESC g> escape g, passwords, and tabs.

This style of italic type is used for notes that provide you with additional information.

Also, in the sample command discussion, the dollar sign ($) is used as the system prompt. The system prompt could be the number symbol (#). In either case, you should not enter the prompt.

USER COMMANDS

ct – Generate a getty Process to a Remote Terminal

General

The **ct** command attempts to connect a remote terminal to your 3B2 Computer by dialing the phone number of the modem attached to the terminal. This modem must be able to automatically answer the call when it is received. When **ct** detects that the call has been answered, it issues a **getty** (login) process and allows the user of the remote terminal to login on the 3B2 Computer. A user at a remote terminal may call your 3B2 Computer, login, and issue a **ct** command to request that the 3B2 Computer hang up the current line and call the remote terminal back.

If **ct** cannot find an available dialer, it indicates that the dialer(s) are busy and asks if it should wait until one becomes available. If you respond yes, it asks how long it should wait for one.

Command Format

The **ct** command has the format

ct [*options*] *telno*

where *telno* is the telephone number of a remote terminal with equal signs (=) for secondary dial tones and dashes (-) for delays.

The **ct** command has the following *options*:

-h	Prevents **ct** from disconnecting the current line and allowing the 3B2 Computer to call the remote terminal back.
-v	Requests that a running narrative of the attempt to contact the remote terminal be displayed on the screen.
-w*n*	Overrides the dialogue where **ct** asks if it should wait for a dialer. *n* is the number of minutes that **ct** should wait for a dialer.
-s*speed*	Selects the transmission rate of the modem attached to the remote terminal where *speed* is expressed in baud (default 1200).
-x*n*	Causes debugging output to be displayed on the screen where *n* is a single digit (0 through 9) indicating the level of debugging; 9 being the highest level of debugging.

Sample Command Usage

If you are logged in on the 3B2 Computer through a local terminal, and you want to connect a remote terminal to your 3B2 Computer, you may enter:

```
$ ct -h -w5 -s1200 9=9323497<CR>
```

which calls the modem connected to 932-3497 using a 1200 baud dialer. If a dialer is not available, the **-w5** option causes **ct** to wait 5 minutes for a dialer to become available before it quits. The **-h** option tells **ct** not to disconnect the local terminal (terminal used to input the command) from the 3B2 Computer.

If you are at a remote terminal some distance away from your 3B2 Computer and you do not want to get billed for long distance charges, you can call the 3B2 Computer initially, login, and enter the following **ct** command line:

```
$ ct -s1200 9=9323497<CR>
```

If no device is available at the time, the following dialogue is received:

```
1 busy dialer at 1200 baud
Wait for dialer?
```

If you reply no (n), the **ct** command exits. If you reply yes (y), you are prompted to indicate how long to wait:

```
Time, in minutes?
```

If a dialer is available, **ct** responds with:

```
Allocated dialer at 1200 baud
```

which indicates that it found a dialer. In any case, **ct** asks if you want the line connecting your remote terminal to the 3B2 Computer to be dropped:

```
Confirm hangup?
```

If you indicate yes (y), you are logged off and **ct** calls the remote terminal back if a dialer was, or becomes, available. If you indicate no (n) the **ct** command exits and you are still logged onto the 3B2 Computer.

cu - Call Another UNIX System

General

The **cu** command calls a remote computer and links your 3B2 Computer to the called computer. The **cu** command can use either direct links, the telephone network, or Local Area Networks (LAN) to establish a connection to a remote computer. Once the connection is made, the called computer will prompt you to login on the computer. After you are logged in on the called computer, you can transfer ASCII coded files from one computer to another and execute commands on either computer.

> ***Note:*** The **cu** command does not have error detection/correction capability. Therefore, it is possible that some loss or corruption of data may occur during file transfers.

Command Format

The **cu** command has the format:

cu [*options*] *telno* | *systemname*

Where:

telno is the telephone number of a remote computer with equal signs (=) for secondary dial tones and dashes (-) for four second delays.

systemname is a UUCP system name that appears in the **Systems** file. In this case, **cu** will obtain the telephone number and baud rate from the **Systems** file and search for a dialer to use for the connection. For this reason, the **-s**, **-n**, and **-l** options should *not* be used in conjunction with *systemname*. The **uuname** command will display the machines listed in the **Systems** file.

The **cu** command has the following *options*:

-s*speed*	Specifies the transmission rate of the device to be used in the connection where *speed* is the transmission rate (usually 300 or 1200); 300 is the default value.
-l*line*	Used if a specific device is to be used in the connection where *line* is the name used to reference a specific line (port) to which the device is connected. This is useful when calling a computer that is hardwired to your 3B2 Computer.
-h	Used to emulate local echo when the called computer expects terminals to be in the half-duplex mode.
-t	Used when dialing an ASCII terminal set up with automatic answer.
-d	Causes diagnostic traces to be printed.
-e (**-o**)	Designates even (odd) parity to be generated and sent to the remote computer.
-n	Indicates that **cu** will prompt you for the telephone number instead of taking it from the command line.

Once the connection is made and you are logged in on the remote computer, any standard (keyboard) input is sent to the remote computer. Figure 7-3 shows the strings that you can execute while you are connected to a remote machine through **cu**.

STRING	INTERPRETATION
~.	Terminates the conversation.
~!	Escape to the local system (3B2 Computer) without dropping the link. To get back to the remote machine, CTRL-d.
~!*cmd*	Execute *cmd* on the local system (3B2 Computer).
~$*cmd*	Execute *cmd* on the 3B2 Computer and send its output to the remote computer.
~%**cd** *path*	Change the directory on the local system where *path* is the pathname or directory name.
~%**take** *from* [*to*]	Copy file named *from* (on the remote computer) to file named *to* on the 3B2 Computer. If *to* is omitted, the *from* argument is used in both places.
~%**put** *from* [*to*]	Copy file named *from* (on 3B2 Computer) to file named *to* on the remote computer. If *to* is omitted, the *from* argument is used in both places.
~~...	Send the line ~... to the remote computer.
~%**break**	Transmits a BREAK to the remote machine (can also be specified as ~%**b**).

Figure 7-3. cu Command Strings (1 of 2)

STRING	INTERPRETATION
~%nostop	Turn off the handshaking protocol for the remainder of the session. This is useful when the remote computer does not respond properly to the protocol characters.
~%debug	Toggles the **-d** debugging option on or off (can also be specified as **~%d**).
~t	Displays the values of the terminal I/O (termio) structure variables for your terminal (useful for debugging).
~l	Displays the values of the termio structure variables for the remote communication line (useful for debugging).

Figure 7-3. cu Command Strings (2 of 2)

Note 1: The use of **~%put** requires **stty** and **cat** on the remote machine. It also requires that the current erase and kill characters on the remote machine be identical to the current ones on the 3B2 Computer.

Note 2: The use of **~%take** requires the existence of **echo** and **cat** on the remote machine. Also, **stty tabs** mode should be set on the remote machine if tabs are to be copied without expansion.

