TCP/IP Network User's Guide

SG–2009 9.0

Cray Research, Inc.

Portions of this product may still be in development. The existence of those portions still in development is not a commitment of actual release or support by Cray Research, Inc. Cray Research, Inc. assumes no liability for any damages resulting from attempts to use any functionality or documentation not officially released and supported. If it is released, the final form and the time of official release and start of support is at the discretion of Cray Research, Inc.

Requests for copies of Cray Research, Inc. publications should be sent to the following address:

Cray Research, Inc.
Distribution Center
2360 Pilot Knob Road
Mendota Heights, MN 55120
USA

Order desk +1–612–683–5907
Fax number +1–612–452–0141

New Features

This printing of the *TCP/IP Network User's Guide* supports the UNICOS 9.0 release.

New features are as follows:

- All references to the OSI product have been removed in this release.
- The `ftp` command is updated in section 4.
- The section on TCP/IP Network Security has been updated.

Record of Revision

The date of printing or software version number is indicated in the footer. Changes in rewrites are noted by revision bars along the margin of the page.

Version	*Description*
	March 1986. Original printing. Documentation to support the 1.0 release of the Cray operating system UNICOS.
A	October 1986. Reprint with revision. Documentation to support the UNICOS 2.0 release. All trademarks are now located in the record of revision. The remote login utility `rlogin` is deferred for CRAY-2 systems and the utilities *`rwho`* and *`ruptime`* are deferred for all Cray computer systems. They are footnoted only at their first appearance in their respective descriptive subsections. All technical changes, additions, and deletions of text are marked by change bars. Conventions changes and editorial changes are not marked. For new conventions, see "Conventions" in Preface, page v. Section 2 has been incorporated into section 1.
B	July 1987. Reprint with revision. Documentation to support the UNICOS 3.0 release. This document has been reorganized and rewritten for clarity and consistency of terminology. All technical changes, additions, and deletions of text are marked by change bars. Convention changes and editorial changes are not marked. Instructions for the `hosts`, `rhosts`, and `hosts.equiv` files have been put into a new section by themselves (section 2) and subsequent sections have been renumbered. The section on network mail has been expanded, with a fuller explanation of options on different systems and with instructions about the `mailx` start-up files. Other additions include new examples for the `remsh` command, a new convention for the `rcp` command, more options for the `finger` command, and documentation for the new `ping` command. The `rwho` and `ruptime` commands have been deleted.
C	June 1988. Reprint with revision. Documentation to support the UNICOS 4.0 release for the CRAY Y-MP, CRAY X-MP, and CRAY-2 systems. This manual has been updated to conform to the new UNICOS command conventions, as explained in the Preface. Editorial and technical corrections have been made.

D January 1989. Rewrite. Documentation to support the UNICOS 5.0 release. The following commands were added to the `telnet`(1) program: `crmod`, `debug`, `negotiate`, and `options`. The `ftp`(1) program has several new commands: `!`, `$`, `account`, `cdup`, `cr`, `disconnect`, `form`, `macdef`, `mode`, `nmap`, `ntrans`, `proxy`, `reset`, `runique`, `struct`, `sunique`, `tenex`, and `trace`. CRAY-2 systems now fully support the `rlogin`(1) program. The `rcp`(1) program has both a new command syntax and a new option `-p`. The `ntalk`(1BSD) command was added.

6.0 December 1990. This manual has been modified to include a description of the OSI protocol, and it supports release 6.0 of the UNICOS operating system.

7.0 April 1992. This reprint with revision supports the UNICOS 7.0 release. The `ntalk` command and the `@rhosts` parameter on the `finger` command were deleted. The OSI `rft` utility and a description of the mandatory security policy were added.

8.0 December 1993. This rewrite supports the UNICOS 8.0 release.

9.0 June 1995. This rewrite supports the UNICOS 9.0 release. All references to the OSI product have been removed.

Preface

This manual introduces network users to the communication capabilities available with Transmission Control Protocol/Internet Protocol (TCP/IP) on Cray Research systems running the 9.0 release of the UNICOS operating system. TCP/IP is available on all Cray Research systems that run the UNICOS operating system.

This manual describes the UNICOS networking commands that you can execute at a Cray Research system to communicate with a remote system. If you want to communicate with the Cray Research system from a remote system, you must learn the networking procedures for that system; before proceeding in this manual, review the set of manuals that describes these procedures. This manual also includes information about how the UNICOS multilevel security (MLS) and Trusted UNICOS systems impact the use of TCP/IP on Cray Research systems.

> **Note:** The Trusted UNICOS system is a configuration of the UNICOS MLS system that supports processing at multiple security labels and system administration using only non-super user administrative roles. The Trusted UNICOS system consists of the subset of UNICOS software that offers these capabilities. The Trusted UNICOS name does not imply maintenance of the UNICOS 8.0.2 security evaluation.

The UNICOS TCP/IP commands are very similar to the equivalent UNIX commands, because UNICOS TCP/IP builds on the 4.3 release of the Fourth Berkeley Software Distribution (4.3BSD).

Related publications

The following publications provide further information about the UNICOS system and TCP/IP. Unless otherwise noted, all publications referred to in this manual are Cray Research publications.

- *UNICOS Text Editors Primer,* publication SG–2050
- *UNICOS Command Conventions*, publication CP–2058

- *UNICOS `vi` Reference Card,* publication SQ–2054
- *UNICOS `ed` Reference Card,* publication SQ–2055
- *UNICOS User Commands Ready Reference,* publication SQ–2056
- *UNICOS User Commands Reference Manual,* publication SR–2011
- *UNICOS Multilevel Security (MLS) Feature User's Guide,* publication SG–2111
- *UNICOS Networking Facilities Administrator's Guide,* publication SG–2304
- *Kerberos User's Guide,* publication SG–2409

The *User Publications Catalog*, publication CP–0099, describes the availability and content of all Cray Research hardware and software manuals that are available to customers.

To order a manual, either call the Distribution Center in Mendota Heights, Minnesota, at +1–612–683–5907 or send a facsimile of your request to fax number +1–612–452–0141. Cray Research employees may send electronic mail to `orderdsk` (UNIX system users).

Conventions

The following conventions are used throughout this manual:

Convention	Meaning
`command`	This fixed-space font denotes literal items such as commands, files, routines, path names, signals, messages, and programming language structures.
`manpage(`*x*`)`	Man page section identifiers appear in parentheses after man page names. The following list describes the identifiers: 1 User commands 1B User commands ported from BSD 2 System calls 3 Library routines, macros, and opdefs 4 Devices (special files) 4P Protocols 5 File formats 7 Miscellaneous topics 7D DWB-related information 8 Administrator commands
variable	Italic typeface denotes variable entries and words or concepts being defined.
`user input`	This bold fixed-space font denotes literal items that the user enters in interactive sessions. Output is shown in nonbold, fixed-space font.
[ ]	Brackets enclose optional portions of a command line.
...	Ellipses indicate that a preceding command-line element can be repeated.
KEY	This convention indicates a key on the keyboard.
<KEY>	On man pages, this convention indicates a key on the keyboard.

The following machine naming conventions may be used throughout this manual:

Term	Definition
Cray PVP systems	All configurations of Cray parallel vector processing (PVP) systems, including the following: CRAY C90 series (CRAY C916, CRAY C92A, CRAY C94, CRAY C94A, and CRAY C98 systems) CRAY C90D series (CRAY C92AD, CRAY C94D, and CRAY C98D systems) CRAY EL series (CRAY Y-MP EL, CRAY EL92, CRAY EL94, and CRAY EL98 systems) CRAY J90 series (CRAY J916 and CRAY J932 systems) CRAY T90 series (CRAY T94, CRAY T916, and CRAY T932 systems) CRAY Y-MP E series (CRAY Y-MP 2E, CRAY Y-MP 4E, CRAY Y-MP 8E, and CRAY Y-MP 8I systems) CRAY Y-MP M90 series (CRAY Y-MP M92, CRAY Y-MP M94, and CRAY Y-MP M98 systems)
Cray MPP systems	All configurations of Cray massively parallel processing (MPP) systems, including the CRAY T3D series (CRAY T3D MC, CRAY T3D MCA, and CRAY T3D SC systems)
All Cray Research systems	All configurations of Cray PVP and Cray MPP systems that support this release
SPARC systems	All SPARC platforms that run the Solaris operating system version 2.3 or later

The default shell in the UNICOS 9.0 release, referred to in Cray Research documentation as the standard shell, is a version of the Korn shell that conforms to the following standards:

- Institute of Electrical and Electronics Engineers (IEEE) Portable Operating System Interface (POSIX) Standard 1003.2–1992
- X/Open Company Standard XPG4

The UNICOS 9.0 operating system also supports the optional use of the C shell.

The POSIX standard uses *utilities* to refer to executable programs that Cray Research documentation usually refers to as *commands*. Both terms appear in this document.

In this publication, *Cray Research*, *Cray,* and *CRI* refer to Cray Research, Inc. and/or its products.

Online information

The following types of online information products are available to Cray Research customers:

- CrayDoc online documentation reader, which lets you see the text and graphics of a manual online. The CrayDoc reader is available on workstations. To start the CrayDoc reader at your workstation, use the `cdoc`(1) command.
- Docview text-viewer system, which lets you see the text of a manual online. The Docview system is available on the Cray Research mainframe. To start the Docview system, use the `docview`(1) command.
- Man pages, which describe a particular element of the UNICOS operating system or a compatible product. To see a detailed description of a particular command or routine, use the `man`(1) command.
- UNICOS message system, which provides explanations of error messages. To see an explanation of a message, use the `explain`(1) command.
- Cray Research online glossary, which explains the terms used in a manual. To get a definition, use the `define`(1) command.

- `xhelp` help facility. This online help system is available within tools such as the Program Browser (`xbrowse`) and the MPP Apprentice tool.

For detailed information on these topics, see the *User's Guide to Online Information*, publication SG–2143.

Reader comments

If you have comments about the technical accuracy, content, or organization of this manual, please tell us. You can contact us in any of the following ways:

- Send us electronic mail from a UNICOS or UNIX system, using the following UUCP address:

  `uunet!cray!publications`

- Send us electronic mail from any system connected to Internet, using the following Internet addresses:

  `pubs2009@timbuk.cray.com` (comments on this manual)

  `publications@timbuk.cray.com` (general comments)

- Contact your Cray Research representative and ask that a Software Problem Report (SPR) be filed. Use `PUBLICATIONS` for the group name, `PUBS` for the command, and `NO-LICENSE` for the release name.

- Call our Software Publications Group in Eagan, Minnesota, through the Technical Support Center, using either of the following numbers:

  1–800–950–2729 (toll free from the United States and Canada)

  +1–612–683–5600

- Send a facsimile of your comments to the attention of "Software Publications Group" in Eagan, Minnesota, at fax number +1–612–683–5599.

- Use the postage-paid Reader's Comment Form at the back of the printed manual.

We value your comments and will respond to them promptly.

Contents

- You can use the resources of a different computer.

  Adam uses a very powerful workstation and sophisticated software to design a rocket nose cone on his screen and to test his design.

  Adam's problem is that, even though his workstation is very powerful, it is not fast enough to run the solids-modeling software. However, because Adam's workstation is part of a computer network, he can execute his program on a Cray Research supercomputer that is attached to his network. He can access the Cray Research system with one or two simple commands and execute the program as if he were directly connected to the Cray Research system.

- You can send messages to other people on the network.

  After reviewing Adam's nose cone design, Jenny wrote a report recommending that the company implement Adam's design. She used her computer text editor to write the report, and then she sent copies electronically to other people who were either in the same building or in one of the regional offices. Everyone on her distribution list received a copy in a matter of minutes.

Forms of computer networks
1.1.2

A computer network is somewhat invisible (or *transparent*) to the people using it. For example, when Adam executes a command on a Cray Research system, he is aware of only two pieces of equipment: his workstation and the Cray Research system on which he executes the program. Adam perceives his network as an indistinct object; its physical components are unimportant to him.

In reality, a computer network is very complex because of the number and types of computers linked together, the various network media used to connect the computers, and the geographic distance between computers. Figure 1 illustrates what Adam's network might actually look like.

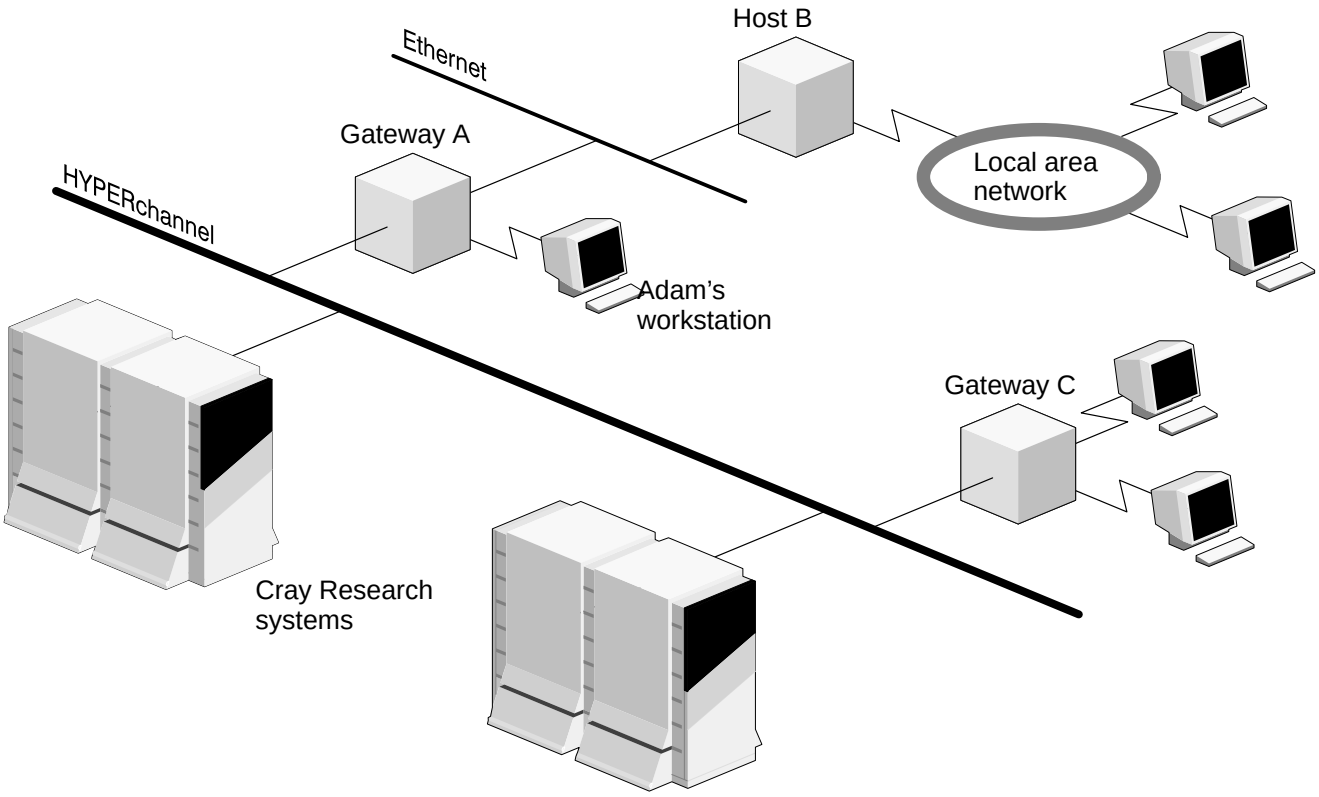


Figure 1. A map of a network

The following list explains how the composition of a network affects its functions.

- A computer network is composed of at least two, and possibly hundreds, of computers and peripheral devices (such as terminals, printers, and fileservers). Each item that can connect to the network is called a *node*.

  Each computer on a network is called a *host*. The computer from which you originate a networking command is called your *local host*. The other computers on the network are called *remote hosts*.

  A *gateway* is a computer that has connections to more than one network, enabling it to accept data from one network and transmit it to another network. (A gateway also can be a type of network hardware called a *router*.)

For example, Figure 1 shows that Adam's workstation gains access to a Cray Research system through gateway A. If this gateway were to become nonfunctional, Adam could not access the Cray Research system.

- Network media form the physical link between computers.

  The physical connection to a Cray Research system is made with one of the following products:

  – Network Systems Corporation (NSC) HYPERchannel

  – FEI-3 network interface provided by Cray Research

  – High-speed External (HSX) Communications Channel provided by Cray Research

  – High Performance Parallel Interface (HIPPI) Channel provided by Cray Research

  – FCA-1 Fiber Distributed Data Interface (FDDI) adapter provided by Cray Research

  – Fiber Distributed Data Interface (FDDI) and/or Ethernet available on CRAY EL systems

  All of these products provide a connection to a Cray Research system. Each medium varies in speed and reliability, which in turn affects your communication.

- A general method of classifying networks is by the geographic distance between connected computer systems.

  Most networks fit into one of the following categories:

  – Local area network (LAN)

  – Wide area network (WAN) (also called *long haul network*)

A LAN consists of computer systems that are located relatively close together, such as in one building or on a campus. For instance, the computer systems linked together in Adam's office building compose a LAN.

A WAN is a network that connects computer systems located over a large geographic area.

Moreover, a LAN can be connected to a WAN, opening doors for even broader communication throughout the state, across the country, or around the world.

Benefits of TCP/IP

1.2

Besides the physical connection that allows computers on a network to communicate, there must be a medium for specifying communication protocols, or rules, that allow hosts to communicate with one another over the physical path. Some communications standard for these protocols must be available to enable two hosts to communicate effectively. The TCP/IP standard that the UNICOS system supports is described in the following subsections.

TCP/IP is made up of two components: Transmission Control Protocol and Internet Protocol. Figure 2 shows how TCP/IP interfaces with network applications.

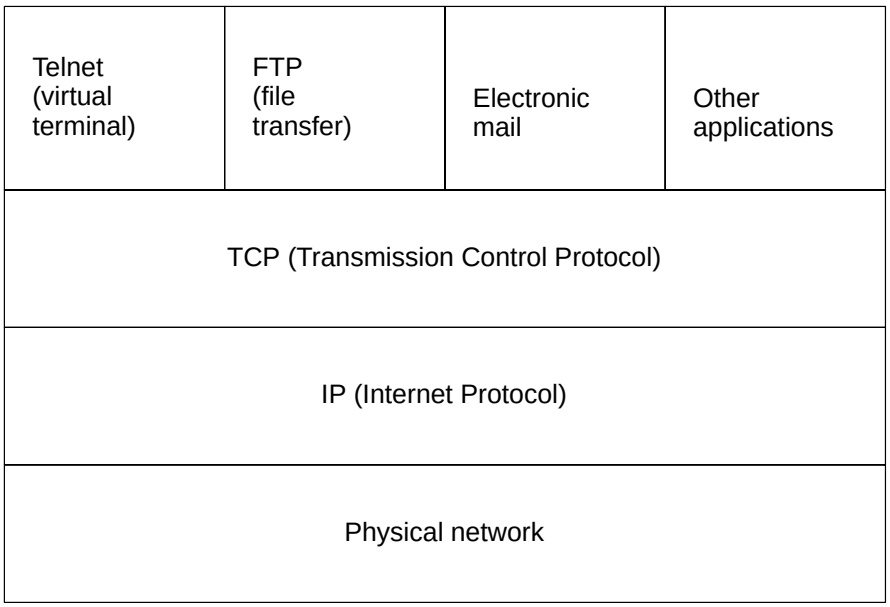


Figure 2. TCP/IP structure

TCP makes logical connections between hosts and ensures that data transmission is accurate. It also adjusts the flow of data between hosts.

IP routes data between hosts, and it forwards data in the network after determining the best available route.

The UNICOS TCP/IP network software offers the following features:

- You can transfer files interactively.
- You can execute commands interactively on remote systems.
- You can send messages interactively.
- You can start a UNICOS shell on a remote system.

Network environment security
1.3

When the UNICOS multilevel security (MLS) feature is enabled, access controls are applied automatically. A check is made to ensure that sensitive information that is transferred to or from a UNICOS system is within the security boundaries for a particular node. For more information about the UNICOS MLS feature, see section 8, page 91, and the *UNICOS Multilevel Security (MLS) Feature User's Guide,* publication SG–2111.

In addition to the UNICOS MLS feature, Cray Research network products support *authorization files* that contain host and user information that is verified by the system before user privileges are granted on a remote system. You create the `.rhosts` and `.netrc` files; the system administrator creates the `/etc/hosts.equiv` and `/etc/ftpusers` files. For more information about security in the UNICOS TCP/IP see section 7, page 77.