Sample Command Usage

To communicate with the remote computer **eagle**, enter

```
$ cu -s1200 9=847-7867<CR>
```

where **eagle**s phone number is 847-7867. The **-s1200** option forces **cu** to use a 1200 baud dialer to call **eagle**. If the **-s** option is not specified, **cu** uses the default speed, 300 baud. When **eagle** answers the call, messages similar to the following appear on the screen:

```
connected

login:
```

At this point, you should enter your login ID and password.

Suppose you want a copy of a file named **proposal** from the remote computer. If **proposal** is located in the current directory (on remote computer), the following string copies the file **proposal** from the remote computer to the 3B2 Computer and places it in the current directory (on local computer) under a file named **proposal**:

```
$ ~%take proposal<CR>
```

To send a file named **minutes** (in the current directory) on your 3B2 Computer to the remote computer, enter:

```
$ ~%put minutes minutes.9-18<CR>
```

This copies the file **minutes** over to the remote computer and places it in a file named **minutes.9-18**.

uucp - UNIX-to-UNIX System Copy

General

The **uucp** command will queue a file transfer as described in Chapter 2. The **uucp** transfer is one that is transparent to the user. Once the command is entered, you may continue with other processes while the UUCP programs attempt to perform the transfer.

Command Format

The **uucp** command has the format

uucp [*options*] *source-file system!destination-file*

where the *source-file* and *destination-file* may contain the prefix *system-name!*, which indicates the computer where the file resides or where it will be copied. If *system-name* is omitted, the local machine is assumed. The *system-name* argument must be one that UUCP knows about. In other words, it must be listed in the **Systems** file (see **uuname**).

In most cases, the *source-file* and *destination-file* will be a specified path name on a given computer. If a shell metacharacter is used in the path name, it will be expanded on the remote computer. Path names may be one of the following:

- A full path name.
- A path name preceded by ~*user* where *user* is a login on the specified computer and is replaced by that users HOME directory.
- A path name preceded by ~/*destination* where *destination* is appended to **/usr/spool/uucppublic**. This *destination* will be treated as a file name unless more than one file is being transferred, or if the destination is already a directory. To ensure that *destination* is a directory, follow it with a "/" (~/**destination**/).

- Anything else is prefixed by the current directory.

If *destination-file* is a directory, the last part of the *source-file* name is used. The various ways to specify path names are shown in "Sample Commands."

The **uucp** command has the following *options*:

-c	Use the source file when copying out to the remote computer rather than copying the file to the spool directory (default).
-C	Copy the source file to the spool directory.
-d	Make all necessary directories for the file copy (default).
-f	Do not make intermediate directories for the file copy.
-g*grade*	The *grade* is a single letter/number. Lower ASCII sequence characters will cause the job to be transmitted earlier during a particular conversation.
-j	Output the job identification number on the screen. This is an ASCII sequence that UUCP assigns to each job. It can be used with the **uustat** command to obtain status information or terminate a job.
-m	Send mail to the requester when the copy is completed.
-s*file*	Report status of the transfer in *file*. Note that *file* must be a full path name.
-n*user*	Notify *user* on the remote computer that a file was sent.

-r	Queue the job but do not start the transfer daemon (**uucico**) and attempt to transfer the job.
-x*debug_level*	Produces debugging output on the screen where *debug_level* is a number between 0 and 9; higher numbers give more detailed information.

Sample Command Usage

To send the file named **minutes** to the remote computer **eagle**, you could enter:

```
$ uucp -s -C -j -ngws minutes eagle!/usr/gws/minutes<CR>
eagleN3f45
$
```

This sends the file **minutes** (located in current directory on 3B2 Computer) to the computer **eagle** and places it under the path name **/usr/gws** in a file named **minutes**. When the transfer is complete, the user **gws** on the remote computer receives mail indicating that a file has been received. You also receive mail indicating a successful or failed transfer as requested by the **-s** option. The **-j** option requested that the job ID (`eagleN3f45`) be displayed.

The previous example uses a full path name to provide the *destination-file*. There are two other ways the *destination-file* can be specified:

1. The login directory of **gws** can be specified through use of the ~ character as shown below:

   eagle!~gws/minutes

   would be interpreted as

   eagle!/usr/gws/minutes

2. The **uucppublic** area is referenced by a similar use of the prefix ~ preceding the path name. For example:

   eagle!~/gws/minutes

   is interpreted as

   /usr/spool/uucppublic/gws/minutes

uuto - Public UNIX-to-UNIX System Copy

General

The **uuto** command uses the same processes as the **uucp** command when transferring a file to a remote computer. The only difference is that the **uuto** command will always send the file to the public area on the remote computer. The person to whom the file was sent will be notified by mail when the transfer is complete.

Command Format

The **uuto** command has the format

uuto [*options*] *source-file destination*

where the *source-file* can be any of the following:

- A full path name
- A file located in the current directory.

Destination has the format

system!user

where *system* is the name of the remote computer and *user* is the login name of the user to which the file is being sent. When the transfer is complete, the transferred file(s) will be placed under

/usr/spool/uucppublic/receive/*user/my3b2/files*

where *user* is the remote users login name as specified in the **uuto** command and *my3b2* is the name of your 3B2 Computer.

The **uuto** command has the following *options*:

-m Causes mail to be sent to you (sender) when the transfer is complete.

-p Causes the *source-file* to be copied to the spool directory on your 3B2 Computer before transmission to the remote computer.

Sample Command Usage

To send the file **minutes** to user **gws** on the remote computer **eagle**, you can enter:

```
$ uuto -m -p minutes eagle!gws<CR>
$
```

This makes a copy of the file **minutes** (from the current directory) in the spool directory prior to the transfer. After the transfer is complete, you will receive mail indicating the completion of the transfer and the remote user **gws**, receives mail indicating that a file has been received from the remote computer **my3b2**. The file is placed on the remote computer under

/usr/spool/uucppublic/receive/gws/my3b2/minutes

uupick - uuto File Retrieval

General

The **uupick** command is used in conjunction with the **uuto** command. After a file is sent to you via the **uuto** command, that file resides in the public area under a directory that has the same name as your login ID. The **uupick** command will search the public area to see if a directory exists that matches your login ID. If this check is true, **uupick** waits for you to tell it what to do with the file(s) located in that directory.

Command Format

The **uupick** command has the format

uupick [*-s system*]

where the **-s** option tells **uupick** to search for files sent only from the remote computer *system*.

If **uupick** finds a directory that matches your login name, it will respond with

from *sys*: **[file** *file-name*] **[dir** *directory-name*]

at which point you should tell **uupick** what you want to do with the file(s). The permitted options are shown in Figure 7-4.

COMMAND	INTERPRETATION
<CR>	Go to next entry.
d	Delete the entry.
m [*dir*]	Move the entry to the directory *dir*. If *dir* is not a full path name, a destination relative to the current directory is assumed. If *dir* is not specified, the current directory is assumed.
a [*dir*]	Same as **m** except **a** moves all files sent from *sys*.
p	Print the contents of the file.
q	Stop.
!*command*	Escape to the shell and execute *command*.
*	Print a command summary.