Topics covered in this manual
1.4

This subsection provides a synopsis of the topics covered in this manual.

"Getting Started," page 9, gives an overview of the ways in which you can use TCP/IP utilities and commands, display information, and obtain network authorization.

"Executing Commands on a Remote Host," page 17, describes the features and use of the TCP/IP `telnet`(1B), `rlogin`(1B), and `rsh` (see `remsh`(1)) utilities.

"Transferring Files Between Hosts," page 35, describes the features and use of TCP/IP file transfer utilities.

"Communicating Across the Network," page 65, describes the features and use of the TCP/IP utilities that send messages across the network.

"Displaying Host and User Information," page 69, describes the features and use of the TCP/IP utilities that display information such as users' names, terminal names, and so on.

"Network Authorization," page 77, describes security in the UNICOS TCP/IP environment.

"TCP/IP Network Security," page 91, describes TCP/IP and the UNICOS multilevel security (MLS) feature.

"Error Messages," page 119, describes system, `ftp`, and `telnet` error messages.

The information in this section provides an introduction to the capabilities of some TCP/IP utilities and illustrates how you can use them to connect to a remote system, to transfer files, and to send mail messages across a network. You can find more complete descriptions of the utilities described in this section, as well as other methods of accomplishing these tasks, in later sections of this manual.

Finding names of remote hosts
2.1

When you use any of the TCP/IP utilities, you must supply the name of the remote host to which you want to refer. The system administrator for your local Cray Research host will have selected one of two methods for maintaining the list of remote hosts known to your local Cray Research host. These methods use either the `/etc/hosts` file or the domain name service (DNS) software package.

Consult with your system administrator to determine which hosts lists are maintained on your local Cray Research host.

The `/etc/hosts` *file*
2.1.1

The `/etc/hosts` file is a text file that contains the names and network addresses of all hosts on the network that are known to the local Cray Research system. You can use any standard UNICOS utility (for example, `vi` or `cat`) to examine the contents of the `/etc/hosts` file.

In the `/etc/hosts` file, comments begin with the # character; each line that is not a comment is an entry that defines a remote host's interface on a network with the following format (spaces or tabs separate the fields):

address name aliases ...

An address specifies the network to which a host has a physical interface and the host's logical location within that network.

For TCP/IP, the address part of the line is a 32-bit logical address usually expressed as four 8-bit integers separated by periods (also called *dots*, yielding the name *Internet dot notation* for this system of network addressing).

The *name* part of the line (that is, the first host name on the line) is the primary name by which the remote host's interface is known to the local host. For TCP/IP, this is the name that must appear, for example, in a `.rhosts` file for proper network authorization. See subsection 7.2.1, page 79, for the format of the `.rhosts` file.

Alternative names that appear after the first name in the entry are aliases that you can use to refer to the remote host when trying to establish a connection. (For TCP/IP, such aliases should not be used, however, in a `.rhosts` file to try to provide authorization for remote connections.)

The following example fragment from an `/etc/hosts` file shows a TCP/IP address:

```
##########################
#  TCP/IP hosts
##########################
# network number 123 -- HYPERchannel addresses
#
123.45.67.89  sn000 sn000-hy cray
123.45.67.90  gateway gateway-hy
#
# network number 234.56.78 -- Ethernet
addresses
#
234.56.78.90  gateway gateway-et
234.56.78.91  workstation workstation-et
```

In this example, you can specify a TCP/IP connection to the Cray Research host, which is indicated on the first entry in the file, by either the address (`123.45.67.89`), the official host name (`sn000`), or either of the two aliases (`sn000-hy` or `cray`).

Domain name service 2.1.2

The domain name service (DNS) is a software package that communicates with domain name services on other systems to coordinate a distributed database of network addresses and host names, as well as additional information.

A *domain name* is a sequence of names separated by periods (also called *dots*); when viewed from right to left, the domain name defines the logical location of a system in a tree-structured organization of available systems. Thus, given a domain name such as `host.company.bigdomain`, `host` is a specific host within the `company.bigdomain domain`, and `company` is a specific subdomain within the `bigdomain` domain. You can ignore the domain or subdomain part of a domain name (that is, you can ignore everything after the first period) when specifying names that refer to other hosts within the same domain or subdomain. For example, when you are connected to `host1.company.bigdomain`, you can simply refer to `host2.company.bigdomain` as `host2`.

Your system administrator will have located your local Cray Research host in a specific administrative domain or subdomain. To find the names of hosts known to your local Cray Research host, you should consult with your system administrator to determine the name of the domain or subdomain in which your local Cray Research host is located, and then use the `nslookup`(1B) utility to review the DNS for a list of known hosts. The `nslookup`(1B) utility prompts for various subcommands to look up information available from the DNS.

To retrieve a list of remote hosts known to your local Cray Research host, use the `ls` subcommand of `nslookup`, followed by the name of your local domain, as in the following example:

```
$ nslookup

Default Server:  gateway.our.domain
Address:  123.45.67.89
> ls our.domain
[gateway.our.domain]
Host or domain name       Internet address
sn000   123.45.67.89
gateway   234.56.78.90
workstation   234.56.78.91
gateway   123.45.67.90
> quit
$
```

Simply specifying your local domain (`our.domain` in the example) lists only those remote hosts that are actually in your local domain. If your network is connected to other networks outside your local domain, you also can use the DNS to find the names and Internet addresses of many more remote hosts; see the `nslookup`(1B) man page and consult with your system administrator for information about your site's network configuration. You also can use the `host`(1B) command for DNS lookup.

Choosing a suitable command
2.2

In this manual, the TCP/IP commands are grouped to function, as follows:

- Accessing hosts
- Transferring files
- Sending network mail
- Displaying user and host information

The following subsections provide an overview of these routines to help you determine which command to use to complete a task. After you decide which command is most appropriate, proceed to the section of this manual that explains in detail how to use the command.

Accessing hosts
2.2.1

If you have user privileges on a remote host, you can use the TCP/IP utilities to access that host from the terminal on your desk. You can then use all of the commands and files that are available to you on your remote host.

To establish a connection from a remote host to a Cray Research host, see the vendor-supplied documentation or consult with the system administrator of the remote host for connection information that is specific to that host. (If the remote host is running a version of the UNIX operating system with TCP/IP networking utilities derived from the 4.3BSD operating system, you should be able to use the `telnet` command in the manner described in this subsection. When in doubt, see the documentation or consult with your system administrator.)

To access hosts, follow this procedure:

- The `telnet`(1B) utility lets you connect to most operating systems that implement TCP/IP protocols. However, the `telnet` utility does not translate commands between different operating systems. This means, for example, that you can invoke only UNICOS commands on hosts running the UNICOS system.

  The `telnet` command has two modes of operation: command and input. When in *command mode*, you can open and close connections to remote hosts, check the status of a connection, and so on. After a connection is established, you are in *input mode*, which lets you use the resources of the remote host as though you were directly connected to it (this is called a *virtual connection*). The `telnet` utility can pass your terminal type to the remote host if the remote machine supports this functionality. If it does not, you should include terminal type information in your remote host login profile.

  The `telnet` command can operate in *character-at-a-time* mode and *line-by-line* mode. In character-at-a-time mode, each typed character is sent to the remote side for processing. In line-by-line mode, each typed character is processed locally, and only when a complete line has been typed is the data sent to the remote side.

  When running in line-by-line mode, `telnet` provides better response time than does a connection made with `rlogin`(1B), because `telnet` generates less network traffic. For more information about `telnet`, see subsection 3.1, page 17.

- The `rlogin` utility also establishes a virtual connection over TCP/IP to a remote host. Distinguishing features of `rlogin` include the following:

  – Provides automatic login (*autologin*) so that you are not prompted for a login name and password if certain conditions are met.

  – Always forwards your terminal type to the remote host. For more information on the use of `rlogin`, see subsection 3.2, page 28.

- The `rsh` utility (see `remsh`(1B)) executes one shell command on a remote host. For more information on the use of `rsh`, see subsection 3.3, page 31.

Transferring files 2.2.2

The following UNICOS utilities let you transfer files on remote hosts.

- The `rcp`(1) utility copies files between hosts. You can copy files between a Cray Research system and a remote host or between two remote hosts. `rcp` allows autologin (for a description, see subsection 7.1, page 77). For more information on `rcp`, see subsection 4.1, page 36.
- The `ftp`(1B) utility invokes the file transfer program; an `ftp` prompt then appears, indicating that the program is waiting for further instructions. At this point, you can use the autologin feature, open a connection to a remote host, transfer one or more files, append a local file to a remote file, change directories, delete remote files and/or directories, close a connection, or perform a number of other functions that `ftp` recognizes.

  To establish a file transfer connection from a remote host to a Cray Research host, see the vendor-supplied documentation or consult with the system administrator of the remote host for connection information that is specific to that host. (If the remote host is running a version of the UNIX operating system with TCP/IP networking utilities derived from the 4.3BSD operating system, you should be able to use the `ftp` utility in the manner described in this subsection. When in doubt, see the documentation or consult with your system administrator.) For more information on `ftp`, see subsection 4.2, page 40.

Sending network mail 2.2.3

When you are logged on to a Cray Research system, you can use the standard UNICOS utilities `mail`(1), `mailx`(1), or `talk`(1B), described in section 5, page 65.

Displaying user and host information 2.2.4

Several commands let you display information about users and hosts in the network.

- You can use the `finger`(1B) command to display information about users connected to a remote host or logged in to the Cray system through `telnet`(1B) or `rlogin`(1B). By default, `finger` lists the login name, full name, write status, terminal name, idle time, login time, office location, and phone number for each current user. You also can display information about a specific user, including the user's login name, full name, home directory, login shell, login time, idle time, and, if available, project and plan.
- You can display the local host's official host name by using the `hostname`(1) utility.
- The `ping`(8) command provides you with a simple way to find out whether a remote host is operating and reachable over the TCP/IP network.

Executing Commands on a Remote Host [3]

This section describes how you can use TCP/IP utilities to execute commands on other computers on your network. Table 1 lists the utilities and summarizes the introductory information about accessing hosts provided in subsection 2.2.1, page 12. Use this information to choose a utility that will allow you to complete your task, then turn to the appropriate subsection for that utility.

Table 1. Features of TCP/IP utilities for accessing remote hosts

Features	`telnet`(1B)	`rlogin`(1B)	`rsh` (see `remsh`(1B))
Passes terminal type to remote host	Yes	Yes	Yes
Automatic login	No†	Yes	Yes
Establishes a virtual connection	Yes	Yes	No (only one command is executed on the remote host)

† Yes, if Kerberos is used in conjunction with `telnet`.

Using the `telnet` utility

3.1

You can use the `telnet` utility in one of the following modes:

- Input mode
- Command mode

When the utility is in input mode, every command you type is executed as though your terminal were directly connected to the remote host. While in command mode, the local `telnet` utility interprets special `telnet` commands that are typed after a `telnet` prompt (`telnet>`). Read the following subsections to learn how to use both modes of `telnet`.

Using `telnet` *in input mode* 3.1.1

The format of the `telnet`(1B) utility is as follows:

```
telnet [-a] [-d] [-n tracefile] [host [port]]
```

`-a`	Tries automatic login by using Kerberos.
`-d`	Sets the initial value of the debug toggle to `TRUE`. (For more information on the `debug` command, see page 27.)
`-n` *tracefile*	Opens the specified file for recording trace information. (For more information on the `set tracefile` command, see page 26.)
host	Indicates the official name, an alias, or the Internet address of a remote host.
port	Indicates a port number (address of an application). If you omit a number, the default `telnet` port is used.

When in input mode, you can log in to and use the resources of a remote host. The two forms of input mode are character-at-a-time or line-by-line; the mode used depends on which mode the remote host supports. In character-at-a-time mode, most typed text is sent immediately to the remote host for processing. In line-by-line mode, all text is echoed locally, but only completed lines of text are sent to the remote host. You can use the local echo character (initially CONTROL-e) to turn off and on the local echo. Most often, this would be used to enter passwords without the password being echoed, and needed only when the remote side does not properly support line-by-line mode. If you connect to a machine that does not support a line-mode `telnet` server, do an explicit mode line command to force you into kludge line mode. You then can use CONTROL-e to turn local echo on and off. Press CONTROL-e CONTROL-d to enter kludge line mode immediately.

The steps for entering input mode are as follows:

1. Type the `telnet` command and the name of the remote host to which you want to connect. You can use either the official host name, one of its aliases, or the Internet address to identify the remote host. Press RETURN. In the following example, the remote host is `remote_host`.

```
$ telnet remote_host
Trying...
Connected to remote_host.
Escape character is '^]'.
remote operating system (remote_host)
login:
```

As shown, after you execute the `telnet` command, the utility displays text as it tries to make the connection. Line 3 shows that the connection is made. Line 4 displays the default escape character, `^]` (or CONTROL-]), which is used to enter command mode. Line 5 shows the remote host's operating system and the name of the remote host. Line 6 is the login prompt.

2. Type your login name for the remote host and press RETURN, as follows:

```
login: my_name
password:
```

3. Type your remote-system password after the prompt and press RETURN. Your password does not appear on the screen. This is a security measure.

   The remote host checks the system authorization files before granting you access. You must have a login account on the remote host to access it through `telnet`. Consult with your system administrator for help in setting up accounts and user profiles on remote hosts.

4. When you complete the login process, a message is issued and the remote host's prompt appears on the screen, as follows:

```
Last successful login was:  Tue Aug 23 14:00:31
from Cray Research host
%
```

Now you have a virtual connection and can use the resources of the remote host as though your terminal were directly attached to it.

Usually, the remote host determines whether the connection will operate in character-at-a-time or line-by-line mode. You can use the `telnet status` command to determine which mode is currently being used. You can manually override the mode by using the `telnet mode` command. For more information on the use of these commands, see subsection 3.1.3, page 23.

If you are using a slow network to get to the remote host, it is useful to switch from character-at-a-time to line-by-line mode if the remote host supports it. This switch causes no loss of functionality and eliminates the generation of network traffic for each character typed, resulting in better command response time. Even if the remote host does not support line-by-line mode, you can still use it. In this case, you must manually disable character echoing on the remote system; consequently, you cannot use any visually oriented commands (such as a full-screen editor) while in line-by-line mode.

After you complete your work on the remote host, terminate input mode by using the remote host's normal logout procedure. When you log out from the remote host, the `telnet` utility returns you to your working directory on the Cray Research system, with the following message:

```
Connection closed by foreign host.
$
```

Using `telnet` *in command mode* 3.1.2

The `telnet` command mode is not used to execute commands on a remote host, but rather, to execute `telnet` commands for the purpose of opening or closing a connection, displaying `telnet` information, or changing the conditions of the `telnet` utility. You can enter the `telnet` command mode in the following ways:

- If you are in your working directory on the Cray Research system, type `telnet` without a remote host name.
- If you are in `telnet` input mode, press the escape character.

Use the following procedure to enter `telnet` command mode from your working directory on a Cray Research system:

Type `telnet` and press RETURN. The utility responds with the `telnet` prompt to indicate that you are in command mode and that the `telnet` utility is ready to execute a `telnet` command.

```
$ telnet
telnet>
```

Enter a `telnet` command (see subsection 3.1.3, page 23, for a list of available commands), using the following format:

`telnet>` *command*

For example, to display the status of the `telnet` utility, type the `status` command, as follows:

```
telnet> status
No connection.
Operating in line-by-line mode.
Escape character is '^]'
telnet>
```

Use the following procedure to enter command mode from input mode:

Enter the escape character CONTROL-], as follows:

```
%^]
telnet>
```

Usually, a connection is closed by logging out from the remote system. If for some reason you cannot do that, use the `close` command to close the `telnet` connection, as follows:

```
telnet> close
Connection is closed.
telnet>
```

To exit the `telnet` utility completely, type the `quit` command and press RETURN. The Cray Research system prompt then appears, as follows:

```
telnet> quit
$
```

Use the following procedure to enter input mode from command mode:

Type the `open` command and the name of the remote host. Press RETURN.

```
telnet> open my_host
Trying...
Connected to my_host
Escape character is '^]'.
remote operating system (my_host)
login:
```

Type your login name and press RETURN. Type your password and press RETURN. The `telnet` utility then establishes a connection, and the remote host displays its system prompt. Now you can work on the remote host.

Note: Invoking the `open` command from command mode is the same as entering `telnet` input mode. Both establish a virtual connection and let you use the remote host.

`telnet` *commands*
3.1.3

This subsection lists the `telnet` commands and describes their functions. You can abbreviate command names to the minimum number of characters required to uniquely identify the command. In the following list, brackets enclose the part of a command name that you can omit.

`c[lose]`

Closes a `telnet` session and returns to command mode.

`d[isplay]` *arguments*

Displays all, or some, of the set and toggle values (see values for `set` and `toggle` commands).

`m[ode]` *type*

The *type* argument is either `line` (for line-by-line mode) or `character` (for character-at-a-time mode). Requests permission from the remote host to go into the requested mode. If the remote host can enter that mode, the requested mode is entered.

`o[pen]` *host* [*port*]

Opens a connection to the specified host. If you do not specify a port number, `telnet` tries to contact a `telnet` server at the default port. The host specification can be either a host name (see `host`(1B)) or an Internet address specified in the "dot notation" (see `inet`(3)).

`q[uit]`

Closes any open `telnet` session and exits `telnet`. An end-of-file (EOF) character (in command mode) also closes a session and exits.

`sen[d]` *arguments*

Sends one or more special character sequences to the remote host. The following are the arguments that can be specified (more than one argument can be specified at a time):

Argument	Description
`ao`	Sends the `telnet` AO (Abort Output) sequence, which causes the remote system to flush all output from the remote system to the user's terminal.
`a[yt]`	Sends the `telnet` AYT (Are You There) sequence, to which the remote system may choose to respond.
`brk`	Sends the `telnet` BRK (Break) sequence, which might have significance to the remote system.

Argument	Description
`ec`	Sends the `telnet` EC (Erase Character) sequence, which causes the remote system to erase the last character entered.
`el`	Sends the `telnet` EL (Erase Line) sequence, which causes the remote system to erase the line currently being entered.
`es[cape]`	Sends the current `telnet` escape character (initially `~]`).
`g[a]`	Sends the `telnet` GA (Go Ahead) sequence, which probably has no significance to the remote system.
`ip`	Sends the `telnet` IP (Interrupt Process) sequence, which cause the remote system to abort the currently running process.
`n[op]`	Sends the `telnet` NOP (No OPeration) sequence.
`s[ynch]`	Sends the `telnet` SYNCH sequence, which causes the remote system to discard all previously typed (but not yet read) input. This sequence is sent as TCP urgent data (and might not work if the remote system is a 4.3BSD system). If the SYNCH sequence does not work, a lowercase `r` might be echoed on the terminal.
`?`	Prints out help information for the `send` command.

`set` *variable values*

Sets any one of a number of `telnet` variables to a specific value. The special value `off` turns off the function associated with the variable. The values of variables can be interrogated with the `display` command. The variables that you can specify are as follows:

Variable	Description
`ec[ho]`	This is the value (initially CONTROL-e) which, when in line-by-line mode, toggles between doing local echoing of entered characters for standard processing, and suppressing echoing of entered characters (for example, for entering a password).
`eo[f]`	If `telnet` is operating in line-by-line mode, entering this character as the first character on a line causes this character to be sent to the remote system. The initial value of the EOF character is taken to be the terminal's EOF character.
`er[ase]`	If `telnet` is in `localchars` mode (see `toggle localchars` that follows), and if `telnet` is operating in character-at-a-time mode, a `telnet` EC sequence (see `send ec` preceding) is sent to the remote system when this character is typed. The initial value for the erase character is taken to be the terminal's erase character.
`es[cape]`	This is the `telnet` escape character (initially CONTROL-]), which causes entry into `telnet` command mode (when connected to a remote system).
`i[nterrupt]`	If `telnet` is in `localchars` mode (see `toggle localchars` that follows), and the interrupt character is typed, a `telnet` IP sequence (see `send ip`, under `send` *arguments*, preceding) is sent to the remote host. The initial value for the interrupt character is taken to be the terminal's interrupt character.