Figure 7-4. uupick Options

Sample Command Usage

You are user **gws** on the computer **eagle**. You login on **eagle** and receive mail indicating that you have received a file from the **my3b2** computer. You can then enter the following command line:

```
$ uupick -s my3b2<CR>
```

and wait for **uupick** to prompt you with

```
from my3b2: file minutes?
```

At this point, you can look at the file **minutes** by entering:

```
p<CR>
```

This will displayed the **minutes** on the screen. After the file has been displayed, the question mark reappears and **uupick** waits for further instructions. If you want to save the file move it to your login directory by entering:

```
m $HOME<CR>
4 blocks
$
```

After the file is moved, the number of blocks taken up by the file is displayed, and you are returned to the shell.

uux - UNIX-to-UNIX System Command Execution

General

The **uux** command is used to execute UNIX System command strings on remote computers. The **uux** command will gather files from various computers, execute a command on a specified computer, and send the standard output to a file on a specified computer. The execution of certain commands may be restricted on the remote machine (**Permissions** file). **Uux** will notify you by mail if the requested command was disallowed.

Command Format

The **uux** command has the format

uux [*options*] *command-string*

where *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 *systemname* is interpreted as command arguments. File names may be one of the following:

- A full path name
- Anything prefixed by the current directory (local machine).

The **uux** command has the following *options*:

–	Indicates that the standard input for *command-string* is taken from the standard input of the **uux** command.
-a*name*	Use *name* as the user ID replacing the actual user ID.

-b	Return standard input to the command if the exit status is non-zero.
-c	Do not copy local files to the spool directory for transfer to the remote machine (default).
-C	Force the copy of local files to the spool directory for transfer.
-g*grade*	The grade is a single letter/number. Lower ASCII sequence characters will cause the job to be transmitted earlier during a particular conversation.
-j	Output the job identification number on the screen. This is an ASCII sequence that UUCP assigns to each job. It can be used with the **uustat** command to obtain status information or terminate a job.
-n	Indicates that no notification is to be sent to the remote user.
-p	Same as "-".
-r	Do not start the file transfer daemon (**uucico**), just queue the job.
-s*file*	Report the status of the transfer in *file*.
-x*debug_level*	Produces debugging output on the screen where *debug_level* is a number between 0 and 9; higher numbers give more detailed information.
-z	Send success notification to the local user.

Sample Command Usage

If your 3B2 Computer is hardwired to a larger host computer you can use **uux** to get printouts of files that reside on your 3B2 Computer by entering:

```
$ pr minutes|uux -p host!lp<CR>
$
```

This command line queues the file **minutes** to be printed on the area printer of the computer **host**.

uulog - UUCP Log Inquiry

General

The **uulog** command is used to display the contents of machine log files associated with a specific remote machine. Each machine will have log files for **uucico** and **uuxqt** processes that have been attempted. These log files reside in the **/usr/spool/uucp/.Log** directory.

Command Format

The **uulog** command has the format

uulog [*options*]

where the following *options* are available:

-s*sys*	Indicates to print the contents of the Log files for the remote machine *sys*.
-f*sys*	Performs a **tail -f** on the file transfer log for machine *sys*.

The two options below can be used in conjunction with the two previous options which can only be used one at a time:

-x	Displays the **uuxqt** log file for the specified machine.
-number	Indicates that the **tail** command should be performed with *number* lines.

Sample Command Usage

If you want to print the current log information associated with the remote computer **eagle**, you can enter:

```
$ uulog -seagle<CR>
uucp eagle  (10/11-1:41:47,1798,0) SUCCEEDED (call to eagle )
uucp eagle  (10/11-1:41:54,1798,0) OK (startup)
uucp eagle  (10/11-1:41:56,1798,0) OK (conversation complete contty 51)
uucp eagle  (10/11-2:11:47,1824,0) SUCCEEDED (call to eagle )
uucp eagle  (10/11-2:11:56,1824,0) OK (startup)
uucp eagle  (10/11-2:11:57,1824,0) OK (conversation complete contty 52)
uucp eagle  (10/11-2:41:48,1846,0) SUCCEEDED (call to eagle )
uucp eagle  (10/11-2:41:58,1846,0) OK (startup)
uucp eagle  (10/11-2:42:00,1846,0) OK (conversation complete contty 55)
uucp eagle  (10/11-3:11:47,1872,0) SUCCEEDED (call to eagle )
uucp eagle  (10/11-3:11:58,1872,0) OK (startup)
uucp eagle  (10/11-3:12:00,1872,0) OK (conversation complete contty 55)
$
```

uustat - UUCP Status Inquiry and Job Control

General

The **uustat** command is used to display general status information of queued jobs (transfers or executions). It also gives you a means of controlling jobs that have been queued, monitoring the size of job queues, and the status of the last attempt to contact all machines.

Command Format

The **uustat** command has the format

uustat [*options*]

where only *one* of the following *options* can be requested at a time:

-a List the jobs that are queued.

-m Report the success/failure status of the last attempt to communicate with all machines listed in the **Systems** file.

-p Executes a **ps -flp** for each process ID that has an associated lock (LCK.) file. This displays detailed information on the UUCP jobs that are currently being processed.

-q List the jobs queued for each machine. If a status file exists for a machine, its date, time and status information are displayed. If a number appears in () next to the number of C. or X. files, it is the age in days of the oldest C. or X. file for that machine. The "Retry" field represents the number of hours until the next possible call. The "Count" field is the number of failure attempts. For machines with a moderate amount of outstanding jobs, this could take 30 seconds or more (real-time).

-r*jobid* Rejuvenate the job whose ID is *jobid*. This prevents the cleanup daemon from deleting the job until the jobs modification time reaches the limit imposed by the daemon.

There are two options that can be requested by themselves or together:

-s*sys* Display the status of all UUCP requests for the remote computer *sys*.

-u*user* Display the status of all UUCP requests issued by *user*.

Sample Command Usage

One of the most useful forms of the **uustat** command is used to display the status of queued transfers. Suppose that you requested a file transfer to the remote computer **eagle**, you can display the status of your requested transfer(s) by entering:

```
$ uustat<CR>
eagleN3f67    10/10-4:58    S    wruxg  chp 374  /usr/chp/minutes
$
```

If you decide not to send **minutes** to **eagle**, you can kill the request using the **-k** option by entering:

```
$ uustat -keagleN3f67<CR>
Job: eagleN3f67 successfully killed
$
```

where **eagleN3f67** is the job ID of the requested transfer.

uuname - UUCP Network Node Names

General

The **uuname** command is useful in determining the names of those computers that are part of your network.

Command Format

The **uuname** command has the format

uuname [*-l*]

where *-l* indicates to display only the name of the local computer.

Sample Command Usage

To display the names of all the remote computers that are part of your network, enter:

```
$ uuname<CR>
```

This displays the names of those remote computers you can communicate with using the UUCP facilities.

To display the node name of your 3B2 Computer, enter:

```
$ uuname -l<CR>
```

which would display only the node name of your 3B2 Computer.