Variable	Description
k[ill]	If telnet is in localchars mode (see toggle localchars that follows), and if telnet is operating in character-at-a-time mode when this character is typed, a telnet EL sequence (see send el, under send *arguments*) is sent to the remote system. The initial value for the kill character is taken to be the terminal's kill character.
q[uit]	If telnet is in localchars mode (see toggle localchars that follows), and the quit character is typed, a telnet BRK sequence (see send brk preceding) is sent to the remote host. The initial value for the quit character is taken to be the terminal's quit character.
t[racefile] *tracefile*	If either netdata or options was toggled to be TRUE, the debugging information generated by them is written into *tracefile*. If *tracefile* is given as –, the debugging information is written to standard output (the default case).

st[atus] Shows the current status of telnet. The information displayed includes the name of the peer to which you are connected, as well as the current mode.

t[oggle] *arguments* Toggles (between TRUE and FALSE) various flags that control the manner in which telnet responds to events. You can specify more than one argument. To interrogate the state of these flags, use the display command. Valid arguments are as follows:

Arguments	Description
autof[lush]	If autoflush and localchars are both TRUE, when the interrupt or quit characters are recognized (and transformed into telnet sequences; see the entry for set *variable values* for details), telnet refuses to display any data on the user's terminal until the remote system acknowledges (by way of a telnet Timing Mark option) that it has processed those telnet sequences. The initial value for this toggle is TRUE (see stty(1)).

Arguments	Description
`autos[ynch]`	If `autosynch` and `localchars` are both `TRUE`, when either the interrupt or quit characters are typed (see the entry for `set` *variable values* for descriptions of the interrupt and quit characters), the resulting `telnet` sequence sent is followed by the `telnet` SYNCH sequence. This procedure should cause the remote system to begin discarding all previously typed input until both `telnet` sequences have been read and acted upon. The initial value of this toggle is `FALSE`.
`crm[od]`	Toggles carriage return mode. When this mode is enabled, most carriage return characters received from the remote host are to be mapped into a carriage return followed by a line feed. This mode does not affect those characters that the user types; it affects only those received from the remote host. This mode is not very useful unless the remote host sends carriage returns, but never line feeds. The initial value for this toggle is `FALSE`.
`d[ebug]`	Toggles socket-level debugging (useful only to the super user). The initial value for this toggle is `FALSE`.
`l[ocalchars]`	If this is `TRUE`, the interrupt, quit, erase, and kill characters (see `set` *variable values*, preceding) are recognized locally, and transformed into appropriate `telnet` control sequences (respectively, BRK, EC, EL, and IP; see the entry for `send` *arguments*, preceding). The initial value for this toggle is `TRUE` in line-by-line mode, and `FALSE` in character-at-a-time mode.
`n[etdata]`	Toggles the display of all network data (in hexadecimal format). The initial value for this toggle is `FALSE`.

Arguments	Description
o[ptions]	Toggles the display of internal `telnet` protocol option processing. The initial value for this toggle is FALSE.
?	Displays the legal toggle commands.

`z` — Suspends the `telnet` command and executes a subshell.

? [*command*] — Gets help. Without arguments, `telnet` prints a help summary. If a command is specified, `telnet` prints the help information for the specified command.

Using the `rlogin` utility

3.2

The `rlogin` utility lets you automatically log in to a remote host on which you have an account. It establishes a virtual connection; that is, your terminal appears to be physically connected to the remote host.

If you are accessing a remote host that does not support `rlogin`, you can use the `telnet` utility as an alternative login service.

Note: When the UNICOS multilevel security (MLS) feature is enabled, the autologin capability of inbound `rlogin` is allowed only if the following requirements are met:

- The `NETW_RCMD_COMPAT` configuration parameter is disabled. Check with your system administrator
- The client host is named in the `/etc/host.equiv` file.
- Your remote and local user IDs are identical.
- Your user ID is specified in the `.rhosts` file of the server.
- The client host is also specified in the `.rhosts` file of the server.
- The workstation access list (WAL) specifies `login` permission for one or more of the following: the remote host you want to access, your account name, or your group ID entry. See subsection 8.2.4, page 104, for more information.

The `rlogin` utility has the following format:

```
rlogin rhost [-ec] [-8] [-l username]
```

rhost Indicates the official name, an alias, or the Internet address of a remote host.

`-e`*c* Lets you change the `rlogin` escape character from the tilde (~) to the new escape character (*c*). Do not type a space between the option character and the argument specifying the new escape character.

`-8` Allows transmission of 8-bit data.

`-l` *username*
Lets you log in automatically, even if your login name on the host to which you want access does not match your login name on the Cray Research system. To use this option, enter your remote-host login name after `-l`. Check with your system administrator about whether this option is disabled if you are on a UNICOS MLS system.

To log in to and out of another host, complete the following steps:

1. Type the `rlogin` command and the name of the remote host. You can specify the host's official name, an alias, or the Internet address. Press RETURN.

```
$ rlogin my_host
Password:
```

2. The password prompt does not appear if you are authorized for automatic login. If the password prompt appears, type your remote-host password after the prompt. As a security measure, your password is not displayed on the screen. After `rlogin` establishes a connection, it displays system information and the remote host's prompt, as follows:

```
Password:
Last successful login was:
Wed Sep 7 13:16:14 from Cray Research host
%
```

Now you can use the resources of the remote host.

3. To terminate the remote connection, type the remote host's logout command. (In the following example, the command is `exit`.) The `rlogin` utility then returns you to your working directory on the Cray Research system.

```
% exit
Logout: Thu Sep 8 13:44:18 CDT 1988
Connection closed.
$
```

Other options that you can use with the `rlogin` command are explained in the following examples.

Example 1:

In this example, the local host is called `cray`, the remote host is `engineering`, and the user's login name on both hosts is `bonnie`. The user types the `rlogin` command followed by the name of the remote host and presses RETURN.

```
$ rlogin engineering
Last login was: Fri Sep 9 15:56:53 from cray
%
```

The remote host `engineering` checks its `/etc/hosts.equiv` file to determine whether host `cray` is listed. When it finds `cray`, it checks to see whether the login name `bonnie` is in its password file. It is, so the user is logged in automatically. User `bonnie` is not asked to provide a password because she already authenticated her account by entering her password when she logged in to her local host. When autologin is complete, the remote host's prompt appears on the screen.

For more information on the autologin feature, see subsection 7.1, page 77, Also see the preceeding note in this subsection about using autologin with UNICOS MLS enabled.

Example 2:

In this example, the user's login name on the local host is `adam`, and his login name on the remote host (`math`) is `apj`. No account exists on the remote host with a login name `adam`. Because the user's login names are different on the two hosts, he must specify the `-l` option with the login name `apj`. The `-l` option tells the `rlogin` utility to look for the user's local host and login name in `the .rhosts` file for `apj` in the home directory on `math`. When the entry is found, the user is automatically logged in to the remote host.

```
$ rlogin math -l apj
Last successful login was:
Fri Sep 9 14:51:58 from cray
%
```

Example 3:

In the following example, the user logs in to remote host `chemistry` and changes the escape character from ~ to @. To do this, the user types the `rlogin` command, the remote host name, and the `-e` option followed by the new escape character, @, and presses RETURN. As demonstrated, the utility recognizes @ as the new escape character and closes the connection.

```
$ rlogin chemistry -e@
Last successful login was:
Fri Sep 9 14:51:58 from cray
% @.
Closed connection.
$
```

Using the rsh utility

3.3

The `rsh` (remote shell) utility initiates a login to a remote host and executes a command. Like `rlogin`, `rsh` gives you automatic authorization to access your accounts on remote hosts. When you use the `rsh` utility to execute a command on a remote host, interrupt, quit, and terminate signals are passed to the remote host. The remote login terminates when the command finishes processing; you are then returned to your working

directory on the Cray Research system. Because the rsh utility lets you invoke only a single shell script before returning you to the Cray Research system, you cannot use interactive commands such as vi. For interactive applications, use telnet or rlogin.

If automatic authorization is not available on your system, you can use either the rlogin or telnet utility as an alternative to rsh.

Note: When the UNICOS multilevel security (MLS) feature is enabled, the use of the rsh utility is allowed only if the following requirements are met:

- The NETW_RCMD_COMPAT configuration parameter is disabled. Check with your system administrator.
- The client host is named in the /etc/host.equiv file.
- Your remote and local user IDs are identical.
- Your user ID is specified in the .rhosts file of the server.
- The client host is also specified in the .rhosts file of the server.
- The workstation access list (WAL) specifies rsh permission for one or more of the following: the remote host you want to access, your account name, or your group ID entry.

The rsh utility, located in the /usr/ucb directory, has an identically named, but entirely different, rsh (restricted shell) command in the /bin directory. Therefore, remsh, an alternative name for the remote shell utility, is also located in /usr/ucb. Thus, to be certain you are using the remote shell utility, you can either use remsh rather than rsh, or place the /usr/ucb directory before the /bin directory in your search path (determined by the $PATH environment variable).

Note: Shell metacharacters (&, ?, *, |, \, ;, <, and >) that are entered without quotation marks around them are interpreted on the Cray Research system; quoted metacharacters are interpreted on the remote host.

Use the following format to invoke the `rsh` command:

```
rsh host [-l username] [-n] [command]
```

host Indicates the official name, an alias, or the Internet address of a remote host.

`-l` *username*
Lets you log in automatically, even if your login name on the host to which you want access does not match your login name on the Cray Research system. To use this option, enter your remote-host login name after `-l`. See the note at the beginning of this subsection for requirements that must be met to use this option.

`-n` If no input is desired, you must use `-n` to redirect the input of `rsh` to the `/dev/null` file.

command A UNIX command or the name of a script on the remote host. If you omit this argument, and the only argument to `rsh` is *host*, an `rlogin` command is executed to establish a connection to *host*.

These options and arguments are explained in the following examples.

Example 1:

In this example, the user types the `rsh` command, the remote host name `bio`, the `who` command, and presses RETURN. The `rsh` utility performs an autologin to remote host `bio` and executes the `who` command, displaying information on the user's terminal. After the command is executed, the utility returns the user to his or her working directory on the local host and displays the local system prompt.

```
$ rsh bio who
bonnie   ttyd1    Sep   9 15:34
bdh      ttyd7    Sep   9 15:07
deb      ttyp1    Sep   9 16:00    (rei-sc)
adam     ttyp3    Sep   9 11:15    (lanman)
jeni     ttypc    Sep   9 16:22    (social-gate)
$
```

Example 2:

In this example, a user on local host `cray` wants to execute the `ls` command on remote host `chemistry` in order to see the files in his remote host directory. The user's login name on the Cray Research system is not the same as the login name of his account on the remote host.

The user types `rsh`, the remote host's name (`chemistry`), the `-l` option followed by the login name `scott`, the `ls` command, and presses RETURN. The `-l` option flags the `rsh` utility to check the `.rhosts` file in the home directory of user `scott`. The utility looks for the host name of the local host and the login name of the user initiating the remote login request. When they are found, the user is logged in automatically. The system executes the `ls` command, displays the file names on the screen, then returns the user to his working directory on the Cray Research system.

See the note at the beginning of this subsection for requirements to use the `-l` option.

```
$ rsh chemistry -l scott ls
data
memo
memo.sched
statistics
$
```

Transferring Files Between Hosts [4]

Table 2 shows the file transfer utilities that UNICOS TCP/IP supports.

Table 2. Functions of TCP/IP utilities for file transfer

Utility	Function
`rcp`(1)	Copies files between operating systems based on the UNIX operating system (much like `cp`(1)).
`ftp`(1B)	General-purpose file transfer utility with an interactive interface. This method provides the fastest transfer rates.
`tftp`(1B)	Provides a very limited file access mechanism.

In addition, the network file system (NFS) remote file system feature allows you to directly access data in remote files. This method provides the most convenient access to remote file data.

Before using the file transfer utilities, set up authorization files according to the instructions provided in section 7, page 77. The `rcp` utility accesses your local host's `/etc/hosts.equiv` file or your `$HOME/.rhosts` file and/or both. The `ftp` utility and `rexec`(3) library routine access your `$HOME/.netrc` file. (If your `.rhost` file is world-writeable, this access will not work.) The `ftp` utility and `rexec`(3) library routine access your `$HOME/.netrc` file. `ftp` also uses the `/etc/ftpusers` and `/etc/shells` files at the remote system.

The following subsections explain the capabilities of each utility and describe how to use them.

Using the rcp utility
4.1

The rcp utility has the following features:

- Automatically logs you in to the remote host.
- Copies files between a Cray Research system and a remote host, or between two remote hosts (known as *third-party copying*).

  Note: When the UNICOS multilevel security (MLS) feature is enabled, you can use the rcp utility only if the following requirements are met:

  - The NETW_RCMD_COMPAT configuration parameter is disabled. Check with your system administrator.
  - The client host is named in the /etc/host.equiv file.
  - Your remote and local user IDs are identical.
  - Your user ID is specified in the .rhosts file of the server.
  - The client host is also specified in the .rhosts file of the server.
  - The workstation access list (WAL) specifies rcp permission for one or more of the following: the remote host you want to access, your account name, or your group ID entry.

You can use the rcp command to copy one file into another file or to copy multiple files into a directory. The command syntax is as follows:

```
rcp [-p] [-r] file1 file2
```

-p	Preserves in its copies the modification times and access modes of the source files, ignoring the user file-creation mode mask (see umask(1)).
-r	Tells rcp to copy each subtree of the directory if *file1* is a directory. In this case, *file2* must be a directory.
file1	The name of the file or directory you want to copy. See subsection 4.1.1 for additional information.
file2	The name of the file or directory to which you want to copy. See subsection 4.1.1 for additional information.

If the remote host does not support `rcp`, you can use the `ftp` utility as an alternative.

Specifying file names 4.1.1

Local and remote file names are specified as described in the following subsections.

Local file names 4.1.1.1

Specify the full path name of the file or the path name relative to the current directory. A local file name cannot contain a colon (:) unless the path name has a slash (/) at some point prior to the first colon (see example 8, page 40).

Remote file names 4.1.1.2

You can specify remote file names in the following ways:

- You can use shell metacharacters to specify a remote file name.
- You can specify files on a remote host, as follows:

  hostname:*filename*

hostname	Either the official name, alias, or Internet address of the remote host.
filename	Path name of the file. If *filename* is not a full path name, `rcp` interprets it relative to your home directory on *hostname*.

- When the account login name differs from your login name on the Cray Research system, you can specify file names on a remote host as follows:

  user@*hostname*:*filename*

user	Login name associated with the account
hostname	Official host name, alias, or Internet address of the remote host
filename	Path name of the file (full or relative to the home directory)

Remember to use an at-sign (@) symbol between the *user* and *hostname* specifications, and a colon (:) between the *hostname* and the *filename*.

`rcp` *utility examples* 4.1.2

The `rcp` utility is simple to use because one command completes numerous types of copies. Therefore, instead of providing step-by-step instructions, the `rcp` utility is described through examples.

Example 1:

To copy file `proposal`, which is on the Cray Research system, into file `report` on remote host `chemistry`, type the following command line and press RETURN.

```
$ rcp proposal chemistry:report
$
```

After the command is executed, the Cray Research prompt appears, indicating that the copy to `chemistry` was successful.

Example 2:

To copy file `whale`, which is on the remote host `biology`, into file `mammal` on the Cray Research system, type the following command line and press RETURN:

```
$ rcp biology:whale mammal
$
```

Again, the Cray Research prompt appears after a successful copy is completed.

Example 3:

If you want to copy a file to or from an account with a login name that differs from your login name on the Cray Research system, review the following example.

In this example, the remote file `letter` must be accessed under the login name `tami`. To copy the file from remote host `engineering` into a file called `memo` on the Cray Research system, type the following command line and press RETURN:

```
$ rcp tami@engineering:letter memo
$
```

Example 4:

To copy files between two remote hosts (same login names on the hosts), specify the host names and file names, separated with a colon. For instance, to copy the file `proposal` on host `biology` into the file `report` on host `chemistry`, type the following command line and press RETURN:

```
% rcp biology:proposal chemistry:report
%
```

Example 5:

To copy all of the files that have names that begin with `h2o` from remote host `chemistry` into your working directory on the Cray Research system, type the following command line and press RETURN:

```
$ rcp chemistry:"h2o*" .
$
```

The dot (.) designates the destination to be your working directory on the local host. The `h2o*` part of the source name designates all files that have names that begin with `h2o*`. Because the desired files are so named on the remote host, double quotation marks are necessary around this part of the name to prevent the local host from trying to interpret the * by substituting the names of any local files with similar names.

Example 6:

To copy the entire contents of local directory `work` to a directory with the same name in your home directory on remote host `eng`, type one of the following command lines and press RETURN:

```
$ rcp -r work eng:.
$
$ rcp -r work eng:
$
```

Example 7:

To copy multiple files on remote host `bio` into the `task` directory on the Cray Research system, type the following command line and press RETURN:

```
$ rcp bio:measure bio:data bio:facts task
$
```

Example 8:

To copy the local file `foo:bar`, you cannot specify the local file name in the usual manner because `rcp` interprets the colon as a separator and, therefore, interprets the operand as file `bar` on machine `foo`. The following example shows what happens if you specify the local file name in the usual manner:

```
$ rcp foo:bar chemistry:
foo: unknown host
$
```

In this case, the local file name must be specified as a full path name, or a path name relative to the current directory. Either of the following examples causes `rcp` to recognize `foo:bar` as the local file name:

```
$ rcp ./foo:bar chemistry:
$
$ rcp /usr/tami/foo:bar chemistry:
$
```

Using the `ftp` utility

4.2

The `ftp` utility offers the following features:

- Provides a full range of interactive file manipulation capabilities, such as copying, deleting, and appending.

- Automatically logs you in to the remote host if you have a `.netrc` file in your home directory on the Cray Research system (see subsection 7.2.2, page 83), or if Kerberos is used in conjunction with `ftp`.
- Communicates with all hosts on your network, regardless of the operating system.
- Offers a helpful, prompt-driven command mode that assists you in transferring files to and from remote hosts.

The syntax of the `ftp` utility is as follows:

```
ftp    [-c copybufsize] [-d] [-g] [-i] [-n] [-s  sockbufsize]
       [-t] [-v] [-Sc tos] [-Sd tos] [host [port]]
```

`-c` *copybufsize*
: Sets the copy buffer size. A numeric argument sets the buffer to that size. The letter `K` or `k` can follow a numeric buffer size to specify a multiple of 1024.

`-d`
: Enables debugging.

`-g`
: Disables file name globbing; that is, metacharacters (`*`, `?`, `[...]`) in file names are not expanded to the string they represent.

`-i`
: Turns off interactive prompting during multiple file transfers.

`-n`
: Tells `ftp` to ignore the `.netrc` file. This disables autologin.

`-s` *sockbufsize*
: Sets the socket buffer size. A numeric argument sets the buffer to that size. The letter `K` or `k` can follow a numeric buffer size to specify a multiple of 1024.

`-t`
: Enables tracing.

`-v`
: Turns off verbose mode, suppressing all responses from the remote server, including data transfer statistics. Verbose mode is set on by default.

`-Sc` *tos* Sets the Internet Protocol (IP) type-of-service (TOS) option for the `ftp` control connection to the value *tos*, which may be a numeric type of service value or a symbolic TOS name that is found in the `/etc/iptos` file.

`–Sd` *tos* Sets the Internet Protocol (IP) type-of-service (TOS) option for the `ftp` data connection to the value *tos*, which may be a numeric TOS value or a symbolic TOS name that is found in the `/etc/iptos` file.

host[*port*] Specifies the host or port with which `ftp` will communicate.

The `ftp` utility requires that you supply your login name and password to the remote system, unless Kerberos is used in conjunction with `ftp`. You can do this in three ways:

- Supply your login name and password each time `ftp` prompts you.
- Execute the `ftp user` command.
- Create a `.netrc` file that forwards your login name and password for autologin. For information on setting up this file, see subsection 7.2.2, page 83.

Common `ftp` *functions* 4.2.1

This subsection describes some of the common uses of the `ftp` utility. To use `ftp` as described in this subsection, you must have an account on the remote host that you are accessing, and you must create a `.netrc` file for autologin.

To use the `ftp` utility, type `ftp` and press RETURN. The `ftp` prompt (`ftp`) appears.

```
$ ftp
ftp>
```

At this point, you can execute any `ftp` commands that do not require a connection to a remote host (for example, `open` or `help`). The `ftp` commands are described in subsection 4.2.2, page 54.

Logging in to a remote host
4.2.1.1

To log in to a remote host from a Cray Research host, type `ftp`, followed by the name of the remote host to which you want to connect and press RETURN. You can use the official host name, an alias, or the Internet address to identify the remote host. (If you have already accessed the `ftp` utility, type `open` and then the remote host name at the `ftp>` prompt.) In the following example, the remote host name is `myhost`, and the login name is `mylogin`.

```
$ ftp myhost
Connected to myhost.
220 myhost FTP server (Version 4.15 Sat Nov 7 15:24:41 PST 1987)
ready.  Name (myhost:mylogin):
```

If `mylogin` is the correct account name on the remote host to which you want to connect, you can simply press RETURN. Otherwise, type in the name of the appropriate account and press RETURN. In the following example, the appropriate account name is `othername`. After you enter the account name, the remote host continues, as follows:

```
Name (myhost:mylogin): othername
331 Password required for othername.
Password:
```

Type the password for the account and press RETURN. You will receive confirmation that you are logged in, and the `ftp` prompt will appear, as follows:

```
Password:
230 User othername logged in.
ftp>
```

You are now logged in.

Copying a file from a remote host
4.2.1.2

To copy a file from a remote host to your working directory on the Cray Research system, type the `get` command and the name of the file you want to copy. If you want the file to have a different name in your Cray Research directory, type the new

name after the name of the file you want to copy. In the following example, `rem_file` is the name of the file you want to copy, and `loc_file` is the name by which the file will be known in your directory.

```
ftp> get rem_file loc_file
200 PORT command okay.
150 Opening data connection for rem_file (84.0.194.5,1038) (18 bytes).
226 Transfer complete.
local: loc_file  remote: rem_file
18 bytes received in 0.031 seconds (0.57 Kbyte/s)
ftp>
```

Copying multiple files
4.2.1.3

To copy multiple files from a remote host to your working directory on the Cray Research system, type the `mget` command and the names of the files you want to copy. The interactive prompt will ask you to verify the transfer of each file. You can toggle prompting on or off. If you do not want to be prompted for each file, follow the steps described for the use of the `ftp prompt` command, page NO TAG, or use the `-i` option when you invoke `ftp`.

The following example shows how the transfer occurs with the interactive prompt enabled. The interactive prompt lets you change your mind and not transfer one or more of the files designated on the command line.

```
ftp>  mget file1 file2 file3
mget file1? y
200 PORT command okay
150 Opening data connection for file1 (84.0.194.5,1043) (57 bytes).
226 Transfer complete.
local: file1 remote: file1
57 bytes received in 0.017 seconds (3.3 Kbyte/s)
mget file2? n
mget file3? y
200 PORT command okay
150 Opening data connection for file3 (84.0.194.5,1045) (60 bytes).
226 Transfer complete.
local: file3 remote: file3
60 bytes received in 0.063 seconds (0.94 Kbyte/s)
ftp>
```

Copying files to a remote host
4.2.1.4

To copy a file from the Cray Research system to your home directory on a remote host, type the `put` command and the name of the file you want to copy. In the following example, only the local file name (`file1`) is specified; therefore, the copy retains the original file name.

```
ftp> put file1
200 PORT command okay.
150 Opening data connection for file1(84.0.194.5,1059).
226 Transfer complete.
local: file1 remote: file1
60 bytes received in 0.00098 seconds (60 Kbyte/s)
ftp>
```

You also can use the `mput` command to copy multiple files from the Cray Research system. It works like `mget`, except that files are moving from the Cray Research system, rather than to the Cray Research system.

Appending files
4.2.1.5

To append a local (Cray Research system) file to a remote file, type the `append` command followed by the local file name and the remote file name, and press RETURN, as in the following example. In both cases, the file name can be full or relative.

```
ftp> append crayfile remfile
200 PORT command okay.
150 Opening data connection for remfile (84.0.194.5,1067).
226 Transfer complete
local: crayfile remote: remfile
60 bytes sent in 0.001 seconds (56 Kbyte/s)
ftp>
```

Deleting files
4.2.1.6

To delete a file from the remote host, type the `delete` command and the path name (full or relative) of the file you want to delete and press RETURN, as follows:

```
ftp> delete my_file
200 DELE command okay.
ftp>
```

You can use the `mdelete` command to delete multiple files, as in the following example. The `ftp` utility prompts you to confirm the deletion of each file, unless you execute the `ftp prompt` command prior to the `mdelete` command or specify the `-i` option when `ftp` is invoked.

```
ftp> mdelete file1 file2
delete file1? y
200 DELE command okay.
delete file2? y
200 DELE command okay.
ftp>
```

Defining macros
4.2.1.7

The `macdef` command lets you define macros within `ftp`. The macros also can be put into the `.netrc` file. The following example illustrates the use of the `macdef` command. Macros `doall` and `mrmdir` are defined. The `doall` macro issues the `pwd` and `dir` commands when executed. The `mrmdir` macro removes specified directories when executed.

```
$ ftp
ftp> open biology
Connected to biology.
220 biology FTP server (Version 4.3 Fri Dec 9 17:36:01 CST 1988)
ready.  331 Password required for bonnie.
Password:
230 User bonnie logged in.
ftp> macdef doall
Enter macro line by line, terminating it with a null line
pwd
dir

ftp> $doall
pwd
257 "/usr/bonnie" is current directory.
dir
200 PORT command successful.
150 Opening data connection for /bin/ls (84.0.194.5,1076) (0 bytes).
total 14
drwxr-x---   2 bonnie   grpA       288 Nov 28 14:25 X
drwx------   2 bonnie   grpA        64 Dec 12 11:31 a
drwx------   2 bonnie   grpA        64 Dec 12 11:31 b
drwxrwxrwx   2 bonnie   grpA       160 Dec  5 11:19 bin
drwx------   2 bonnie   grpA        64 Dec 12 11:30 c
drwx------   2 bonnie   grpA        64 Dec 12 11:31 d
drwxr-x---   2 bonnie   grpA       480 Nov  9 12:41 ip
drwxr-x---   2 bonnie   grpA       448 Aug 16 17:17 perf
drwxr-x---   2 bonnie   grpA       992 Oct  6 09:30 sim
drwx------   2 bonnie   grpA       544 Nov 11 18:50 socket
drwxr-x---   3 bonnie   grpA       384 Dec  6 13:47 src
drwxr-xr-x   2 bonnie   grpA       128 Aug 25 09:42 stress
drwxr-x---   6 bonnie   grpA       480 Dec  9 12:35 temp
drwxr-x---   9 bonnie   grpA       480 Oct 18 16:18 test
226 Transfer complete.
819 bytes received in 0.072 seconds (11 Kbyte/s)

ftp>
ftp> macdef mrmdir
Enter macro line by line, terminating it with a null line
rmdir $i

ftp> $mrmdir a b c d
rmdir a
250 RMD command successful.                                    (continued)
```

```
rmdir b
250 RMD command successful.
rmdir c
250 RMD command successful.
rmdir d
250 RMD command successful.
ftp> dir
200 PORT command successful.
150 Opening data connection for /bin/ls (84.0.194.5,1077) (0 bytes).
total 10
drwxr-x---   2 bonnie   grpA       288 Nov 28 14:25 X
drwxrwxrwx   2 bonnie   grpA       160 Dec  5 11:19 bin
drwxr-x---   2 bonnie   grpA       480 Nov  9 12:41 ip
drwxr-x---   2 bonnie   grpA       448 Aug 16 17:17 perf
drwxr-x---   2 bonnie   grpA       992 Oct  6 09:30 sim
drwx------   2 bonnie   grpA       544 Nov 11 18:50 socket
drwxr-x---   3 bonnie   grpA       384 Dec  6 13:47 src
drwxr-xr-x   2 bonnie   grpA       128 Aug 25 09:42 stress
drwxr-x---   6 bonnie   grpA       480 Dec  9 12:35 temp
drwxr-x---   9 bonnie   grpA       480 Oct 18 16:18 test
226 Transfer complete.
595 bytes received in 0.082 seconds (7.1 Kbyte/s)
ftp> quit
221 Goodbye
$
```

Connecting to two hosts
4.2.1.8

You can use the `proxy` command to connect to two different hosts in the same `ftp` session. In the following example, the user connects to host `biology` and then uses the `proxy` command to connect to host `chemistry` in the same `ftp` session. The `proxy` command is used to issue other `ftp` commands to host `chemistry` during the `ftp` session.

```
$ ftp
ftp> open biology
Connected to biology.
220 biology FTP server (Version 4.3 Fri Dec 9 17:36:01 CST 1988) ready.
331 Password required for bonnie.
Password:
230 User bonnie logged in.
ftp> proxy open chemistry
Connected to chemistry.
220 chemistry FTP server (Version 4.1 Fri Nov 4 22:53:26 CST 1988) ready.
Name (chemistry:bonnie):
331 Password required for bonnie.
Password:
230 User bonnie logged in.
ftp> dir
biology:200 PORT command successful.
biology:150 Opening data connection for /bin/ls (84.0.194.5,1065) (0 bytes)
total 10
drwxr-x---   2 bonnie   grpA       288 Nov 28 14:25 X
drwxrwxrwx   2 bonnie   grpA       160 Dec  5 11:19 bin
drwxr-x---   2 bonnie   grpA       480 Nov  9 12:41 ip
drwxr-x---   2 bonnie   grpA       448 Aug 16 17:17 perf
drwxr-x---   2 bonnie   grpA       992 Oct  6 09:30 sim
drwx------   2 bonnie   grpA       544 Nov 11 18:50 socket
drwxr-x---   3 bonnie   grpA       384 Dec  6 13:47 src
drwxr-xr-x   2 bonnie   grpA       128 Aug 25 09:42 stress
drwxr-x---   6 bonnie   grpA       480 Dec  9 12:35 temp
drwxr-x---   9 bonnie   grpA       480 Oct 18 16:18 test
biology:226 Transfer complete.
595 bytes received in 0.038 seconds (15 Kbyte/s)
ftp> proxy dir
chemistry:200 PORT command okay.
chemistry:150 Opening data connection for /bin/ls (84.0.194.5,1066) (0 bytes)
total 40
drwxr-xr-x   2 bonnie   grpA        512 Aug  5 16:27 conf
drwxr-xr-x   2 bonnie   grpA        352 Aug 17 11:07 csim
-rwxr-xr-x   1 bonnie   grpA     132056 Nov 28 07:02 mkhsxdev
-rw-r-----   1 bonnie   grpA       7950 Nov 28 07:01 mkhsxdev.c
drwxr-xr-x   2 bonnie   grpA        576 Nov  2 09:48 perf
drwxr-xr-x   3 bonnie   grpA         96 Dec  2 12:34 src
drwxr-xr-x   2 bonnie   grpA        128 Sep 22 09:29 stats
chemistry:226 Transfer complete.                                    (continued)
```

```
432 bytes received in 0.083 seconds (5.1 Kbyte/s)
ftp> proxy binary
chemistry: Type set to I.
ftp> proxy get X/benchmark.tar X/benchmark.tar
chemistry: 227 Entering Passive Mode (128,162,62,1,18,81)
biology: 200 Type set to I.
biology: 200 PORT command.successful.
biology: 150 Opening BINARY mode data connection for X/benchmark.tar (974848
bytes)
chemistry:150 Opening BINARY mode data connection for benchmark.tar
chemistry:226 Transfer complete.
biology:226 Transfer complete.
biology:200 Type set to A.
local: benchmark.tar remote: X/benchmark.tar
ftp> clo
biology:221 Goodbye.
ftp> proxy clo
chemistry:221 Goodbye.
ftp> quit
$
```

Closing the `ftp` *connection*
4.2.1.9

To close the `ftp` connection, type either the `bye` or the `quit` command and press RETURN.

```
ftp> bye
221 Goodbye.
$
```

Extended `ftp` *example*
4.2.1.10

The following extended example further illustrates the use of the `ftp` utility.

Example:

In this example, Adam is the user, and his login ID is `adam`. He is currently logged in to a Cray Research system, but he must access files that are not on the local system.

- First, he types `ftp` and the remote host name `moon` and presses RETURN.

```
$ ftp moon
Connected to moon.
220 moon FTP server (Version 4.7 Sun) ready.
331 Password required for adam.
Password:
230 User adam logged in.
ftp>
```

- Next, Adam turns off the interactive prompt.

```
ftp> prompt
Interactive mode off.
ftp>
```

- Adam is now ready to copy the files from host `moon`. Because all of the file names begin with the prefix `graphix.`, Adam uses file name globbing to copy all of the files at once.

```
ftp> mget graphix.*
200 PORT command okay.
150 Opening data connection for graphix.1 (84.0.194.5,1066)(45 bytes).
226 Transfer complete.
local: graphix.1 remote: graphix.1
45 bytes received in 0.16 seconds (0.27 Kbyte/s)
200 PORT command okay.
150 Opening data connection for graphix.2 (84.0.194.5,1067)(45 bytes).
226 Transfer complete.
local: graphix.2 remote: graphix.2
45 bytes received in 0.29 seconds (0.15 Kbyte/s)
200 PORT command okay.
150 Opening data connection for graphix.3 (84.0.194.5,1068)(45 bytes).
226 Transfer complete.
local: graphix.3 remote: graphix.3
45 bytes received in 0.49 seconds (0.09 Kbyte/s)
ftp>
```

- Adam closes the connection with host `moon`.

```
ftp> close
221 Goodbye.
ftp>
```

- Next Adam opens a connection with host `saturn`.

```
ftp> open saturn
Connected to saturn.
220 saturn FTP server (Version 4.7 Sun) ready.
331 Password required for adam.
Password:
230 User adam logged in.
ftp>
```

- Adam wants to copy a file called `oct.10` on the Cray Research system to directory `results` on host `saturn`. To do this, he uses the `ftp` command `cd` to change from his home directory on `saturn` to the `results` directory, and then he copies the file, as follows:

```
ftp> cd results
200 CWD command okay.
ftp> put oct.10
200 PORT command okay.
150 Opening data connection for oct.10 (84.0.194.5,1066).
226 Transfer complete.
local: oct.10 remote: oct.10
232 bytes sent in 0.0011 seconds (2e+02 Kbyte/s)
ftp>
```

- Adam wants to return to his home directory on `saturn`; however, he does not remember the name of the appropriate command, so he types `help` and presses RETURN.

```
ftp> help

Commands can be abbreviated.  Commands are as follows:

!       cr          macdef      proxy       send
$       delete      mdelete     sendport    status
account debug       mdir        put         struct
append  dir         mget        pwd         sunique
ascii   disconnect  mkdir       quit        tenex
bell    form        mls         quote       trace
binary  get         mode        recv        type
bye     glob        mput        remotehelp  user
case    hash        nmap        rename      verbose
cd      help        ntrans      reset       ?
cdup    lcd         open        rmdir
close   ls          prompt      runique

ftp>
```

Adam types `cdup` and then lists the names of the files in his directory to make sure that there is no other file he must access. Finally, he closes the `ftp` connection.

```
ftp> cdup
200 CWD command okay.
ftp> ls
200 PORT command okay.
150 Opening data connection for /bin/ls (84.0.194.5,1068)
   (0 bytes).
hypothesis
results
226 Transfer complete
147 bytes received in 0.24 seconds (0.61 Kbyte/s)
ftp> bye
221 Goodbye
$
```

`ftp` *commands*
4.2.2

The following lists the `ftp` commands and describes their functions. You can abbreviate command names to the minimum number of characters required to identify the command uniquely. In the following list, brackets enclose the part of a command name that you can omit.

`!` [*command* [*arguments*]]

Returns you to the shell on the Cray Research system. To return to `ftp` from the shell, press CONTROL-d. You may specify a UNICOS command as an argument to `!`. If you do this, `!` executes your command in the Cray Research system's UNICOS shell and then returns you to `ftp` command mode.

`?` [*ftp_command*]

(Synonymous with the `help` command.) Displays a list of valid `ftp` commands. You can then request help information for a specific command. If you specify a valid `ftp` command as an argument to `?`, `ftp` displays help information on that command.

`$` *macro-name* [*arguments*]

Executes the macro *macro-name*, which was defined with the `macdef` command. Arguments are passed to the macro unglobbed.

<INTERRUPT>

Pressing the interrupt key terminates any `ftp` operation in progress, or, if no operation is in progress, terminates `ftp` itself. You can redefine the interrupt key; by default, it is CONTROL-c.

`account` [*password*]

Supplies a supplemental password required by a remote system for access to resources after a login completes successfully. If you do not specify an argument on the command line, you are prompted for an account password; the password you enter is not echoed on the screen.

`allo` [*argument*]

Specifies whether the `ftp allo` and `ftp size` commands will be sent before `put` and `get` commands are executed. The `ftp allo` command informs the server of the size of the file that will be sent by using the `put` command so that it can preallocate the disk space, if necessary. The `ftp size` command inquires about the size of a file that will be retrieved by using the `get`(1) command, so that the disk space can be preallocated before the file is fetched.

`append` *lfile* [*rfile*]

Appends *lfile* to *rfile*; *lfile* is a valid file name on the local host, and *rfile* is a valid file name on the remote host. If you do not specify a file name for *rfile*, *rfile* is given the same file name as *lfile*. If *rfile* does not exist on the remote host, a new file is created.

`ascii`	Sets the file transfer type to ASCII; ASCII is the default.
`bell`	Toggles the terminal bell setting. Turning on the bell causes the terminal bell to ring when a file transfer is completed. The bell is turned off by default.
`binary`	Sets the file transfer type to binary.
`bye`	(Synonymous with the `quit` command.) Closes the connection to the remote host and exits from the `ftp` program.
`case`	Toggles `mget` uppercase and lowercase ID mapping. The default setting is off.
`cd` *rdir*	Changes the working directory on the remote host to *rdir*; *rdir* is a valid working directory on the remote host. The initial working directory on the remote system is the home directory of the user.
`cdup`	Changes the remote host's working directory to the parent of the working directory.
`chmod` *remote–file*	Changes file permissions on a remote file.
`clear`	Sets the file protection level to clear text for file transfer. All commands sent over the control channel are protected by cryptographic checksum until a different protection level is specified.
`close`	Closes the connection to the remote host but does not exit the `ftp` program. After you are disconnected from the remote host, you can open a connection to a new host by using the `open` command.
`copybuf` [*argument*]	Sets the copy buffer size. This buffer is used for reading and writing between the data socket and the source or destination file. `ftp` automatically chooses a copy buffer size; however, you can alter the selection. `copybuf` takes an optional argument. An argument of `off` turns off copy buffer sizing and the buffer defaults to the size specified in the `tcp_config.h` file by the `COPYBUFSIZE` variable. An argument of `auto` returns the buffer sizing to automatic. A numeric argument sets the buffer to that size. The letter `K` or `k` can follow a numeric buffer size to specify a multiple of 1024. Without an argument, `copybuf` shows the current copy buffering. The default setting is `auto`.

`cr`	Toggles carriage return stripping during ASCII-type file retrieval. The default setting is on.
`debug`	Toggles debugging mode. When debugging is on, `ftp` prints each command sent to the remote machine, preceded by the string —>. The default setting is off.
`delete` *rfile*	Deletes *rfile*; *rfile* is the name of a file on the remote host.
`dir` [*rdir*] [*lfile*]	Writes a listing of all file names in directory *rdir* (*rdir* is a valid working directory on the remote host) to *lfile* (*lfile* is the name of a file on the local host). If you do not specify *rdir*, `dir` lists the contents of your working directory on the remote machine when the command is executed. If you do not specify *lfile*, `dir` displays the listing on the standard output device. The listing that this command produces is in long format and includes the following information about each file: file name or names, length, owner, access permissions, and the date and time of last modification.
`disconnect`	An alias for the `close` command.
`form`	Sets file transfer format to `nonprint`.
`fullbuf`	Toggles the use of full buffers on write-to-disk functions. During a `get` operation, `ftp` fills the copy buffer with reads from the data socket before writing the contents of the buffer to the destination file. You can use the `fullbuf` command to disable this feature.
`get` *rfile* [*lfile*]	(Synonymous with the `recv` command.) Copies *rfile* into *lfile*; *rfile* is the name of a file on the remote host, and *lfile* is the name of a file on the local host. The transfer is done in ASCII mode unless another mode (for example, binary) was specified. If you do not specify *lfile*, `get` copies *rfile* into a new file called *rfile* on the local host. If you do not specify either file name, `get` prompts for both the remote and local file names. If verbose mode is on (see the `verbose` command in this list), various statistics, including the number of bytes transferred, the elapsed time in seconds, and the effective baud rate, are sent to the standard output device when the file transfer is complete. (By default, the standard output device is usually your terminal.)

`glob` — Toggles the status of file globbing. *Globbing* is the process of expanding ambiguous file references. With file name globbing enabled, each local file or path name is processed for the shell metacharacters `*`, `?`, `[`, and `]`; the metacharacters are expanded to match file names. With globbing disabled, all local file and path names are treated literally. File globbing is turned on by default unless you specify the `-g` option on the `ftp` command line; globbing is always on with reference to remote files. Globbing affects only the "m" commands (`mget`, `mput`, and so on).

`hash` — Toggles the status of `hash`. Turning `hash` on causes a # to be displayed on the standard output each time a packet is sent or received during a file transfer. This feature is turned off by default.