ADMINISTRATIVE COMMANDS

uucleanup - UUCP Spool Directory Cleanup

General

The **uucleanup** command is used to clean-up the UUCP spool directory (**/usr/spool/uucp**). This is usually executed automatically by **uudemon.cleanu**, but the command can be invoked manually to clean-up certain files in the spool directory. **Uucleanup** will inform owners of send/receive requests of machines that cannot be reached, return mail that cannot be delivered, and delete or execute **rnews** for rnews type files. Also has the capability to warn users of requests that have been waiting for a given number of days. You must login as **uucp** or **root** to invoke the **uucleanup** command.

Command Format

The **uucleanup** command has the format

/usr/lib/uucp/uucleanup [*options*]

where the following *options* are available:

-C*time*	Any C. (work) files greater or equal to *time* days old will be removed with appropriate information sent to the owner (default 7 days).
-D*time*	Any D. (data) files greater or equal to *time* days old will be removed. An attempt will be made to deliver mail messages and execute rnews when appropriate (default 7 days).
-W*time*	Any C. (work) files equal to *time* days old (default 1 day) will cause a mail message to be sent to the owner warning about the delay in contacting the remote. The message includes the JOBID, and in the case of mail, the mail message.

The administrator may include a message line telling who to call to check the problem (-m option).

-X*time* Any X. (execute) files greater or equal to *time* days old will be removed. The D. files are probably not present (if they were, the X. would have been executed). But if there are D. files, they will be taken care of by D. processing (default 2 days).

-m*string* *String* will be included in the warning message generated by the -W option. The default line is

See your local administrator to locate the problem.

-o*time* Other files whose age is more than *time* days will be deleted (default 2 days).

-s*system* Execute for the directory *system* in the spool directory. The directory *system* is the spool directory for the specific computer.

Sample Command Usage

The following **uucleanup** command line is the one that executes out of uudemon.cleanu:

```
/usr/lib/uucp/uucleanup  -D7 -C7 -X2 -o2 -W1
```

This deletes all " work" and " data" files equal or greater than 7 days old. Any other files (including " execute") are removed if they are equal or greater than two days old. The -W1 option sends a message to the user when the file remains in the spool for one day.

Uutry - Try to Contact a Machine With Debugging On

General

The **Uutry** program is used to invoke **uucico -s***systemname* (with a moderate amount of debugging output) to try contacting the specified machine. The debugging output is placed in /**tmp**/*systemname*, as well as displayed on the screen. A **RUBOUT** or **BREAK** will return the terminal back to the shell while **uucico** continues to run, putting its output in /**tmp**/*systemname*. The minimum retry time for a machine that is busy or does not answer is 5 minutes. The **uustat -m** output will indicate if a machine is busy or does not answer. You cannot attempt a **Uutry** as long as a " retry" exists for the particular system. Notice that **Uutry** is initial cap.

Command Format

The **Uutry** command has the following format:

/usr/lib/uucp/Uutry [*options*] *systemname*

where *systemname* is the name of the remote machine to be called.

The **Uutry** command has the following *options*:

-x*debug_level*	Overrides the default debugging level (5). The *debug_level* is a single digit (0 through 9) number with higher numbers providing more debugging output.
-r	Allows you to input a **Uutry** command before the retry time has expired.

Sample Command Usage

If you have experienced problems in communicating with a remote machine, use the **Uutry** command to aid in the resolution of the problem as shown below:

```
$ /usr/lib/uucp/Uutry eagle<CR>
conn(eagle)
Device Type ACU wanted
expect: ("")
got it
sendthem (DELAY
^M)
expect: (>)
^M^JAT&T ACU/Modem ^M^J1200 BPS^M^J>got it
sendthem (PAUSE
<NO CR>)
expect: (E)
9^M^JSUREgot it
sendthem (<NO CR>)
expect: (:)
? (Y/N)y^M^JNO.:got it
sendthem (ECHO CHECK ON
 3P2P5P1P^M)
expect: (>)
^M^M^J>got it
sendthem (9<NO CR>)
expect: (OK)
^M^JDIALING: 3251^M^JOKgot it
getto ret 5
expect: (in:)
^M^J^J^M^JEAGLE login:got it
sendthem (^M)
expect: (word:)
 ^M^J>nuucp^M^JPassword:got it
sendthem (^M)
Login Successful: System=eagle
wmesg 'U'g
Proto started g
wmesg 'H'
wmesg 'H'Y
send OO 0,Conversation Complete: Status SUCCEEDED
$
```

Additional debugging output can be obtained by using the **-x** option (i.e. **-x9**).

uucheck - Check UUCP Directories and Permissions File

General

The **uucheck** command is used to check for the presence of files and directories required by UUCP. It also checks for obvious errors in the **Permissions** file.

Note: A **root** or **uucp** login is required to execute **uucheck**.

Command Format

The **uucheck** command has the format

/usr/lib/uucp/uucheck [*options*]

where the following **options** are available:

-v	Provides a detailed explanation of how the UUCP programs will interpret the **Permissions** file.
-x*debug_level*	Produces debugging output where *debug_level* is a single digit (0 through 9) number with higher numbers providing more debugging output.

Sample Command Usage

To see a detailed explanation of how the UUCP programs would interpret your **Permissions** file, enter:

```
# /usr/lib/uucp/uucheck -v<CR>
*** uucheck:  Check Required Files and Directories
*** uucheck:  Directories Check Complete

*** uucheck:  Check /usr/lib/uucp/Permissions file
** LOGNAME PHASE (when they call us)

When a system logs in as: (nuucp)
                    We DO allow them to request files.
                    We WILL send files queued for them on this call.
                    They can send files to
                        /
                    They can request files from
                        /
                    Myname for the conversation will be my3b2.
                    PUBDIR for the conversation will be /usr/spool/uucppublic.


** MACHINE PHASE (when we call or execute their uux requests)

When we call system(s): (eagle) (raven) (hawk)
                    We DO NOT allow them to request files.
                    They can send files to
                        /usr/spool/uucppublic (DEFAULT)
                    Myname for the conversation will be my3b2.
                    PUBDIR for the conversation will be /usr/spool/uucppublic.

Machine(s): (eagle) (raven) (hawk)
CAN execute the following commands:
command (rmail), fullname (rmail)
command (lp), fullname (lp)
command (uuxqt), fullname (uuxqt)

*** uucheck:  /usr/lib/uucp/Permissions Check Complete
#
```

Appendix A

MANUAL PAGES

This appendix contains the UNIX System manual pages for the Basic Networking Utilities. Manual pages for the following commands are provided in alphabetical sequence.

ct	cu	uucheck	uucico
uucleanup	uucp	uulog	uuname
uugetty	uusched	uustat	uuto
uupick	Uutry	uux	uuxqt

Certain of these commands are described on a manual page along with one or more other commands. Commands described on a different manual page than indicated by the name of the command are as follows:

COMMAND	MANUAL PAGE
uulog	uucp
uuname	uucp
uupick	uuto

You may arrange the manual pages in this manual to support your specific needs. The yellow sheet at the front of this manual describes your options for filing the manual pages and descriptive information.

NAME

ct — spawn getty to a remote terminal

SYNOPSIS

ct [**−w**n] [**−x**n] [**−h**] [**−v**] [**−s**speed] telno ...