`help` [*command*] — (Synonymous with the `?` command.) Prints help information about *command*; *command* is a valid `ftp` command. If you do not specify *command*, `ftp` prints a list of valid commands.

`idle` [*seconds*] — Gets or sets idle timer on the remote side.

`image` — Synonymous with the `binary` command. Sets the file transfer type to binary.

`lcd` [*ldir*] — Changes the working directory on the local host to *ldir; ldir* is a valid working directory on the local host. If you do not specify *ldir*, the user's home directory is used by default.

`ls` [*rdir*] [*lfile*] — Retrieves a listing of the names of all files in directory *rdir*; *rdir* is a valid working directory on the remote host. The command puts the listing into file *lfile* on the local host. If you do not specify *rdir*, `ls` lists the working directory. If you do not specify *lfile*, the listing is displayed on the standard output device, which is usually your terminal. The default listing format resembles the short format of the `ls`(1) command.

`macdef` *macro-name* — Defines a macro. Subsequent lines are stored as the macro *macro-name*. A null line indicates the end of the macro definition.

`mdelete` *rfile* [*rfile2...*] — Deletes files on the remote host; each *rfile* is the name of a file on the remote host. You can specify any number of remote file names (separated by spaces) on the command line.

`mdir` *rfile* [*rfile2...*] *lfile*	Obtains a directory listing of *rfile* and places it in *lfile*; *rfile* is the name of a file on the remote host, and *lfile* is the name of a file on the local host. The listing that this command produces is in long format and includes file name(s), length, owner, access permissions, and the date and time of last modification. You can specify any number of remote file names (separated by spaces) on the command line. You must specify *lfile*.
`mget` *rfile* [*rfile2* ...]	Retrieves a copy of file *rfile* from the remote host and places it in a file of the same name in the working directory on the local host. You can specify any number of remote file names (separated by spaces) on the command line.
`mkdir` *rdir*	Creates a directory called *rdir* on the remote host.
`mls` *rdir1* [*rdir2* ...] *lfile*	Obtains a listing of directory *rdir1* and places it in *lfile*; *rdir1* is the name of a directory on the remote host, and *lfile* is the name of a file on the local host. You can specify any number of remote directory names (separated by spaces) on the command line. The format of the listing generated by this command resembles the short format of the `ls`(1) command. You must specify *lfile*.
`mode`	Sets file transfer mode to stream.
`modtime` *remote-file*	Shows last modification time of a remote file.
`mput` *lfile1* [*lfile2* ...]	Puts a copy of file *lfile1* in a file of the same name in the working directory on the remote host. *lfile* is the name of a file on the local host. You can specify any number of local file names (separated by spaces) on the command line.
`newer` *remote-file* [*local-file*]	Gets file if the remote file is newer than the local file.
`nlist` [*remote-directory* [*local-file*]]	Lists the contents of the remote directory.
`nmap` [*inpattern outpattern*]	Sets or unsets file name mapping. If you omit arguments, the file name mapping mechanism is unset. If you specify arguments, remote file names are mapped during the execution of `mput` commands and `put` commands that were issued without a specified remote target file name. Similarly, local file names are mapped during the execution of `mget` commands and `get` commands that were issued without a specified local target file name.

`ntrans` [*inchars outchars*]	Sets or unsets the file name character translation. If you omit arguments, the file name character translation mechanism is unset. If arguments are specified, characters in remote file names are translated during the execution of `mput` commands and `put` commands that were issued without a specified remote target file name. Similarly, characters in local file names are translated during the execution of `mget` commands and `get` commands that were issued without a specified local target file name.
`open` *rhost*	Opens a connection to *rhost*; *rhost* is a valid name, alias, or Internet address of a remote host.
`private`	Sets the file protection level to `private` for the control channel and file transfer. This command is not available outside of the United States and Canada. All messages sent on the control and data channels are protected by encryption.
`protect [clear | safe | private]`	Sets the protection level for file transfer. The `private` option is not available outside of the United States and Canada.
`prompt`	Toggles the status of interactive prompting. Enabling prompting causes all `mget`, `mput`, and `mdelete` commands to prompt with the names of the files in the working directory. If you respond to this prompt with an `n` or `N`, the specified command skips the file you select. Responding with anything else causes the file to be processed. If your respond with anything other than a `y`, the file is skipped. If prompting is turned off, commands process all files. Prompting is turned on by default unless you specify the `-i` option on the `ftp` command line.
`proxy` *ftp-command*	Executes an `ftp` command (*ftp-command*) on a secondary control connection. Allows simultaneous connection of two remote `ftp` servers for transferring files between servers.
`put` *lfile* [*rfile*]	(Synonymous with the `send` command.) Copies *lfile* into *rfile*; *lfile* is the name of a file on the local host, and *rfile* is the name of a file on the remote host. The transfer is done in ASCII mode unless you specified another mode (for example, binary). If you omit *rfile*, `put` copies *lfile* into a file called *lfile* on the remote host. If verbose mode is on, various statistics (including the number of bytes transferred, the elapsed time in seconds, and the effective baud rate) are printed to the standard output device when the file transfer is complete.
`pwd`	Prints the name of the working directory on the remote host.

`quit`	(Synonymous with the `bye` command.) Closes the connection to the remote host and exits from the `ftp` program.
`quote` *arg1 arg2...*	Passes *arg1* directly to the remote host, without parsing it on the local host. This lets you use `ftp` commands that are implemented by the remote host but not by the local host. To obtain the list of `ftp` commands that the remote host supports, use the `remotehelp` command.
`rawbuf`	Toggles the use of raw I/O on write-to-disk and read-from-disk functions. Usually, the system overhead is lower when raw I/O is used. The default is that raw I/O is enabled.
`recv` *rfile* [*lfile*]	(Synonymous with the `get` command.) Copies *rfile* into *lfile*; *rfile* is the name of a file on the remote host, and *lfile* is the name of a file on the local host. The transfer is done in ASCII mode unless you specify another mode (for example, binary). If you do not specify *lfile*, `recv` copies *rfile* into a new file called *rfile* on the local host. If you do not specify either file name, `recv` prompts for both the remote and local file names. If verbose mode is on (see the command `verbose` in this list), various statistics, including the number of bytes transferred, the elapsed time in seconds, and the effective baud rate, are sent to the standard output device when the file transfer is complete. (By default, the standard output device is usually your terminal.)
`reget` *remote-file* [*local-file*]	Gets file restarting at the end of the local file.
`rename` [*from*][*to*]	Renames the file *from* on the remote host to the file *to*.
`reset`	Clears the reply queue.
`restart` *bytecount*	Restarts the file transfer at *bytecount*.
`rhelp` [*command*]	Displays help information about *command* as it is implemented on the remote host; *command* must be a valid command on the remote host. If you omit *command*, `help` displays all the `ftp` commands that the remote host recognizes.
`rmdir` *rdir*	Deletes directory *rdir*; *rdir* is a valid directory on the remote host.
`rstatus`	Shows the status of the remote machine.

`runique` — Receive unique. Provides a unique file name when a file with the same name as the target local file for a `get` or `mget` command already exists. A `.1` is appended to the name, unless this matches another existing file. The process continues up to `.99`; at that point, an error message appears, and the transfer does not occur. This facility is turned off by default.

`safe` — Sets the file protection level to `safe` for the control channel and file transfer. All messages sent on the control and data channels are protected by cryptographic checksum.

`send` *lfile* [*rfile*] — (Synonymous with the `put` command.) Copies *lfile* into *rfile*; *lfile* is the name of a file on the local host, and *rfile* is the name of a file on the remote host. The transfer is done in ASCII mode unless another mode (for example, binary) was specified. If you omit *rfile*, `send` copies *lfile* into a file called *lfile* on the remote host. If verbose mode is on, various statistics (including the number of bytes transferred, the elapsed time in seconds, and the effective baud rate) are printed to the standard output device when the file transfer is complete.

`sendport` — Toggles the use of the `PORT` protocol command for each data connection.

`showbuf` — Toggles display of copy buffer and socket buffer sizing information during a transfer. Default is no display of information.

`site` *arguments* — Sends *arguments* to the remote `ftp` server as arguments to an `ftp site` command. In return, one `ftp` reply code is expected.

`size` *remote-file* — Shows the size of the remote file.

`sockbuf` [*argument*] — Sets the socket buffer size. This kernel buffer is used for data transfer on the data socket. `ftp` automatically chooses a socket buffer size; however, you can alter the selection. `sockbuf` takes an optional argument. An argument of `off` turns off socket buffer sizing and the buffer defaults to the kernel's default socket buffer size. An argument of `auto` returns buffer sizing to automatic. A numeric argument sets the buffer to that size. The letter `K` or `k` can follow a numeric buffer size to specify a multiple of 1024. Without an argument, `sockbuf` shows the current socket buffering. The default setting is `auto`.

`status` — Shows the current connection status, transfer mode, messages, and the settings for `hash`, `bell`, and other options of the current `ftp` session.

`struct` [*struct-name*] — Sets the file transfer structure to *struct-name*. By default, file structure is used. Currently, only the default is supported.

`sunique` — Send unique. Provides a unique file name when a file with a name equal to the target remote file name for a `put` or `mput` command already exists. A `.1` is appended to the name, unless this matches another existing file. The process continues up to `.99`; at that point, an error message appears, and the transfer does not occur. This facility is off by default.

`system` — Shows the remote system type.

`tenex` — Sets the file transfer type to the type needed to talk to TENEX machines.

`trace` — Toggles packet tracing. The default setting is off.

`type` [*type–name*] — Sets the file transfer type to *type-name*. If *type-name* is not specified, the current type is printed. Valid types are `tenex`, `binary`, and `ascii`; `ascii` is the default.

`umask` *mask* — Gets and sets `umask` to *mask* on the remote side.

`user` *login_name* — Initiates the login process on the remote host; *login_name* is a valid login name on a remote host. The `ftp` program prompts you for your password to complete the login process.

`verbose` — Toggles verbose mode. When verbose mode is on, all responses from the `ftp` server are displayed on your standard output device. Verbose mode is turned on by default.

`winshift` [*value*] — Gets or sets the value for the TCP window shift option to be used on data connections. If no argument is specified, this option reports the current value. If *value* is `off`, the option is disabled; if *value* is `on`, the option is enabled with a value of 4; if *value* is an integer value between 0 and 14, the TCP window shift option is enabled with that value. Because the client side of an `ftp` always performs a passive option operation, you do not have to disable the sending of the TCP window shift option. If the incoming `SYN` packet does not contain the option, none will be sent in the `SYN,ACK` packet, regardless of whether the application has enabled the option.

Using the `tftp` utility

4.3

The `tftp` utility is a very limited file access mechanism. Because this utility poses a security threat to the system, the system administrator might choose to turn it off. Contact your system administrator for information about the availability of and access permissions for `tftp`. Following is an example of `tftp` use.

```
$ tftp
tftp> connect chemistry
tftp> get /usr/bonnie/courses
Received 214 bytes in 0 seconds.
tftp> quit
$
```

For more information, see the `tftp`(1B) man page.

Communicating Across the Network [5]

The following utilities are available for communicating across a TCP/IP network:

- `mail`(1) and `mailx`(1)

  The `mail` and `mailx` utilities let you send and receive mail. When you send mail, `mailx` enables you to edit, review, and perform other modifications to the message as you enter it. When you read mail, both `mail` and `mailx` let you save, delete, and respond to messages.

- `talk`(1B)

  The `talk` utility enables you to communicate with another user. This utility copies lines from your terminal to that of the other user.

The following subsections explain how to use these TCP/IP utilities.

Using the `mail` and `mailx` utilities

5.1

This subsection explains how to send mail messages and how to receive (or read) mail messages. The `mail` and `mailx` utilities are used for both sending and receiving.

> **Note:** For more information about `mail` and `mailx` if you have a UNICOS MLS system or a Trusted UNICOS system, see subsection 8.3, page 112.

Note that you cannot trust mail “From” lines. The mail system is designed to deliver mail worldwide across a huge collection of systems that cannot be trusted. In most cases, you can count on the mail you receive to be from the person whose name is in the header, but this is not guaranteed. It is relatively simple to forge mail when it is sent across the network. You should always verify through an alternative communication method the source and context of any sensitive mail message.

Sending mail messages 5.1.1

To send mail to a user on your local host, use the following formats:

```
mail login_name
  mailx login_name
```

login_name Valid login name on the local host

When sending mail to a user on the network, you can specify the network address by adding an at-sign (@) and the remote host name to *login_name*. You can specify more than one addressee for the same message by separating the network addresses with a space. Use the following format:

```
mail login_name[@rhost] ...
  mailx login_name[@rhost] ...
```

login_name Valid login name on the remote host.

@rhost Official host name, alias, or Internet address of a remote host; if @ is not included, `mail` and `mailx` do not recognize the host name.

After you press RETURN, `mailx` prompts you for a message subject. Enter a line identifying the subject of the message, then go to a new line to begin the text of your message. The `mail` command does not prompt for message subject; simply begin the text of your message. To end the message, go to a new line and enter CONTROL-d.

Unlike the `mail` command, `mailx` has associated commands. You can enter `mailx` commands while you are creating the message by beginning a line with the tilde (~) escape character, followed by one command letter and optional arguments. See the `mailx`(1) man page for a summary of these commands. Two of the most useful tilde escape commands, which you can use in `mailx` input mode, are `~r` and `~v`. These commands are described in the following list.

`~r` *filename* Reads the file called *filename* into the body of the message.

`~v` Lets you edit the current message by using a text editor; the default text editor is `vi`(1).

Example:

In this example, the user addresses mail to the user with the login name `pattie` on the remote host `cray2`, using the network mail address `pattie@cray2`. The system prompts for a subject on the following line. The user enters the subject, then begins the body of the message on the next line. At the end of the message, the user goes to a new line and presses CONTROL-d .

The EOT (end-of-transmission) message appears on the screen, followed by the shell prompt.

```
$ mailx pattie@cray2
Subject: Sending mail
I'm practicing sending mail, so tell me if you get this.
CONTROL-d
EOT
$
```

Reading mail messages
5.1.2

To read incoming mail, enter either `mail` or `mailx` without any arguments, as shown in the following formats:

```
mail
  mailx
```

The `mail` and `mailx` commands let you change the way incoming mail is displayed and stored. They are described on the `mail`(1) and `mailx`(1) man pages.

Using the `talk` utility
5.2

The `talk` utility is a visual communication program that copies lines from your terminal to that of another user.

> **Note:** The `talk` command is not supported on a Trusted UNICOS system. The `talk` command allows communication only between users who are at the same security label in a UNICOS MLS system. See the *UNICOS Multilevel Security (MLS) Feature User's Guide*, publication SG–2111, for more information.

This command has the following format:

```
talk address [terminal]
```

address	User to whom you are sending the message
terminal	Terminal name to be used if the user is logged in on more than one terminal

The message recipient is notified that you are sending a message and is instructed to return a response that opens the connection. You then type your message. You and the recipient can communicate simultaneously if desired. To exit the `talk` facility, type an interrupt character. The cursor then moves to the bottom of the screen, and the command restores the terminal.

Example:

In this example, your network mail address is `curtis@host4`. You are going to send a message to the user at network mail address `jerry@host4`.

```
$ talk jerry@host4
```

The recipient receives the following message:

```
Message from TalkDaemon@host4 at 13:06...
talk:  connection requested by curtis@host4
talk:  respond with: talk curtis@host4
```

You receive the message:

```
Connection established
```

You can then begin typing your message:

```
Shall we get together for lunch today?
```

The recipient can type a reply, and the communication proceeds until you or the recipient types an interrupt character.

Displaying Host and User Information [6]

This section describes the TCP/IP utilities you can use to display host and user information about the network. Table 3 summarizes the utilities.

Table 3. Functions of TCP/IP utilities for information display

Utility	Function
`finger`(1B)	Lists information about one user or all users on the host.
`hostname`(1)	Prints the name of the local host system.
`ping`(8)	Checks connections to other TCP/IP hosts in the network.

Using the `finger` utility

6.1

When no option or operand is specified after the `finger` command, the `finger` utility displays the following information about each user currently logged in to the local host:

- Login name
- Full name
- Write status (given as an asterisk (*) before the terminal name if write permission is denied)
- Terminal name
- Idle time (represented in minutes if it is a single integer, hours and minutes if a colon (:) is present, or days and hours if a `d` is present in the date field)
- Login time

When you specify the appropriate option or operand after the `finger` command, the `finger` utility displays a longer list. The list can include all of the information in the preceding list as well as the following items:

- User's home directory and login shell.
- Any plan that the user has placed in file `.plan` in his or her home directory, and the project on which he or she is working from the file `.project`, also in the home directory; for `.project` file, text must begin on line 1 of the file for `finger` to read it.

  Note: To access the listed information in a UNICOS MLS system, your security label must dominate the security label of the information. See the *UNICOS Multilevel Security (MLS) Feature User's Guide*, publication SG–2111, for more information.

  A `nobody` account must exist on the system also. However, the label range of the account may not allow `finger` to be used across the network. See your system administrator to find out if you can use `finger`.

You can specify one or more user names on the `finger` command, which causes it to print information about each user specified. The `finger` command accepts account (login) names, as well as first and last names.

Invoke the `finger` command with the following format:

```
finger [-b] [-f] [-h] [-i] [-l] [-m] [-p] [-q] [-s] [-w] [name . . .]
```

`-b`	Gives a briefer long-form list of users.
`-f`	Suppresses the heading in the short and quick output format requested by the `-s` or `-q` option.
`-h`	Suppresses printing of the `.project` file.
`-i`	Gives the same quick list as the `-q` option, but includes idle time.
`-l`	Gives the long-form list; this is the default mode.
`-m`	Matches arguments only for login name.

`-p` Suppresses printing of the `.plan` file.

`-q` Gives a quick list that includes only login name, terminal name, and login time.

`-s` Gives a short-form list of users.

`-w` Suppresses printing of the full name in the short-form list format.

name Specifies the login name or the full name of a validated user on the local host. You can specify any number of users; separate names with blanks.

Example 1:

In this example, the `finger` command is entered without arguments or options. It lists the login name, full name, terminal name, write status, idle time, login time, and office location for each current user on the local host.

```
$ finger

Login    Name       TTY         Idle      When         Office
mpd      MelDio     tty00       41        Thu 17:24    14
karen    K.Jone     ttyp2       2:05      Thu 16:24    2
mpd      MelDio     *ttyq1      4:35      Thu 13:03    14
```

Example 2:

In this example, the `finger` command is followed by a login name (`mpd`). In response, `finger` lists the login name, full name, terminal name, shell, write status, idle time, login time, and plan for user `mpd` on the local host. By default, the listing is displayed in long format.

```
$ finger mpd

Login name: mpd         In real life: Mel Dio
Directory: /u1/mpd      Shell: /bin/sh
On since Dec 19 17:24:42 on ttyp000 from
mywork
2 hours 16 minutes Idle time
Plan:
To make a plan by Tuesday
```

Using the hostname utility
6.2

The hostname utility prints the name of the current host system. Systems personnel can use the hostname command to set the name of the host system.

To invoke the hostname utility, enter the following command:

```
hostname
```

Example:

```
$ hostname
crayhost
```

Using the ping utility
6.3

The ping(8) utility provides a simple way to find out whether a host connected to the TCP/IP network is functioning. It also gives you some idea of the performance you can expect when communicating with the specified host. The ping utility sends out an ICMP ECHO_REQUEST packet (called a *ping*) to a host or gateway; it then waits for an ICMP ECHO_RESPONSE packet. Because every IP host is required to respond to ICMP packets, the absence of a response indicates that the host is not up. If a response is received, the amount of time it takes to get the response indicates the speed with which the host is responding.

The command format is as follows:

```
/etc/ping [-drv host [packetsize [count]]]
```

`-d`	Sets the socket-level debugging option (`SO_DEBUG`).
`-r`	Sets the `SO_DONTROUTE` option to bypass the routing tables.
`-v`	Turns on the verbose mode for packet descriptions; tells you whether you receive a packet other than the one the system expected. By default, verbose mode is off.
host	The official host name or Internet address of a remote host whose status you want to check.
packetsize	Specifies the number of bytes in the packet. Values can range from 0 to 4096; the default length is 64 bytes.
count	Specifies that the `ping` utility should send the echo packet *count* times. If *count* is 0 or not specified, the default is one packet per second until responses are no longer sent or until you stop `ping` with a `SIGINT` signal.

The `ping` utility sends one datagram per second and displays one line of output on your screen for every `ECHO_RESPONSE` returned. If no response is received, no lines are displayed. The `ping` utility also computes round-trip times and packet-loss statistics. These are displayed when all responses are received; when the program reaches the maximum number specified in *count*; or if the program is terminated with a `SIGINT` signal.

If you have not specified the maximum number of datagrams in the *count* operand, or if the host to which you are sending a ping goes down, the `ping` utility can loop forever unless you end the process by telling the shell to send a `SIGINT` signal. Do this by entering the user-definable character that sends the signal (usually `CONTROL-C`).

An example of `ping`, using the default settings, follows. The output shows that the remote host, Internet address `128.1.0.1`, is responding to the local host. The statistics summary shows the number of packets transmitted, the number of packets received, the percentage of packets lost, and a summary of the response time.

```
$ /etc/ping 128.1.0.1
PING 128.1.0.1: 56 data bytes
64 bytes from 128.1.0.1: icmp_seq=0. time=7. ms
64 bytes from 128.1.0.1: icmp_seq=1. time=12. ms
64 bytes from 128.1.0.1: icmp seq=2. time=16. ms
64 bytes from 128.1.0.1: icmp_seq=3. time=6. ms
64 bytes from 128.1.0.1: icmp_seq=4. time=7. ms
----128.1.0.1 PING Statistics----
5 packets transmitted, 5 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 6/9/16
```

The following example shows output with a packet size of 4096 bytes and a count of 10, with 1 packet dropped.

```
$ /etc/ping  128.1.0.1  4096  10
PING 128.1.0.1: 4069 data bytes
4096 bytes from 128.1.0.1: icmp seq=0. time=41. ms
4096 bytes from 128.1.0.1: icmp seq=1. time=22. ms
4096 bytes from 128.1.0.1: icmp seq=3. time=32. ms
4096 bytes from 128.1.0.1: icmp seq=4. time=31. ms
4096 bytes from 128.1.0.1: icmp seq=5. time=32. ms
4096 bytes from 128.1.0.1: icmp seq=6. time=30. ms
4096 bytes from 128.1.0.1: icmp seq=7. time=27. ms
4096 bytes from 128.1.0.1: icmp seq=8. time=34. ms
4096 bytes from 128.1.0.1: icmp seq=9. time=27. ms

----128.1.0.1 PING Statistics----
10 packets transmitted, 9 packets received, 10% packet loss
round-trip (ms)  min/avg/max = 22/31/41
```

The last example shows the `ping` output when a remote host is not responding.

```
$ /etc/ping 192.0.0.1 4096 10
PING 192.0.0.1: 4069 data bytes

----192.0.0.1 PING Statistics----
10 packets transmitted, 0 packets received, 100% packet loss
```

Network Authorization [7]

This section describes security in the UNICOS TCP/IP environment. It focuses on actions users should take to prevent malicious intrusion into the system. The following topics are discussed:

- The autologin feature
- Authorization files
- Solving authorization problems

Using autologin through Kerberos is discussed in the *Kerberos User's Guide*, publication SG–2409.

The autologin feature

7.1

UNICOS supports *autologin*, a feature that lets a user log in automatically across the network to an account that belongs to that user or to another user. This feature might be restricted when the UNICOS multilevel security (MLS) feature is enabled. See subsection 8.2.4, page 104, for more information. Check with your security administrator. The following files support autologin:

- The `.rhosts` and `/etc/hosts.equiv` files for incoming `rlogin`(1B), `rsh` (see `remsh`(1B)), and `rcp`(1) commands (this applies only to TCP/IP networks)
- The `.netrc` file for the outgoing `ftp`(1B) command and the `rexec`(3) library routine (for TCP/IP)

Although autologin is very convenient, it does present a major security threat to the system, and the following precautions should be taken when creating the `.rhosts` and `.netrc` files.

Precautions for the `.rhosts` file:

- Ensure that only the owner can read or write to the file.
- Ensure that the file contains only those hosts that are needed.
- Do not use wildcard characters.

Precautions for the `.netrc` files:

- Ensure that only the owner can read or write to the file.
- Do not put passwords in the `.netrc` file. The password parameter is for facilitating anonymous `ftp` and the `rexec`(3) library routine. If you need more information on anonymous `ftp`, contact your system administrator.

For information on setting up the `.rhosts` and `.netrc` files, see the following subsection.

Authorization files 7.2

Authorization files contain host and user information that is verified by the system before user privileges are granted on a remote system. These files can be used to ensure system security by limiting access to directories or to the UNICOS system. You can create the `.rhosts` and `.netrc` files; the system administrator can create the `/etc/hosts.equiv`, `/etc/ftpusers`, and `/etc/shells` files. In a UNICOS MLS system, the system administrator has additional control with the network access list (NAL) and workstation access list (WAL) in the `/etc/config/spnet.conf` file. The TCP/IP `telnet` utility does not use authorization files. The following list describes the authorization files:

File name	Description
`/etc/ftpusers`	System file that lists users who are prohibited from accessing a system by using the `ftp` program. If the system administrator has not set up an `ftpusers` file, or the file is empty, all valid UNICOS users can use `ftp`.
`/etc/hosts.equiv`	System file that lists equivalent host and alternative user names used during autologin. Direction is inbound. Associated commands are `rlogin`, `rcp`, and `rsh`.
`/etc/shells`	System file that lists valid login shells for inbound `ftp` users. If a UNICOS user's login shell is not in this file, the user is denied `ftp` access.

File name	Description
`$HOME/.netrc`	User file that lists autologin information for `ftp` and `rexec`(3) requests. Direction is outbound.
`$HOME/.rhosts`	User file that lists remote host names and login names of users who are allowed access to your home directory. Used during autologin. Direction is inbound. Associated commands are `rlogin`, `rcp`, and `rsh`.
`/etc/config/spnet.conf` (network access list)	On a UNICOS MLS system, the NAL defines the label range allowed for each connection to and from a remote host. This might prevent a user from connecting to or from specific remote hosts, or might prevent access at specific labels. The NAL also defines the level of trust for each node in a Trusted UNICOS system. (The NAL class must be C1 or higher to allow automatic authorization from that node. The class must be B1 or higher to allow connections at more than one label. See subsection 8.1.2 page 92, for more information.
`/etc/config/spnet.conf` (workstation access list)	On a UNICOS MLS system, the WAL defines the remote services allowed for specified users on specified remote nodes. This might prevent a remote copy, or remote login, for example. See subsection 8.1.5, page 95, for more information.

The `.rhosts` *and* `/etc/hosts.equiv` *files* 7.2.1

The `.rhosts` file in your home directory and the `/etc/hosts.equiv` file that the system administrator maintains provide authorization for the `rlogin`, `rcp`, and `rsh` utilities. For both files, authorization occurs in much the same

manner: both files on the destination host must contain an entry (or entries) authorizing connections from the desired source host; a similar mechanism permits connections from a specific source host to be explicitly denied authorization.

Note: When the UNICOS MLS feature is enabled, the autologin capability of inbound `rlogin`, `rcp`, and `rsh` might be restricted. See subsections 8.2.4, page 104; 8.2.5, page 106; and 8.2.7, page 112, for more information, and check with your security administrator.

Authorizing connections from remote hosts by using `.rhosts` 7.2.1.1

To authorize `rlogin`, `rcp`, and `rsh` connections to your Cray Research account from remote hosts, create a `.rhosts` file in the home directory of your account on the Cray Research system and place in it entries that authorize connections from specific account names on specific remote hosts. The `.rhosts` file is a simple text file; you can use any standard UNICOS text editor (for example, `vi`) to create or modify it.

Each entry in the `.rhosts` file is a separate line of text with the following format:

```
host account optional_comment
```

Such an entry in your `.rhosts` file in the home directory of your account on the local Cray Research host authorizes the user with the account name of *account* on the remote host with name *host* to access your account on the local Cray Research system. Some implementations require *host* to be a fully qualified domain name. To operate the host and account name parts of each entry, the account name part and the optional comment, use either a space or a tab character.

If you omit the account name part of an entry in your `.rhosts` file, the authorization mechanism assumes your local account name in its place (for example, if your account name on the local Cray Research host is `andrea`, and you place an entry in your `.rhosts` file with just the host name `other`, it will authorize the identically named account name `andrea` on host `other`). You can explicitly deny authorization to all accounts on a specific system by placing a hyphen (–) in the account name part of an entry for that system, as in the following example:

host –

This specification denies authorization to any connection from the remote host called *host*. Similarly, you can explicitly deny authorization to a specific account name on all systems by placing a hyphen in the host name part of an entry for that user, as in the following example:

– user

This specification denies authorization to any account called *user*, no matter from which remote host the connection originates.

Multiple names are not permitted in the host or in the account parts of a single entry. To authorize an account name from more than one remote host, or several accounts from a single remote host, simply create multiple entries in the `.rhosts` file, as in the following `.rhosts` file example:

```
# .rhosts file
#
host1.cray.com jen  # User jen can access the local host from host1.cray.com
host2 jen   # User jen can access the local host from host2
host3 –     # No accounts from host3 can access the local host
 – ted      # User ted cannot access the local host
```

Note: Although the convenience of the `.rhosts` feature makes it tempting to include many host names in the `.rhosts` file (for example, when a connection from an otherwise seldom-used remote host is desired), serious security considerations stem from indiscriminate listing of host names in `.rhosts` files. For guidelines on how to include host names in your `.rhosts` file in a secure manner, see subsection 7.1, page 77.

Authorizing connections from remote hosts by using `/etc/host.equiv` 7.2.1.2

The system administrator of your local Cray Research host can place the names of various remote hosts in the `/etc/hosts.equiv` file. Hosts listed in this file are considered to be equivalent to the Cray Research host; any account name that is identical on the two hosts is authorized automatically for connection from the remote host to the Cray Research host through the `rlogin`, `rcp`, and `rsh` utilities. Conversely, the system administrator can specifically deny authorization to connections from certain remote hosts by placing a hyphen beside entries in the `/etc/hosts.equiv` file.

These security measures are in effect regardless of any entries in `.rhosts` files; that is, entries in the `/etc/hosts.equiv` file override individual users' `.rhosts` files. Any host listed in the `/etc/hosts.equiv` file will have its `rlogin`, `rcp`, and `rsh` connections to the Cray Research host automatically authorized, even if you try to restrict connections from such a remote host by placing a – entry for it in your `.rhosts` file. Similarly, any host with a – entry in the `/etc/hosts.equiv` file will have authorization of its `rlogin`, `rcp`, and `rsh` connections denied automatically, even if you try to authorize connections from such a remote host by placing its name in your `.rhosts` file.

Because the `/etc/hosts.equiv` file is a text file (like the `.rhosts` file), you can use any standard UNICOS utility (such as `vi`, `ed`, or `cat`) to determine whether an entry in `/etc/hosts.equiv` is impeding your attempts to permit or deny authorization by way of your `.rhosts` file. Following is a sample `/etc/hosts.equiv` file (which overrides the sample `.rhosts` file in the previous example):

```
# hosts.equiv
#
host1 ted    # Allows user ted to access the local host from host1
host2 – jen  # Allows all host2 users except jen to access local host
host3        # Allows all users on host3 to access local host
```

Note: When the UNICOS MLS feature is enabled and the configuration parameter `NETW_RCMD_COMPAT` is not set, special restrictions apply. You may have to place entries in the `/$HOME/.rhosts` file even when the `/etc/hosts.equiv` file has entries for corresponding hosts. See subsection 8.2.4, page 104, for more information.

Authorizing connections from remote hosts by using `.rhosts` 7.2.1.3

When trying to connect to a remote host by using the `rlogin`, `rcp`, or `rsh` utility, authorization is handled in a manner specific to the remote host. Therefore, you should consult with the system administrator of the remote host, or see the remote host's vendor documentation, for the correct information about authorization on that host. Nevertheless, in practice, if the remote host is running an operating system derived from the 4.3BSD operating system, you can probably authorize a connection to that host from your local Cray Research host by connecting and logging in to the remote host (using, for example, `telnet`) and then using any available text editor to enter the

name of your local Cray Research host in the `.rhosts` file in your home directory on the remote host. If this does not appear to work, consult with the system administrator or see the vendor-supplied documentation for the remote host.

The following example illustrates a sample `.rhosts` file on a Cray Research host for a user with a login name of `scott`. `scott` wants to authorize connections from his accounts with the same login name on another Cray Research host (`othercray`) and from the front-end workstation (`biology`). `scott` also is working on a project with a user with a login name of `betsy` on another workstation (`math`), and he wants to authorize connections from that account on that workstation. Finally, he wants to deny explicitly authorization to any account from the host called `chemistry` (possibly because the account called `scott` on that system belongs to a different user), and to the account called `trouble` on any host (possibly because the user of that account name is a known security risk). A `.rhosts` file to set up these authorizations might contain the following entries:

```
# sample .rhosts file on a Cray Research system
#
othercray scott # my login (scott) from our other Cray Research system
biology    # my login (scott) from my workstation
math  betsy     # while we're working on XYZ project
chemistry -     # no one from host chemistry
- trouble  # no one named "trouble"
```

The authorization lines for `othercray` and `biology` show that `scott` can authorize the identical login name on another system either explicitly by listing the login name or simply by omitting the login name part of the entry.

The `.netrc` *file* 7.2.2

You can create the `.netrc` file in your home directory on the Cray Research system to provide authorization for the `ftp` facility and `rexec`(3) library routine. When you invoke either one, the program looks for a `.netrc` file in your home directory. If the program finds this file, it uses the information contained in the file to log you in automatically to the remote host. If you do not have a `.netrc` file, you will be prompted for your login name and password.

The `.netrc` file is a simple text file; you can use any standard UNICOS text editor, such as `vi`, to create or modify it. If `.netrc` contains password or account information, `ftp` will use the file only if the file permissions are set so that only the owner of the file has read and write permissions. That means that the owner should use `chmod`(1) to set the file permissions to `600`. If the file permissions allow any other user to read and write the file, `ftp` will fail. For information on security concerns for the `.netrc` file, see subsection 7.1, page 77.

The `.netrc` file can contain one or more entries. Each entry describes default values and macros to use when connecting to a specified remote host. Each entry is made up of token pairs, which includes a keyword and a value. Each token is a string of characters, separated by a space, tab, comma, newline, or a string of characters between two double quotation marks. The backslash (\) is a special character. You can embed any of the special characters (space, tab, comma, newline, double quotation mark, or backslash) into a token by preceding it with a backslash. Usually, each entry is on a separate line.

Permissible token pairs 7.2.2.1

The recognized keywords are `machine`, `login`, `password`, `account`, and `macdef`.

A list of the known token pairs follows. The `machine` *remote_hostname* token pair defines the start of an entry. All other token pairs are optional and can be specified in any order, though they are usually given in the order that follows. You will be prompted for any information that is missing from the `.netrc` file (for example, the `password` *password* token pair) and is needed to establish a connection. The `macdef` token pair differs from the other token pairs; after the `macdef` *macro_name* token pair, all characters up to a blank line are assumed to be the definition of the macro.

Token	Description
`machine` *remote_hostname*	Identifies the name of the remote host to which a connection is to be established. The `.netrc` file is searched for a `machine` token that matches the remote host name specified on the `ftp` or `rexec` command line or as an `open` command argument. After a match is found, the subsequent `.netrc` tokens are processed until the end of the file is reached or until another `machine` token is found.
`login` *login_name*	Specifies the name of a user at the remote host. If this token is present, the autologin process logs in to the remote host by using the specified name.
`password` *password*	Specifies a password. If this token is present, the autologin process supplies the specified string when the file transfer server requires a password as part of the login process. If the `.netrc` file can be read by anyone other than the user, and this token is present in the file, the autologin process is aborted. For security purposes, clear-text passwords should not be used.
`account` *account_name*	Supplies an additional account password. If this token is present, the autologin process supplies the specified string if the file transfer server requires an additional account password.

Token	Description
`macdef` *macro_name macro*	Defines a macro for use in the `ftp` session. This token is similar to the `macdef` command of `ftp`. A macro is defined with the specified name; its contents begin with the next `.netrc` line and continue until a blank line is encountered. If a macro called `init` is defined, it is executed automatically as the last step of the autologin process.

Example of a `.netrc` *file* 7.2.2.2

This subsection shows an example of a `.netrc` file. This example contains a set of tokens for three different remote hosts. The first set indicates that, when connecting to host `biology`, you must use the login name `bonnie`. Because the password was omitted, you are prompted for the password during each login process. The second set indicates that, when connecting to the host `chemistry`, you must use the login name `bonnie2`, and it also defines two macros, `lsf` and `pwdlsf`. The third set is an entry for anonymous `ftp`. The anonymous facility is the ability to use `ftp` to access another host without having an account or password on that host. The login name for anonymous `ftp` is usually `anonymous`. The password should be a name that describes the user; however, for security purposes, you should not use clear-text passwords in `.netrc`. To identify the user, this entry contains the password `bonnie`. Usually, the anonymous facility is not enabled; when it is enabled, only a limited number of files can be accessed on that host.

```
# .netrc file example

machine biology login bonnie
machine chemistry login bonnie2
    macdef lsf
    ls -CF

    macdef pwdlsf
    pwd
    ls -CF
machine blackhole login anonymous password bonnie
```

The `/etc/shells` *and* `/etc/ftpusers` *files* 7.2.3

The system administrator maintains the `/etc/shells` file to determine what command shells can be used to access the UNICOS system. The system administrator maintains the `/etc/ftpusers` file to determine who can use `ftp` to access the UNICOS system. The `/etc/shells` file is an ASCII file that contains valid login shells; the `/etc/ftpusers` file is an ASCII file that contains login names that are not valid, one user name per line. When the `ftp` daemon is invoked, it makes the following checks for the login name of the user who is trying to gain access:

1. Determines whether the login shell of the user is listed in `/etc/shells`.
2. If the `/etc/shells` file does not exist, the daemon uses a default list of `/bin/sh`, `/bin/csh`, and `/bin/ksh`.
3. If the user's login shell is not listed, the user is denied access.
4. Checks the `/etc/ftpusers` file for the login name of the user who is trying to gain access.
5. If the name is there, `ftp` denies the user access.
6. If `/etc/ftpusers` is nonexistent or empty, all valid UNICOS users are considered valid users of `ftp`.

Following is an example of an `/etc/shells` file:

```
# /etc/shells file example
#
# Valid login shells for FTP users
#
/bin/csh
/bin/sh
/bin/ksh
/usr/lbin/oursh    # A local shell
```

An example of an `/etc/ftpusers` file follows:

```
# /etc/ftpusers file example
#
# Denied ftp and ftam users
deb
mk
adam
```

Solving authorization problems
7.3

Table 4 gives examples of possible autologin problems and solutions.

Table 4. Authorization problems and solutions

Problem	Solution
You cannot achieve autologin to a remote host.	See the remote host's vendor documentation for the proper authorization procedures on the remote host. (If the remote host is running an operating system based on 4.3BSD, the solution may be as simple as putting an entry that contains the name of the local host in the `.rhosts` file in your home directory on the remote host.)
You cannot achieve autologin to a remote host because your login name on the local host is different from the login name on the remote host.	See the remote host's vendor documentation for the proper remote host procedures to authorize an autologin from a different login name. (If the remote host is running an operating system based on 4.3BSD, the solution may be as simple as putting an entry that contains the local host name and your account name in the `.rhosts` file that is in your home directory on the remote host, and then using the `-l` *login_name* option on the command line whenever you use the `rlogin` or `rsh` utility.) On a UNICOS MLS system, you may be prohibited from using autologin for a different account name than your local account name. Check with your system administrator and see subsection 8.2.4, page 104, for information about restrictions.
You cannot achieve autologin to a Cray Research host.	Place an entry containing the name of the host from which you are trying to perform the autologin in the `.rhosts` file in your home directory on the Cray Research host. Ensure that only the owner has write permission. Check with your system administrator about whether a UNICOS MLS system is running, and whether autologin is restricted. You may need to place an entry for your local host in the `/etc/hosts.equiv` file, and you may have to use the same account name on both local and remote systems.

Table 4. Authorization problems and solutions
(continued)

Problem	Solution
You want to authorize specific users on certain remote hosts to access your account on a Cray Research host.	Place entries in your `.rhosts` file in your home directory on the Cray Research host, listing every remote host and user you want to authorize for autologin access to your account. Such authorized users must then use the `-l` *login_name* option (*login_ name* represents your account name on the Cray Research host) on the command line whenever using the `rlogin` or `rsh` utility to access your account. Check with your system administrator about whether a UNICOS MLS system is running, and whether autologin is restricted. You may be prohibited from using different account names between the local and remote systems.
You want to forbid users on certain remote hosts access to your account on a Cray Research host.	Place entries in your `.rhosts` file in your home directory on the Cray Research host, listing every remote host from which you want to forbid access to your Cray Research account. Follow each entry with a –. (This designation does not deny access to a user whose account name on a remote host matches your account name on the Cray Research host, and/or whose remote host name the system administrator has placed in the `/etc/hosts.equiv` file on the Cray Research host; the entry in `/etc/hosts.equiv` overrides the entry in your `.rhosts` file.) If you are on a UNICOS MLS system, you might be able to prevent access to your account even when entries to the `/etc/hosts.equiv` file exist. See your system administrator for more information.