DESCRIPTION

Ct dials the telephone number of a modem that is attached to a terminal, and spawns a *getty* process to that terminal. *Telno* is a telephone number, with equal signs for secondary dial tones and minus signs for delays at appropriate places. (The set of legal characters for *telno* is 0 thru 9, -, =, *, and #. The maximum length *telno* is 31 characters). If more than one telephone number is specified, *ct* will try each in succession until one answers; this is useful for specifying alternate dialing paths.

Ct will try each line listed in the file **/usr/lib/uucp/Devices** until it finds an available line with appropriate attributes or runs out of entries. If there are no free lines, *ct* will ask if it should wait for one, and if so, for how many minutes it should wait before it gives up. *Ct* will continue to try to open the dialers at one-minute intervals until the specified limit is exceeded. The dialogue may be overridden by specifying the **−w***n* option, where *n* is the maximum number of minutes that *ct* is to wait for a line.

The **−x***n* option is used for debugging; it produces a detailed output of the program execution on stderr. The debugging level, *n*, is a single digit; **−x**9 is the most useful value.

Normally, *ct* will hang up the current line, so the line can answer the incoming call. The **−h** option will prevent this action. If the **−v** option is used, *ct* will send a running narrative to the standard error output stream.

The data rate may be set with the **−s** option, where *speed* is expressed in baud. The default rate is 1200.

After the user on the destination terminal logs out, there are two things that could occur depending on what type of getty is on the line (*getty* or *uugetty*). For the first case, *ct* prompts, **Reconnect?** If the response begins with the letter **n**, the line will be dropped; otherwise, *getty* will be started again and the **login:** prompt will be printed. In the second case, there is already a getty (*uugetty*) on the line, so the **login:** message will appear.

Of course, the destination terminal must be attached to a modem that can answer the telephone.

FILES

/usr/lib/uucp/Devices
/usr/adm/ctlog

SEE ALSO

cu(1C), uucp(1C), uugetty(1M).
login(1) in the *3B2 Computer System User Reference Manual.*
getty(1M) in the *3B2 Computer System Administration Utilities Guide.*

BUGS

For a shared port, one used for both dial-in and dial-out, the *uugetty* program running on the line must have the **−r** option specified (see *uugetty*(1M)).

NAME

cu − call another UNIX system

SYNOPSIS

cu [−s*speed*] [−l*line*] [**−h**] [**−t**] [**−d**] [**−o** | **−e**] [**−n**] telno
cu [**−h**] [**−d**] [**−o** | **−e**] systemname

DESCRIPTION

Cu calls up another UNIX system, a terminal, or possibly a non-UNIX system. It manages an interactive conversation with possible transfers of ASCII files.

Cu accepts the following options and arguments:

−s*speed* Specifies the transmission speed (300, 1200, 2400, 4800, 9600); The default value is "Any" speed which will depend on the order of the lines in the **/usr/lib/uucp/Devices** file. Most modems are either 300 or 1200 baud. Directly connected lines may be set to a speed higher than 1200 baud.

−l*line* Specifies a device name to use as the communication line. This can be used to override the search that would otherwise take place for the first available line having the right speed. When the **−l** option is used without the **−s** option, the speed of a line is taken from the Devices file. When the **−l** and **−s** options are both used together, cu will search the Devices file to check if the requested speed for the requested line is available. If so, the connection will be made at the requested speed; otherwise an error message will be printed and the call will not be made. The specified device is generally a directly connected asynchronous line (e.g., **/dev/ttyab**) in which case a telephone number (*telno*) is not required. If the specified device is associated with an auto dialer, a telephone number must be provided. Use of this option with *systemname* rather than *telno* will not give the desired result (see *systemname* below).

−h Emulates local echo, supporting calls to other computer systems which expect terminals to be set to half-duplex mode.

−t Used to dial an ASCII terminal which has been set to auto answer. Appropriate mapping of carriage-return to carriage-return-line-feed pairs is set.

−d Causes diagnostic traces to be printed.

−o Designates that odd parity is to be generated for data sent to the remote system.

−n For added security, will prompt the user to provide the telephone number to be dialed rather than taking it from the command line.

−e Designates that even parity is to be generated for data sent to the remote system.

telno When using an automatic dialer, the argument is the telephone number with equal signs for secondary dial tone or minus signs placed appropriately for delays of 4 seconds.

systemname A uucp system name may be used rather than a telephone number; in this case, *cu* will obtain an appropriate direct line or telephone number from **/usr/lib/uucp/Systems.** Note: the *systemname* option should not be used in conjunction with the **−l** and **−s** options as *cu* will connect to the first available line for the system name specified, ignoring the requested line and speed.

After making the connection, *cu* runs as two processes: the *transmit* process reads data from the standard input and, except for lines beginning with ~, passes it to the remote system; the *receive* process accepts data from the remote system and, except for lines beginning with ~, passes it to the standard output. Normally, an automatic DC3/DC1 protocol is used to control input from the remote so the buffer is not overrun. Lines beginning with ~ have special meanings.

The *transmit* process interprets the following:

~.	terminate the conversation.
~!	escape to an interactive shell on the local system.
~!*cmd*...	run *cmd* on the local system (via **sh -c**).
~$*cmd*...	run *cmd* locally and send its output to the remote system.
~%**cd**	change the directory on the local system. Note: ~!**cd** will cause the command to be run by a sub-shell, probably not what was intended.
~%**take** *from* [*to*]	copy file *from* (on the remote system) to file *to* on the local system. If *to* is omitted, the *from* argument is used in both places.
~%**put** *from* [*to*]	copy file *from* (on local system) to file *to* on remote system. If *to* is omitted, the *from* argument is used in both places. For both ~%**take** and **put** commands, as each block of the file is transferred, consecutive single digits are printed to the terminal.
~~ *line*	send the line ~ *line* to the remote system.
~%**break**	transmit a **BREAK** to the remote system (which can also be specified as ~%**b**).
~%**debug**	toggles the **-d** debugging option on or off (which can also be specified as ~%**d**).
~**t**	prints the values of the termio structure variables for the user's terminal (useful for debugging).
~**l**	prints the values of the termio structure variables for the remote communication line (useful for debugging).
~%**nostop**	toggles between DC3/DC1 input control protocol and no input control. This is useful in case the remote system is one which does not respond properly to the DC3 and DC1 characters.

The *receive* process normally copies data from the remote system to its standard output. A line from the remote that begins with ~> initiates an output diversion to a file. The complete sequence which the program provides is:

~>[>]:*file*
zero or more lines to be written to *file*
~>

Data from the remote is diverted (or appended, if >> is used) to *file* on the local system. The trailing ~> terminates the diversion.

The use of ~%**put** requires *stty*(1) and *cat*(1) on the remote side. It also requires that the current erase and kill characters on the remote system be

identical to these current control characters on the local system. Backslashes are inserted at appropriate places.

The use of ~% **take** requires the existence of *echo*(1) and *cat*(1) on the remote system. Also, *tabs* mode (See *stty(1)*) should be set on the remote system if tabs are to be copied without expansion to spaces.