TCP/IP Network Security [8]

This section describes how the UNICOS multilevel security (MLS) feature is used to control access over TCP/IP connections. The *UNICOS Multilevel Security (MLS) Feature User's Guide*, publication SG–2111, describes UNICOS security features, including the UNICOS station call processor (USCP) and the Remote Queuing System (RQS). *The Network Queuing System (NQS) User's Guide,* publication SG–2105, discusses security for NQS systems.

This section describes TCP/IP network controls and TCP/IP user commands.

TCP/IP network controls

8.1

On a UNICOS MLS system, mandatory access controls (MACs) are applied for network daemon access to remote workstations or hosts and for UNICOS user processes. These controls check all user process and daemon network connections, based on the classification of the information and the classification of the remote node in the network access list (NAL).

The workstation access list (WAL) provides an additional layer of authorization control over network services. Using the WAL, an administrator can designate who can use the services of a remote host.

The network connection must pass MAC NAL checks and WAL checks. The following subsections explain these controls in more detail:

- TCP/IP NAL and WAL checks
- Network access list
- Login label
- NAL and UDB access procedure
- Workstation access list

TCP/IP NAL and WAL checks
8.1.1

TCP/IP NAL and WAL checks do the following:

- Verify that the connection's MAC label is within the remote node MAC label range, as specified in the NAL.
- Verify that the connection's security label also is bounded by the system's and network interface's security label ranges.
- Verify that the user or group is allowed access by the WAL to the requested service from the given network node.
- Record security violations in the security log.

Network access list
8.1.2

TCP/IP references the NAL for security control information about each network node.

Your security administrator (or `root` on a `PRIV_SU` system) creates a NAL entry for each remote node or network. The entry contains the Internet address, minimum security label, maximum security label, send and receive privileges, the type of security option (`basic`, `cipso`, or `none`), classification, and either the host input and output protection authorities (for `basic`) or the domain of interpretation (for `cipso`).

In Trusted UNICOS, only one label (minimum is the same as maximum) for any remote node can exist when the specified security option is `none`. The `localhost` entry is an exception to this rule.

If a remote node does not have an entry in the NAL, no communication is allowed.

Login label
8.1.3

For a connection that does not send labeled IP packets, the most restrictive security label in the intersection of the NAL, the network interface, and the user database (UDB) is given to you when you log in, as shown in the following example that concerns a user named Mary and her UDB entry.

After the connection label has been identified, the following three things are considered when setting the active MAC label and MAC label range for a session:

- Connection label range

- User's MAC attributes from the UDB, including the following:
  - Minimum level
  - Maximum level
  - Maximum compartments
  - Default level
  - Default compartments
- Login configuration value of `deflbl_as_minlbl`; this configuration value applies to all identification and authentication (I/A) mechanisms. If set, the user's default label is treated as the user's minimum label, allowing a site to define the minimum compartments. See your system administrator for more information.

The second and third considerations (MAC attributes from UDB and login configuration value of `deflbl_as_minlbl`) determine the user's allowed MAC label range. This range is used in conjunction with the connection MAC label range to determine the MAC label range of the session. This does not imply that the session range is made up of the end points of the two ranges, as demonstrated in the following example:

	User label range from the UDB	Connection label range from the NAL
Minimum	1, A, B	0, A, B, C
Maximum	5, A, B, C	2, A, B, C, D E

In the preceding example, the MAC label range for the session is as follows:

minimum 1, A, B, C
maximum 2, A, B, C

The active MAC label for the session is set to the user's default label if it is within the session MAC label range. The session's active MAC label is set to the session minimum label if the user's default is not within the session's label range.

The following list shows a UDB entry for a user named Mary on a UNICOS MLS system that is not sending labeled IP packets:

- Maximum security label is level 5, compartments A, B, and C
- Minimum security label is level 0, compartment null

- Default security label is level 0, compartment A

The network interface has a security label that consists of the following items:

- Maximum security label is level 5, compartments A, B, and C
- Minimum security label is level 0, compartment null

In addition, the NAL entry establishes the following security label for Mary's workstation:

- Maximum security label is level 3, compartments A and C
- Minimum security label is level 0, compartment null

Based on these entries, access is granted to Mary because her default security label of level 0, compartment A falls within the workstation's minimum security label of level 0, compartment null and maximum of level 3, compartments A, B, and C. The system automatically changes Mary's maximum security level from 5 to 3, because her maximum level (as defined in the UDB) exceeds the maximum level defined in the NAL for the workstation. Also, Mary can add only compartment C to her active set, because the NAL defines only compartments A and C as authorized for the workstation.

If the remote host sends labeled packets, the kernel checks the NAL and sets the specified security label to the socket. In this case, the session has the security label of the incoming packet, assuming that the label is within the security label range of the system, the UDB entry for the user, the NAL, and the network interface. You cannot change your security label while working in the session.

Note: In Trusted UNICOS, the label of a login session is set the same as the label of the connection for the duration of the session. This label is the intersection of the label of the originating host and the NAL entry on the destination host. This label cannot be changed. The user is denied access if the connection label is not within the NAL entry range, the network interface label for the connection, or the user's label range in the UDB. The option `NETW_STRICT_B1` controls the label of a login session, and must be set in a Trusted UNICOS system.

On a UNICOS MLS system, the label of a login session is always the same as the label of the connection if either the `basic` or `cipso` security option is used on the connection.

NAL and UDB access procedure
8.1.4

The NAL and UDB access procedures are as follows:

- For a connection not sending labeled IP packets: If access is granted, and the NAL specifies a security label that is more restrictive than specified in the UDB, your minimum/maximum security levels and compartments are made to match those specified by the NAL. The opposite is also true; if the UDB specifies a more restrictive security label than specified in the NAL, your minimum/maximum security levels and compartments are made to match those specified in the UDB.
- For a system sending labeled packets: Your security label is restricted to the socket connection's security label (that is, you cannot change your security label while connected to the socket). To establish the connection, the socket connection's security label must fall within the range defined for you in the NAL, network interface, and UDB.

Workstation access list
8.1.5

Your security administrator uses the WAL to define the users and groups that are granted access to remote services from a given remote node. The WAL also defines the services that are allowed at that remote node. The services that the WAL allows (by specification string) are `login`, `lpd`, `ftp`, `rsh`, `rexec`, `nfs` (deferred), `mail` (deferred), and `nqs`. The WAL `login` service governs interactive sessions through `rlogin` and `telnet`. The specifications `all` and `none` are also possible in the WAL.

If your workstation is not defined in the WAL, you are granted access to services. If your workstation is defined in the WAL, but your user name and group name is not listed, you are denied access to services.

TCP/IP user commands
8.2

The following subsections describe how the UNICOS MLS and Trusted UNICOS features affect TCP/IP commands:

- Remote nodes and user security ranges
- Generalized connection examples
- The `telnet` command
- The `rlogin` command

- The `remsh` command
- The `ftp` command
- The `rcp` command

For information about interface security labels on your Cray Research system, see your security administrator.

All of the examples in these subsections use the two remote nodes and users specified in subsection 8.2.1.

Note: In a Trusted UNICOS system, the label of a network connection cannot be changed. The following examples apply to UNICOS MLS systems. If an example has an IP security option specified, it applies also to Trusted UNICOS systems.

Remote nodes and user security ranges 8.2.1

In the examples in this subsection, the `snoopy` and `friend` remote nodes and user security ranges are used.

The NAL entry for `snoopy` on the UNICOS MLS system called `cray` is as follows:

- Minimum security level of 0
- Maximum security level of 6
- Minimum compartment of 0
- Maximum compartment of `train`
- Class C2
- IPSO is `none`

Note: This NAL definition is not allowed on a Trusted UNICOS system because hosts that do not have any IPSO security option must use only one label. The `localhost` entry is an exception.

The NAL entry for `friend` on the UNICOS MLS system called `cray` is as follows:

- Minimum security level of 0
- Maximum security level of 16
- Minimum compartment of 0
- Maximum compartments of all the compartments in the system

- Class B2
- IPSO is `cipso`

The network interface to the UNICOS MLS system called `cray` has the following limits:

- Minimum security level of 0
- Maximum security level of 16
- Minimum compartment of 0
- Maximum authorized compartments of `test` and `train`

Two users, Jack and Jill, have accounts on the UNICOS MLS system called `cray`.

Jack's UDB entry assigns him the following limits:

- Minimum security level of 0
- Maximum security level of 5
- Default security level of 0
- Authorized compartments are `train` and `test`
- Active compartments of null

Jill's UDB entry assigns her the following limits:

- Minimum security level of 0
- Maximum security level of 5
- Default security level of 0
- Authorized compartments are `train` and `test`
- Active compartments of null

Generalized connection examples
8.2.2

The following examples apply to all connection methods (`telnet`, `rsh`, `rlogin`, and so forth).

They use the previous definitions.

Example 1: Jack attempts to connect to `cray` from `snoopy` at active level 0, and active compartment `admin`.

Jack is denied access to `cray` because the interface device does not allow the compartment `admin`.

Example 2: Jack attempts to connect to `cray` from `snoopy` at active level 0, and active compartment `test`.

Jack is denied access to `cray` because the NAL states that `cray` does not allow a connection from `snoopy` with active compartment `test`.

Example 3: Jill attempts to connect to `cray` from `friend` at active level 6 and active compartment `test`.

Jill is denied access to `cray` because the UDB states that user Jill is not allowed a level higher than 5.

Example 4: Jill attempts to connect to `cray` from `friend` at active level 5 and active compartments `train` and `test`.

Jill is allowed access to `cray`.

The `telnet` *command* 8.2.3

When you use the `telnet` command to connect to a UNICOS MLS system from your remote node, you are assigned the most restrictive set of security levels and compartments from the combination of your UDB entry, the incoming host's NAL entry, and the security label for the network interface.

In example 1, Jack logs into `cray` from `snoopy`. Because his UDB entry is more restrictive for security level than the NAL entry for `snoopy`, and because the NAL is more restrictive for security compartments than his UDB entry, Jack's security environment reflects the intersection of these two entries.

Example 1:

```
snoopy$ telnet cray
Trying...
Connected to cray
Escape character is '^]'.
Cray UNICOS (cray) (ttyp051)
login: jack
Password:

Active label set to : level0,none


Last successful login was : Tue May 22 13:45:04 from snoopy

                Welcome to the UNICOS 9.0 system
cray$ spget

 permits equal 00
                none
 security level is 0
                level0
 maximum level is  5
                level5
 minimum level is  0
                level0
 authorized compartments are 040
                train
 active compartments are 00
                none
 integrity class is 0
                class0
 maximum class is   0
                class0
 active categories are 00
                none
 authorized categories are 00
                 none
cray$ setulvl 1
setulvl: New security label is
Level[1:level1] Compartments[none]
cray$
```

In example 2, Jack logs into `cray` from `friend`. Because `friend` uses the Common IP Security Option (CIPSO) method of labeling network packets, Jack's label range is constrained to a single label value, which is the label of the connection. The label of the connection is the same as the label of the session that Jack is using on `friend`. To operate at a different label, Jack needs to create a different session with the label he wants on `friend`, and initiate a connection from that session.

Example 2:

```
friend$ telnet cray
Trying...
Connected to cray
Escape character is '^]'.
Cray UNICOS (cray) (ttyp051)
login: jack
Password:

Active label set to : level0,none

Last successful login was : Tue May 22 13:45:04 from friend

                Welcome to the UNICOS 9.0 system
cray$ spget

 permits equal 00
                none
 security level is 0
                level0
 maximum level is  0
                level0
 minimum level is  0
                level0
 authorized compartments are 0
                none
 active compartments are 00
                none
 integrity class is 0
                class0
 maximum class is   0
                class0
 active categories are 00
                none
 authorized categories are 00
                 none
cray$ setulvl 1
sh:  cannot set security label: 1
cray$
```

The label at which you log in (the default) can be modified if the connection does not use Internet Protocol Security Options (IPSO) and if the `NETW_STRICT_B1` configuration parameter for the connection is not set. For example, if your default security level is 4 in the UDB but the NAL entry allows only a maximum and minimum security level of 0 for your remote node, you receive security level 0 as your default security level. To be allowed access, the UDB entry must allow the label. One exception exists. If the login configuration option `deflbl_as_minlbl` is not set, then users can log in at a lower label than the default label, if the default label is higher than 0.

An outgoing `telnet` request from a UNICOS system requires that your active security level and compartments must be within the range of levels and compartments assigned to the NAL entry for the remote node to which you are trying to connect, and must be within the range of the network interface that is used.

In example 3, Jill tries to log into `snoopy`. Her current security level is 0 and her active compartment is `test`. She is denied the connection to `snoopy`, because `snoopy` does not have the `test` compartment listed in the NAL.

In example 4, Jill tries to log into `friend`. She can connect to `friend`, because the NAL entry for `friend` supports compartment `test`.

Example 3:

```
cray$ id
uid=1234(jill) gid=28(trng)
cray$ setucmp test
setucmp: New security label is
Level[0:level0] Compartments[test]
cray$ telnet snoopy
Trying 128.162.121.3...
telnet: Unable to connect to remote host: Security level outside host range
cray$
```

Example 4:

```
cray$ id
uid=1234(jill) gid=28(trng)
cray$ telnet friend
Trying 234.6.12.4...
Connected to friend.
Escape character is '^]'.

4.2 BSD UNIX (friend)

login:
```

The `rlogin` *command* 8.2.4

Incoming and outgoing `rlogin` requests abide by the same rules as `telnet` requests.

There are two `rlogin` behaviors, with the configuration parameter `NETW_RCMD_COMPAT` in the `SECURE_NET_OPTIONS` configuration entry serving as a toggle between them. When you set `NETW_RCMD_COMPAT`, `.rhosts` and `/etc/host.equiv` provide the normal BSD functionality. If it is not set, `.rhosts` and `/etc/hosts.equiv` work in restricted fashion. For the `r` commands to work without a password, the following must be true:

- The host is listed in the `/etc/hosts.equiv` and `.rhosts` files.
- The remote user ID is the same as the local user ID (the `-l` option does not work).
- The user is not `root`.

In example 1 (following), Jack has created a `.rhosts` file on `cray` that allows an automatic login for Jack from `friend`, and the system administrator has added `friend` to the `hosts.equiv` file. The connection from `friend` to `cray` uses CIPSO, so Jack's range is restricted to a single label (level 0, no compartments), which is what Jack had on `friend` before he started the session on `cray`. The actual security label values on `friend` are translated on the `cray` into UNICOS MLS security label values by taking values from the Domain Of Interpretation (DOI) translate table that are appropriate for `friend`.

> **Note:** The NAL parameter of class must be C2 or higher to enable automatic login, `r` commands, remote printing by using `lpd`, and NFS clients.

In example 2, Jack has created a `.rhosts` file on `snoopy` that allows an automatic login for `jack` from `cray`. However, Jack's active security compartment (`test`) is not supported in the NAL entry for `snoopy` on the `cray` host.

Example 1:

```
friend$ rlogin cray
Last successful login was : Tue May 22 14:12:38 from snoopy

               Welcome to the UNICOS 9.0 system

cray$ spget

 permits equal 00
                 none
 security level is 0
                 level0
 maximum level is  0
                 level0
 minimum level is  0
                 level0
 authorized compartments are 000
                 none
 active compartments are 00
                 none
 integrity class is 0
                 class0
 maximum class is   0
                 class0
 active categories are 00
                 none
 authorized categories are 00
                 none
```

Example: 2

```
cray$ id
uid=2345 (jack) gid=28(trng)
cray$ setucmp test
setucmp: New security label is
Level[0:level0] Compartments[test]
cray$ rlogin snoopy
snoopy.cray.com: Security level outside host range
cray$
```

The `remsh` *command* 8.2.5

Outgoing `remsh` requests execute the same as outgoing `telnet` or `rlogin` requests. Your security label must be within the boundary of the host's security range, as defined in the NAL and the interface range.

In example 1, Jill tries to execute the `remsh` command to `snoopy`. She is denied access because the NAL entry for `snoopy` does not have the `test` compartment.

In example 2, Jill tries to execute the `remsh` command to `friend`. She is granted access because the NAL entry for `friend` supports the `test` compartment and her `.rhosts` file on `friend` grants her access.

Example 1:

```
cray$ id
uid=1234(jill) gid=28(trng)
cray$ setucmp test
setucmp: New security label is
Level[0:level0] Compartments[test]
cray$ remsh snoopy ls
snoopy: Security level outside host range
$cray
```

Example 2:

```
cray$ id
uid=1234(jill) gid=28(trng)
cray$ remsh friend ls
calendar       letter  pers    read    testfile
file           mbox    ows     roster  work
cray$
```

The `ftp` *command*
8.2.6

When transferring classified files between a UNICOS MLS system and a remote node, you should use the `ftp` command. If you use the `ftp` command from the remote node and are not running with IPSO, you can transfer files only at the default security label assigned to you at login. No mechanism exists for changing your security label within `ftp`. However, the active label when starting an `ftp` session is the label for the `ftp` session.

When you have logged into your UNICOS MLS system, set your active security label to that of the file you want to transfer, then execute the `ftp` command. If the remote node supports your security label, you can transfer the file. Files that are transferred to the UNICOS MLS system are labeled with the active security label of the `ftp` session. Also, you must be in a directory that can accommodate a file created at that label.

The following three examples illustrate use of the `ftp` command.

In example 1, Jill wants to transfer the file called `testdata` from `cray` to `snoopy`. `testdata` has a security level of 1 and the `test` compartment. Jill adjusts her security level and compartment settings to match the file's security level and compartments. She then executes the `ftp` command, but she is denied access because the NAL entry for `snoopy` does not support the `test` compartment.

Example 1:

```
cray$ spget -f testdata
 Security Values for: testdata
          level:   1
                   level1
 compartments:   010
                  test
          class:   0
                 class0
    categories:   0
                 none
          flags:   0
                 none
cray$ setulvl 1
setulvl: New security label is
Level[1:level1] Compartments[none]
cray$ setucmp test
setucmp: New security label is
Level[1:level1] Compartments[test]
cray$ ftp snoopy
ftp: connect: Security level outside host range
ftp> quit
cray$
```

In example 2, Jill transfers the file to `friend`. The transfer is successful because the NAL entry for `friend` supports her security label.

Example 2:

```
cray$ ftp friend
Connected to friend
220 friend FTP server (Version 4.15 Sat Nov 7 15:24:41 PST 1987)
ready.
Name (friend:jill):
331 Password required for jill.
Password: 230 User jill logged in.
ftp> put testdata
200 PORT command okay.
150 Opening data connection for testdata (234.6.12.4,1035).
226 Transfer complete.
16 bytes sent in 0.022 seconds (0.71 Kbytes/s)
ftp> quit
221 Goodbye.
cray$
```

In example 3, Jill transfers a file to `cray` from `friend`. First, she changes to a directory where she can create a file with a security level of 1 and `test` compartment. The transfer is successful because the NAL entry for `friend` supports her security label and the file can be created in her current directory.

Example 3:

```
cray$ cd lev_1_comp_test-dir
cray$ spget -f .
Security Values for: .
         level:   1
                level1
  compartments:   010
                test
         class:   0
                class0
    categories:   0
                none
         flags:   0
                none
cray$ ftp friend
Connected to friend
220 friend FTP server (Version 4.15 Sat Nov 7 15:24:41 PST 1987)
ready.
Name (friend:jill): jill
331 Password required for jill.
Password:
230 User jill logged in.
ftp> get friend.file
200 PORT command okay.
150 ASCII data connection for friend.file(128.162.82.15,1187
226 ASCII Transfer complete.
56821  bytes received in 0.58 seconds (96 Kbytes/s)
ftp> quit
211 Goodbye.
cray$ spget -f friend.file
Security Values for: friend.file
         level:   1
                level1
  compartments:   010
                test
         class:   0
                class0
    categories:   0
                none
         flags:   0
                none
cray$
```

The `rcp` *command*
8.2.7

On a UNICOS MLS system, all files sent or received when executing the `rcp` command are accessed at your active security label. Outgoing `rcp` requests can copy files to or from the remote node. The NAL entry for the remote node must accommodate this security label. Any file that is created in this manner is labeled with your active security label.