When *cu* is used on system X to connect to system Y and subsequently used on system Y to connect to system Z, commands on system Y can be executed by using ~~. Executing a tilda command reminds the user of the local system uname. For example, uname can be executed on Z, X, and Y as follows:

```
uname
Z
~[X]!uname
X
~~[Y]!uname
Y
```

In general, ~ causes the command to be executed on the original machine, ~~ causes the command to be executed on the next machine in the chain.

EXAMPLES

To dial a system whose telephone number is 9 201 555 1212 using 1200 baud (where dialtone is expected after the 9):

```
cu -s1200 9=2015551212
```

If the speed is not specified, "Any" is the default value.

To login to a system connected by a direct line:

```
cu -l /dev/ttyXX
```

or

```
cu -l ttyXX
```

To dial a system with the specific line and a specific speed:

```
cu -s1200 -l ttyXX
```

To dial a system using a specific line associated with an auto dialer:

```
cu -l culXX 9=2015551212
```

To use a system name:

```
cu systemname
```

FILES

/usr/lib/uucp/Systems
/usr/lib/uucp/Devices
/usr/spool/locks/LCK..(tty-device)

SEE ALSO

ct(1C), uucp(1C).
cat(1), echo(1), stty(1), uname(1) in the *3B2 Computer System User Reference Manual.*

DIAGNOSTICS

Exit code is zero for normal exit, otherwise, -1.

WARNINGS

Cu does not do any integrity checking on data it transfers.
Data fields with special *cu* characters may not be transmitted properly. Depending on the interconnection hardware, it may be necessary to use a ~. to terminate the conversion even if **stty 0** has been used.

BUGS

There is an artificial slowing of transmission by *cu* during the ~%**put** operation so that loss of data is unlikely.

NAME

uucheck – check the uucp directories and permissions file

SYNOPSIS

/usr/lib/uucp/uucheck [**–v**] [**–x** debug_level]

DESCRIPTION

Uucheck checks for the presence of the *uucp* system required files and directories. Within the *uucp* makefile, it is executed before the installation takes place. It also checks for some obvious errors in the Permissions file (**/usr/lib/uucp/Permissions**). When executed with the **–v** option, it gives a detailed explanation of how the uucp programs will interpret the Permissions file. The **–x** option is used for debugging; it takes a single digit, higher numbers for more detail.

Note that *uucheck* can only be used by the super-user or *uucp*.

FILES

/usr/lib/uucp/Systems
/usr/lib/uucp/Permissions
/usr/lib/uucp/Devices
/usr/lib/uucp/Maxuuscheds
/usr/lib/uucp/Maxuuxqts
/usr/spool/uucp/*
/usr/spool/locks/LCK*
/usr/spool/uucppublic/*

SEE ALSO

uucico(1M), uusched(1M), uucp(1C), uustat(1C), uux(1C).

BUGS

The program does not check file/directory modes or some errors in the Permissions file such as duplicate login or machine name.

NAME

uucico − file transport program for the uucp system

SYNOPSIS

/usr/lib/uucp/uucico [**−r** role_number] [**−x** debug_level]
[**−d** spool_directory] **−s** system_name

DESCRIPTION

Uucico is the file transport program for *uucp* work file transfers. The **−r** option should be specified as the digit 1 for master mode when *uucico* is started by a program or *cron*. *Uux* and *uucp* both queue jobs that will be transferred by *uucico*. It is normally started by the scheduler, *uusched* but can be started manually; this is done for debugging. For example, the shell *Uutry* starts *uucico* with debugging turned on. A single digit must be used for the **−x** option with higher numbers for more debugging.

FILES

/usr/lib/uucp/Systems
/usr/lib/uucp/Permissions
/usr/lib/uucp/Devices
/usr/lib/uucp/Maxuuxqts
/usr/lib/uucp/Maxuuscheds
/usr/spool/uucp/*
/usr/spool/locks/LCK*
/usr/spool/uucppublic/*

SEE ALSO

uusched(1M), uutry(1M), uucp(1C), uustat(1C), uux(1C).
cron(1M) in the *3B2 Computer System Administration Utilities Guide*.

NAME

uucleanup − uucp spool directory clean-up

SYNOPSIS

/usr/lib/uucp/uucleanup [options]

DESCRIPTION

Uucleanup will scan the spool directories for old files and take appropriate action to remove them in a useful way: Inform the requestor of send/receive requests for systems that can not be reached. Return mail, which cannot be delivered, to the sender. Delete or execute rnews for rnews type files (depending on where the news originated−−locally or remotely). Remove all other files. In addition, there is provision to warn users of requests that have been waiting for a given number of days (default 1). Note that *uucleanup* will process as if all option *times* were specified to the default values unless *time* is specifically set.

The following options are available.

−C*time* Any **C.** files greater or equal to *time* days old will be removed with appropriate information to the requestor. (default 7 days)

−D*time* Any **D.** files greater or equal to *time* days old will be removed. An attempt will be made to deliver mail messages and execute rnews when appropriate. (default 7 days)

−W*time* Any **C.** files equal to *time* days old will cause a mail message to be sent to the requestor warning about the delay in contacting the remote. The message includes the *JOBID*, and in the case of mail, the mail message. The administrator may include a message line telling whom to call to check the problem (**−m** option). (default 1 day)

−X*time* Any **X.** files greater or equal to *time* days old will be removed. The **D.** files are probably not present (if they were, the **X.** could get executed). But if there are **D.** files, they will be taken care of by D. processing. (default 2 days)

−m*string* This line will be included in the warning message generated by the **−W** option.

−o*time* Other files whose age is more than *time* days will be deleted. (default 2 days)

−s*system* Execute for *system* spool directory only.

This program is typically started by the shell *uudemon.cleanup*, which should be started by *cron*(1M).

FILES

/usr/lib/uucp	directory with commands used by *uucleanup* internally
/usr/spool/uucp	spool directory

SEE ALSO

uucp(1C), uux(1C).
cron(1M) in the *3B2 Computer System Administration Utilities Guide.*

NAME

uucp, uulog, uuname – UNIX-to-UNIX system copy

SYNOPSIS

uucp [options] source-files destination-file
uulog [options] **–s**system
uulog [options] system
uulog [options] **–f**system
uuname [**–l**]

DESCRIPTION

Uucp

Uucp copies files named by the *source-file* arguments to the *destination-file* argument. A file name may be a path name on your machine, or may have the form:

system-name!path-name

where *system-name* is taken from a list of system names that *uucp* knows about. The *system-name* may also be a list of names such as

system-name!system-name!...!system-name!path-name

in which case an attempt is made to send the file via the specified route, to the destination. See WARNINGS and BUGS below for restrictions. Care should be taken to ensure that intermediate nodes in the route are willing to foward information (see **WARNINGS** below for restrictions).

The shell metacharacters **?**, ***** and **[...]** appearing in *path-name* will be expanded on the appropriate system.

Path names may be one of:

(1) a full path name;

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

(3) a path name preceded by *~/destination* where *destination* is appended to **/usr/spool/uucppublic**; (NOTE: This destination will be treated as a file name unless more than one file is being transfered by this request or the destination is already a directory. To ensure that it is a directory, follow the destination with a '/'. For example ~/dan/ as the destination will make the directory /usr/spool/uucppublic/dan if it does not exist and put the requested file(s) in that directory).

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

If the result is an erroneous path name for the remote system the copy will fail. If the *destination-file* is a directory, the last part of the *source-file* name is used.

Uucp preserves execute permissions across the transmission and gives 0666 read and write permissions (see *chmod*(2)).

The following options are interpreted by *uucp*:

–c Do not copy local file to the spool directory for transfer to the remote machine (default).

–C Force the copy of local files to the spool directory for transfer.

–d Make all necessary directories for the file copy (default).

–f Do not make intermediate directories for the file copy.

−g*grade* *Grade* is a single letter/number; lower ascii sequence characters will cause the job to be transmitted earlier during a particular conversation.

−j Output the job identification ASCII string on the standard output. This job identification can be used by *uustat* to obtain the status or terminate a job.

−m Send mail to the requester when the copy is completed.

−s*file* Report status of the transfer to *file*. Note that the *file* must be a full path name.

−n*user* Notify *user* on the remote system that a file was sent.

−r Do not start the file transfer, just queue the job.

−x*debug_level*
Produce debugging output on standard output. The *debug_level* is a number between 0 and 9; higher numbers give more detailed information.

Uulog

Uulog queries a log file of *uucp* or *uuxqt* transactions in a file **/usr/spool/uucp/.Log/uucico/***system*, or **/usr/spool/uucp/.Log/uuxqt/***system*.

The options cause *uulog* to print logging information:

−s*sys* Print information about file transfer work involving system *sys*.

−f*system* Does a "tail **−f**" of the file transfer log for *system*. Other options used in conjunction with the above:

−x Look in the *uuxqt* log file for the given system.

−*number* Indicates that a "tail" command of *number* lines should be executed.

Uuname

Uuname lists the *uucp* names of known systems. The **−l** option returns the local system name.

FILES

/usr/spool/uucp	spool directories
/usr/spool/uucppublic/*	public directory for receiving and sending (/usr/spool/uucppublic)
/usr/lib/uucp/*	other data and program files

SEE ALSO

uustat(1C), uux(1C), uuxqt(1M).
mail(1) in the *3B2 Computer System User Reference Manual.*
chmod(2) in the *3B2 Computer System Programmer Reference Manual.*

WARNINGS

The domain of remotely accessible files can (and for obvious security reasons, usually should) be severely restricted. You will very likely not be able to fetch files by path name; ask a responsible person on the remote system to send them to you. For the same reasons you will probably not be able to send files to arbitrary path names. As distributed, the remotely accessible files are those whose names begin **/usr/spool/uucppublic** (equivalent to ~/).

All files received by *uucp* will be owned by *uucp*.
The **−m** option will only work sending files or receiving a single file. Receiving multiple files specified by special shell characters **? * [...]** will not activate the **−m** option.

The forwarding of files through other systems may not be compatible with the previous version of *uucp*. If forwarding is used, all systems in the route must have the same version of *uucp*.

BUGS

Protected files and files that are in protected directories that are owned by the requestor can be sent by *uucp*. However, if the requestor is root, and the directory is not searchable by "other" or the file is not readable by "other", the request will fail.

NAME

uugetty – set terminal type, modes, speed, and line discipline

SYNOPSIS

/usr/lib/uucp/uugetty [**–h**] [**–t** timeout] [**–r**] line
[speed [type [linedisc]]]
/usr/lib/uucp/uugetty **–c** file

DESCRIPTION

Uugetty is identical to *getty*(1M) but changes have been made to support using the line for *uucico*, *cu*, and *ct*; that is, the line can be used in both directions. The *uugetty* will allow users to login, but if the line is free, *uucico*, *cu*, or *ct* can use it for dialing out. The implementation depends on the fact that *uucico*, *cu*, and *ct* create lock files when devices are used. When the "open()" returns (or the first character is read when **–r** option is used), the status of the lock file indicates whether the line is being used by *uucico*, *cu*, *ct*, or someone trying to login. Note that in the **–r** case, several <carriage-return> characters may be required before the login message is output. The human users will be able to handle this slight inconvenience. *Uucico* trying to login will have to be told by using the following login script:

"" \r\d\r\d\r\d\r in:--in: . . .

where the . . . is whatever would normally be used for the login sequence.

Here is an **/etc/inittab** entry using *uugetty* on an 801/212 dialer:

30:2:respawn:/usr/lib/uucp/uugetty -t 60 cul04 1200

The line name, cul04, is the name that appears in the **/usr/lib/uucp/Devices** file for the 212 dialer.

An entry for an intelligent modem or direct line that has a *uugetty* on each end must use the **–r** option. (This causes *uugetty* to wait to read a character before it puts out the login message, thus preventing two uugettys from looping.) If there is a *uugetty* on one end of a direct line, there must be a *uugetty* on the other end as well. Here is an **/etc/inittab** entry using *uugetty* on an intelligent modem or direct line:

30:2:respawn:/usr/lib/uucp/uugetty -r -t 60 tty12 1200

FILES

/etc/gettydefs
/etc/issue

SEE ALSO

uucico(1M), ct(1C), cu(1C).
getty(1M), init(1M), tty(7) in the *3B2 Computer System Administration Utilities Guide.*
login(1) in the *3B2 Computer System User Reference Manual.*
ioctl(2), gettydefs(4), inittab(4) in the *3B2 Computer System Programmer Reference Manual.*

BUGS

Ct will not work when uugetty is used with an intelligent modem such as penril or ventel.

NAME

uusched − the scheduler for the uucp file transport program

SYNOPSIS

/usr/lib/uucp/uusched [**−x** debug_level] [**−u** debug_level]

DESCRIPTION

Uusched is the *uucp* file transport scheduler. It is usually started by the daemon *uudemon.hour* that is started by *cron*;

39 * * * * /bin/su uucp -c "/usr/lib/uucp/uudemon.hour > /dev/null"

The two options are for debugging purposes only; **−x** *debug_level* will output debugging messages from *uusched* and **−u** *debug_level* will be passed as **−x** *debug_level* to *uucico*. The *debug_level* is a number between 0 and 9; higher numbers give more detailed information.

FILES

/usr/lib/uucp/Systems
/usr/lib/uucp/Permissions
/usr/lib/uucp/Devices
/usr/spool/uucp/*
/usr/spool/locks/LCK*
/usr/spool/uucppublic/*

SEE ALSO

uucico(1M), uucp(1C), uustat(1C), uux(1C).
cron(1M) in the *3B2 Computer System Administration Utilities Guide.*

NAME

uuto, uupick – public UNIX-to-UNIX system file copy

SYNOPSIS

uuto [options] source-files destination
uupick [**–s** system]

DESCRIPTION

Uuto sends *source-files* to *destination*. *Uuto* uses the *uucp*(1C) facility to send files, while it allows the local system to control the file access. A source-file name is a path name on your machine. Destination has the form:

system!*user*

where *system* is taken from a list of system names that *uucp* knows about (see *uuname*). *User* is the login name of someone on the specified system.

Two *options* are available:

–p Copy the source file into the spool directory before transmission.
–m Send mail to the sender when the copy is complete.

The files (or sub-trees if directories are specified) are sent to PUBDIR on *system*, where PUBDIR is a public directory defined in the *uucp* source. Specifically the files are sent to

PUBDIR/receive/*user*/*mysystem*/files.

The destined recipient is notified by *mail*(1) of the arrival of files.

Uupick accepts or rejects the files transmitted to the user. Specifically, *uupick* searches PUBDIR for files destined for the user. For each entry (file or directory) found, the following message is printed on the standard output:

from *system*: [file *file-name*] [dir *dirname*] **?**

Uupick then reads a line from the standard input to determine the disposition of the file:

<new-line>	Go on to next entry.
d	Delete the entry.
m [*dir*]	Move the entry to named directory *dir*. If *dir* is not specified as a complete path name (in which $HOME is legitimate), a destination relative to the current directory is assumed. If no destination is given, the default is the current directory.
a [*dir*]	Same as **m** except moving all the files sent from *system*.
p	Print the content of the file.
q	Stop.
EOT (control-d)	Same as **q**.
!*command*	Escape to the shell to do *command*.
*	Print a command summary.

Uupick invoked with the **-s***system* option will only search the PUBDIR for files sent from *system*.

FILES

PUBDIR/usr/spool/uucppublic public directory

SEE ALSO

uucleanup(1M), uucp(1C), uustat(1C), uux(1C).
mail(1) in the *3B2 Computer System User Reference Manual.*

WARNINGS

In order to send files that begin with a dot (e.g., .profile) the files must by qualified with a dot. For example: .profile, .prof*, .profil? are correct; whereas *prof*, ?profile are incorrect.

NAME

Uutry − try to contact remote system with debugging on

SYNOPSIS

Uutry [**−x** debug_level] [**−r**] system_name

DESCRIPTION

Uutry is a shell that is used to invoke *uucico* to call a remote site. Debugging is turned on (default is level 5); **−x** will override that value. The **−r** overrides the retry time in **/usr/spool/uucp/.status**. The debugging output is put in file **/tmp/***system_name*. A tail **−f** of the output is executed. A <RUBOUT> or <BREAK> will give control back to the terminal while the *uucico* continues to run, putting its output in **/tmp/***system_name*.

FILES

/usr/lib/uucp/Systems
/usr/lib/uucp/Permissions
/usr/lib/uucp/Devices
/usr/lib/uucp/Maxuuxqts
/usr/lib/uucp/Maxuuscheds
/usr/spool/uucp/*
/usr/spool/locks/LCK*
/usr/spool/uucppublic/*
/tmp/system_name

SEE ALSO

uucico(1M), uucp(1C), uux(1C).

NAME

uux – UNIX-to-UNIX system command execution

SYNOPSIS

uux [options] command-string

DESCRIPTION

Uux will gather zero or more files from various systems, execute a command on a specified system and then send standard output to a file on a specified system. Note that, for security reasons, many installations will limit the list of commands executable on behalf of an incoming request from *uux*. Many sites will permit little more than the receipt of mail (see *mail*(1)) via *uux*.

The *command-string* is made up of one or more arguments that look like a shell command line, except that the command and file names may be prefixed by *system-name*!. A null *system-name* is interpreted as the local system.

File names may be one of

(1) a full path name;

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

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

As an example, the command

uux "!diff usg!/usr/dan/file1 pwba!/a4/dan/file2 > !~/dan/file.diff"

will get the *file1* and *file2* files from the "usg" and "pwba" machines, execute a *diff*(1) command and put the results in *file.diff* in the local PUBDIR/dan/ directory.

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

Uux will attempt to get all files to the execution system. For files that are output files, the file name must be escaped using parentheses. For example, the command

uux a!cut -f1 b!/usr/file \(c!/usr/file\)

get **/usr/***file* **from system "b" and send it to system "a", perform a** *cut* **command on that file and send the result of the** *cut* command to system "c".

Uux will notify you if the requested command on the remote system was disallowed. This notification can be turned off by the **–n** option. The response comes by remote mail from the remote machine.

The following *options* are interpreted by *uux*:

– The standard input to *uux* is made the standard input to the *command-string*.

–a*name* Use *name* as the user identification replacing the initiator user-id. (Notification will be returned to the user.)

–b Return standard input to the command if the exit status is non-zero.

–c Do not copy local file to the spool directory for transfer to the remote machine (default).

–C Force the copy of local files to the spool directory for transfer.

–g*grade* *Grade* is a single letter/number; lower ASCII sequence characters will cause the job to be transmitted earlier during a particular conversation.

−j Output the jobid ASCII string on the standard output which is the job identification. This job identification can be used by *uustat* to obtain the status or terminate a job.

−n Do not notify the user if the command fails.

−p Same as −: The standard input to *uux* is made the standard input to the *command-string*.

−r Do not start the file transfer, just queue the job.

−s*file* Report status of the transfer in *file*.

−x*debug_level*
Produce debugging output on the standard output. The *debug_level* is a number between 0 and 9; higher numbers give more detailed information.

−z Send success notification to the user.

FILES

/usr/spool/uucp/*	spool directories
/usr/lib/uucp/*	other data and programs

SEE ALSO

uucp(1C), uustat(1C).
cut(1), mail(1) in the *3B2 Computer System User Reference Manual.*

WARNINGS

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

The execution of commands on remote systems takes place in an execution directory known to the *uucp* system. All files required for the execution will be put into this directory unless they already reside on that machine. Therefore, the simple file name (without path or machine reference) must be unique within the *uux* request. The following command will NOT work:

```
uux "a!diff b!/usr/dan/xyz c!/usr/dan/xyz > !xyz.diff"
```

but the command

```
uux "a!diff a!/usr/dan/xyz c!/usr/dan/xyz > !xyz.diff"
```

will work. (If *diff* is a permitted command.)

BUGS

Protected files and files that are in protected directories that are owned by the requestor can be sent in commands using *uux*. However, if the requestor is root, and the directory is not searchable by "other", the request will fail.

NAME

uuxqt − execute remote command requests

SYNOPSIS

/usr/lib/uucp/uuxqt [**−s** system] [**−x** debug_level]

DESCRIPTION

Uuxqt is the program that executes remote job requests from remote systems generated by the use of the *uux* command. (*Mail* uses *uux* for remote mail requests). *Uuxqt* searches the spool directories looking for *X.* files. For each *X.* file, *uuxqt* checks to see if all the required data files are available and accessible, and file commands are permitted for the requesting system. The *Permissions* file is used to validate file accessibility and command execution permission.

There are two environment variables that are set before the *uuxqt* command is executed:
UU_MACHINE is the machine that sent the job (the previous one).
UU_USER is the user that sent the job.
These can be used in writing commands that remote systems can execute to provide information, auditing, or restrictions.

FILES

/usr/lib/uucp/Permissions
/usr/lib/uucp/Maxuuxqts
/usr/spool/uucp/*
/usr/spool/locks/LCK*

SEE ALSO

uucico(1M), uucp(1C), uustat(1C), uux(1C).
mail(1) in the *3B2 Computer System User Reference Manual.*

Index

A

B

C

D

E

F

H

I

L

M

N

P

Q

R

S

T

U

V

W

X