Effect of security labels on electronic mail
8.3

The following subsections describe the effect of security labels on sending and receiving electronic mail.

To become familiar with security concepts in this subsection, see the *UNICOS Multilevel Security (MLS) Feature User's Guide*, publication SG–2111. For information about how to use the `mail` and `mailx` utilities, see subsection 5.1, page 65.

Mail is sent both locally and across the network. From a user perspective, mail sent across the network is the same as mail sent locally. In the examples in this subsection, no differentiation is made between the two. Subsection 8.3.2.3 on page 116 describes what happens to security labels on mail delivered over a network.

To illustrate how electronic mail works with systems having security labels, this subsection begins with the simplest example of using `mail`(1) and `mailx`(1). Subsequent examples become more complex, and describe how labeling and delivery of messages is affected when the security label on the mail sent to you is different from your active security label.

Sending and receiving labels are the same
8.3.1

In the simplest example, all users on the system and all network connections to the system have the same security label. Therefore, all mail is sent at that label. On this system, you can always read mail; you can always forward mail; and you can always receive forwarded mail. Editing, saving, and deleting mail follows the mandatory access control (MAC) rules for security labels and discretionary access control (DAC) considerations.

Sent mail label differs from the label of the receiver
8.3.2

The next example describes sending and receiving mail when all security labels are not the same. In this example, all users on the system can have one of two security labels, label A or label B. Label A has a level of 0 and a null compartment set. Label B has a level of 1 and a null compartment set. (All examples have the null compartment set to simplify understanding. The same rules apply to labels having compartment sets that are not null.)

Receiving mail at both label A and label B
8.3.2.1

When someone logs in to a UNICOS MLS system at label A and sends mail, the mail is delivered to your mailbox at label A. Likewise, when a sender logs in to a UNICOS MLS system at label B and sends mail, the mail is delivered at label B. Suppose that mail messages at each label were sent to you.

In this case, when you log in at label A, the following message is displayed:

```
You have unreadable mail at label 1/0.
```

This means that mail with a level of 1 and a null compartment set, that is, mail at label B, was delivered to you.

If you execute `mail`(1) or `mailx`(1), a similar message appears, and you can read the mail sent at label A. If your saved mail directory is at label A, you can save, edit, delete, and forward the mail at label A as usual.

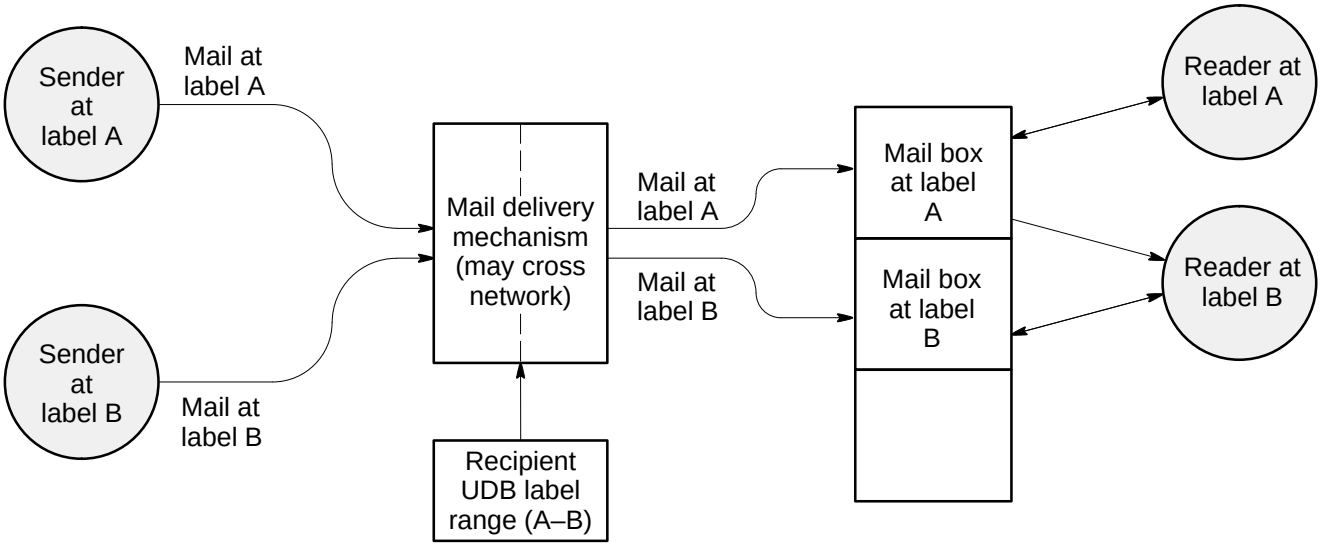
To read the mail sent at label B, you must change your security label to label B. Note that the `unreadable mail` message will no longer be displayed, since your label now dominates all mail labels.

Because you are at label B, you cannot delete the mail at label A. If you are using `mail`(1) and try to delete the mail, the mail simply stays. If you are using `mailx`(1), a message is displayed stating that the mail you are trying to delete is read-only. You must log out and log back in at label A in order to delete the mail.

If you are logged on at label B, but your saved mail directory is at label A, you cannot save either message in that directory. In order to write to that directory, you must be at label A. You can save either message in a file or a directory having label B. UNICOS MLS does allow mail to be saved in a single directory at more than one label if the directory to which you save is a multilevel directory (MLD).

If you forward a message or reply to one, the message is sent at your currently active label, which in this example is label B. Even though you receive a forwarded mail message at label A, the message is delivered at label B. To keep the message at label A, you must log out and then log in at label A. However, the opposite is not possible. If you are at label A, and receive a message with label B, you must log out and log back in at label B in order to read it.

See Figure 3 for an illustration of this example.



		Read	Edit/ delete	Reply/ forward	Save	Unreadable message
Reader at label A	Stored mail A	Y	Y	Label A	Label A	N
	Stored mail B	N	N	N/A	N/A	Y

		Read	Edit/ delete	Reply/ forward	Save	Unreadable message
Reader at label B	Stored mail A	Y	N	Label B	Label B	N
	Stored mail B	Y	Y	Label B	Label B	N

Figure 3. Functions of mail at different security labels

Mail label and your label are at several different labels
8.3.2.2

The next example involves four labels: labels A, B, C, and D. Label A is at level 0 with a null compartment set; label B is at level 1 with a null compartment set; label C is at level 3 with a null compartment set; and label D is at level 4 with a null compartment set.

The sender can log in at any of the four labels. You can log in only at labels B and C.

The sender logs in at labels A, B, and D, and sends mail at each label to you. Each mail message is treated differently, depending on its label and your active security label.

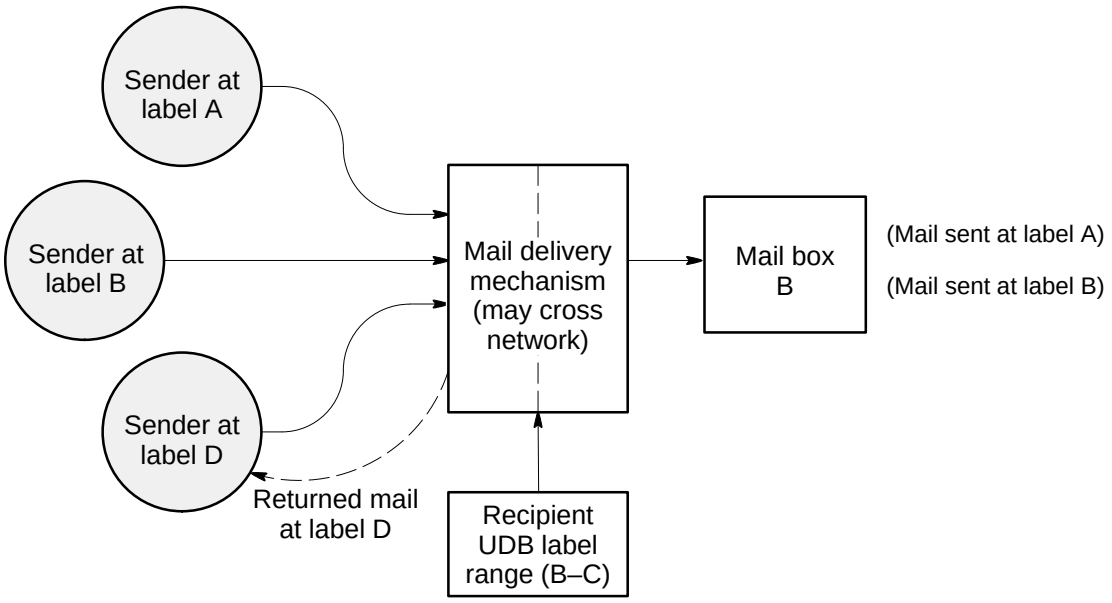
The mail at label A is outside your allowed range of labels. However, the message is dominated by label B, which is the lowest label at which you can log in. The `sendmail`(1) command, which manages mail delivery, automatically changes the label from A to B, your lowest level, which is more restrictive than the mail's label A. The mail is delivered at label B. Thus, you are assured of receiving the mail.

This label change to label B also prevents user-specified mail programs (specified in your `.forward` file or the system `alias` file) from executing on your behalf outside your label range.

The mail at label B is within your range, and is delivered at label B.

The mail at label D is also outside your label range, but in this case, your label range does not dominate the mail label. You will never be able to read or delete the message as a result. The mail is returned to the sender with the error message `User unknown`. Returning the mail also ensures that the program mailer will run only within your label range.

Figure 4 illustrates delivery of mail at different labels to you when you are logged in at different labels.



	Mail delivered	Label at which mail is delivered
Mail sent at label A	Y	B
Mail sent at label B	Y	B
Mail sent at label D	N	N/A

Figure 4. Delivery of mail at different labels to recipients at different labels

Delivering mail across the network
8.3.2.3

When delivering across the network, the security label of the incoming mail is checked against the security label ranges of the network interface and the network node (as specified in the NAL).

A connection is established across the network at the label of the mail being sent. This connection fails if the label is outside the label range of the NAL or the interface. In this case, the response to the sender indicates that the remote host could not be reached.

Figure 5 illustrates how different security labels affect mail delivery across a network.

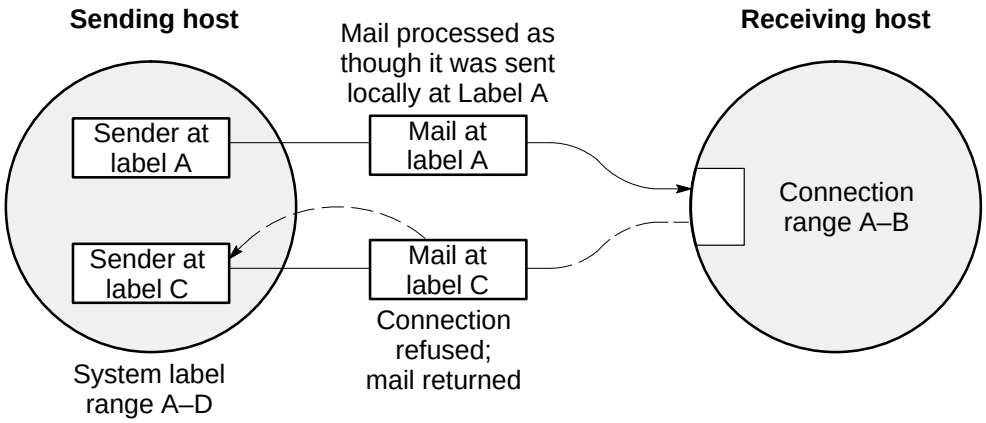


Figure 5. Mail delivery to a system with a different label range at the connection

Error Messages [A]

This appendix contains the following subsections:

- System error messages
- `ftp` and `tftp` error messages
- `telnet` error messages

Note: Authorization problems associated with the autologin feature are discussed in subsection 7.3, page 88.

The messages are listed in alphabetical order, along with a description of their probable causes.

System error messages
A.1

The system error messages described in this subsection are displayed when an improper operation occurs at the kernel level. The lower-level protocols return these messages.

`Address already in use`

Addresses must be unique to each host and can be used only once.

`Address family not supported by protocol family`

An address incompatible with the requested protocol was used.

`Bad protocol option`

An option that is not valid was specified in a `getsockopt` or `setsockopt` system call (see `getsockopt`(2)).

`Can't assign requested address`

An attempt was made to bind a socket to an illegal address or to an address already in use.

`Can't send after socket shutdown`

A request to send data was disallowed because the socket was already shut down by a previous `shutdown`(2) system call.

`Connection refused`

A connection could not be made because the remote host actively refused it. This usually occurs because of an attempt to connect to a service that is inactive on the remote host.

`Connection reset by peer`

A network peer connection forcibly closed. This usually occurs when the peer executes a `shutdown`(2) system call.

`Connection timed out`

A connect request failed because the connected user did not respond properly after a specified period of time. This might indicate that the remote host is down.

`Control block: symbol not in namelist`

A problem exists with the symbol table in the kernel.

`Destination address required`

A required address was omitted from an operation on a socket.

`Macro not found`

You have not defined the macro or have not typed the macro name correctly.

`Message too long`

A message sent on a socket was larger than the internal message buffer.

`Network dropped connection on reset`

The host to which you were connected crashed and rebooted on reset, but it did not renew the connection.

`Network is down`

A socket operation encountered an inactive network.

`Network is unreachable`

A socket operation was tried on an unreachable network. Check addresses, routing, and labeling information.

`No buffer space available`

An operation on a socket or pipe was not performed because the system had insufficient buffer space.

`Operation already in progress`

An operation was tried on a nonblocking object that already had an operation in progress.

`Operation now in progress`

An operation requiring a long time to complete (such as a connect) was tried.

`Operation would block`

A blocking operation was tried on an object in nonblocking mode.

`Socket is already connected`

A connect request was made on a socket that is already connected, or a `sendto` or `sendmsg` (see `send`(2)) request made on a connected socket specified a destination other than the connected user.

`Socket is not connected`

A request to send data was disallowed because the socket was already shut down with a previous `shutdown`(2) system call.

`Socket operation on non-socket`

A socket operation was tried on a nonsocket device.

`Socket type not supported`

Support for the specified socket type is not configured in the system or is not implemented.

`Software caused connection abort`

The local host aborted a connection.

`ftp` and `tftp` error messages

A.2

The `ftp`(1B) and `tftp`(1B) error messages are as follows:

`?Ambiguous command`

You tried to invoke an `ftp` command with an illegal abbreviation. Specify enough characters of the command name to distinguish it from all other `ftp` commands.

`?Ambiguous help command` *command*

You tried to invoke a `help` command with an illegal name. Specify enough of the characters in the name to make it unique.

`?Invalid command`

You tried to invoke a *command* that is unknown to `ftp`. To display a complete list of valid commands, enter a question mark (?).

`?Invalid help command` *command*

You tried to invoke a `help` command by using a command name that is not valid. For a complete list of valid commands, enter a question mark (?).

`Already connected to` *hostname*`, use disconnect first.`

You are currently connected to remote host *hostname*, although you may not be logged in. Either log in, or disconnect and retry.

`Bad port number`

A 0 or a negative port number was specified for the *port* operand on the `open` command line.

`Can't find list of remote files, oops`

When you invoke a multiple command (for example, `mput`), a temporary file is created in the `/tmp` directory. The system cannot find the file, or it was accidentally removed. Retry the procedure.

`Filename not a plain file`

The name of a directory or some special device was specified for a transfer operation when only a file name was appropriate.

`Ftp/tcp: unknown service`

The `/etc/services` file does not contain an entry for the specified `ftp` service. You should create an entry for this service in the file.

`Login failed`

You specified a user name or password that is not valid. Retry.

`Lost connection`

The server on the remote host closed the connection. This could occur because of a system crash, automatic logout, or other similar situations.

`No port available for data connection`

This indicates possible software problems.

`No target machine specified`

This message is associated with `tftp`. It appears when a file transfer is invoked without the specification of a remote host. Use the `open` command before invoking a file transfer command.

`Not connected`

You tried to use an `ftp` command that requires a connection to an `ftp` server before you established the connection. Make the connection before invoking the file transfer command.

`Unknown host`

The host name specified is not in the `/etc/hosts` file. Verify the host name and retry.

`Unknown mode`

You tried to set a mode that is not supported. The supported modes are `binary`, `ascii`, and `tenex`.

`User already logged in`

A user who is already logged in to remote host tried to log in again.

`We only support file structure, sorry`

Currently, `ftp` supports only file structure.

`We only support non-print`

Currently, `ftp` supports only nonprint format.

`telnet` error messages

A.3

The `telnet` error messages are as follows:

`?Already connected to` *hostname*

You tried to establish a connection with a remote host to which you are already connected.

`?Ambiguous command`

You tried to invoke a `telnet` command with an illegal abbreviation. Specify sufficient command information to make the abbreviation unique with respect to other commands.

`?Ambiguous help command` *command*

You tried to invoke a `help` command with an illegal abbreviation. Specify enough of the characters in the command name to make it unique.

`Bad port number`

The port specified by the *port* argument to the `open` command was 0, or, if a symbolic port name was used, it was not found in the `/etc/services` file.

`connect: security level outside host range`

The network access list (NAL) security label range for the remote host and/or interface security label range do not include your active label.

`Connection closed by foreign host`

A peer forcibly closed a connection. This usually occurs because of the peer executing a `shutdown`(2) system call, although it also can indicate that the remote host has crashed.

`?Invalid command`

You tried to invoke a command that is unknown to `telnet`. To display a complete list of valid commands, enter a question mark (?).

`?Invalid help command` *command*

You tried to invoke the `help` command with an invalid command name. To display a complete list of valid commands, enter a question mark (?).

`?Need to be connected first`

You tried to execute a `telnet` command that requires an open connection to a remote system, and you do not currently have an open connection. Before retrying this command, use the `open` command to establish a connection.

`Telnet: tcp/telnet:  unknown service`

The `/etc/services` file does not contain an entry for the `telnet` service. You should create an entry for it in the file.

`Unknown host`

The host name that you specified is not recognizable for one of the following reasons:

- Your machine is not using the name server and the host name that you specified is not listed in the `/etc/hosts` file.
- You specified the host name incorrectly.
- Your machine is using the name server and the host name that you specified is not known to the domain name service; you might have to use the fully qualified domain name of the remote host, or you might be using an old alias name that is no longer supported.

`Unable to connect to remote host:  security level outside host range`

The NAL security label range for the remote host and/or interface security label range do not include your active label.

Glossary

<INTERRUPT> The interrupt key, usually specified by CONTROL-c.

account name See *login name*.

address A number that uniquely identifies each host on a network. The Internet address families are the only address families that UNICOS currently recognizes.

ASCII An acronym for American Standard Code for Information Interchange, an industry-wide standard for coding characters.

authorization files Files that contain host and user information that is verified by the system before user privileges are granted on a remote system.

autologin A feature that lets a user log in to a file that belongs to another user. The user logging in is not required to enter a login name or password because the system verifies the information automatically.

BSD Berkeley Software Distribution; a version of the UNIX operating system developed at the University of California at Berkeley.

computer network A system of computers and terminals linked together for the purpose of moving data from one machine to another.

connection The establishment of a temporary link between two communication endpoints for the purpose of transmitting data between them.

DARPA Defense Advanced Research Projects Administration (DARPA). An agency of the U.S. Department of Defense, originally defined TCP/IP.

DAP Data access protocol, a Digital Equipment Corporation proprietary protocol used for file transfer.

DoD The U.S. Department of Defense.

domain name The logical location of a system in a tree-structured organization of available systems.

escape character A character reserved for `telnet`(1B) and `rlogin`(1B) utilities and used to invoke command mode. When you enter the escape character with a command, the command is interpreted by the `telnet`, `rlogin`, or `vt` utilities and not by the shell of the remote host. The default is: `telnet`, CONTROL-] (control right-bracket); `rlogin`, ~(tilde); or `vt`, CONTROL-_ (control-underscore).

/etc/hosts A file on the Cray Research system that contains the official host name, Internet address, and valid aliases for each host on the network.

/etc/hosts.equiv A file set up on each host by the system or network administrator; this file contains the names of other network hosts granted access to the host. The system uses this file to authorize autologin from a remote host.

FEI-3 Front-end interface. A line of interface products for interconnecting a Cray low-speed channel into a system with a VME backplane.

file transfer service A UNICOS utility that performs file transfer operations for a particular domain.

FTP Abbreviation for file transfer protocol. The `ftp`(1) program is a TCP/IP utility that copies files between a local and remote host.

gateway A computer or router that has connections to more than one network.

globbing Globbing enables the expansion of shell metacharacters (*, ?, [...]) within file and path names.

host An individual computer on a network.

host Domain name server host name look-up command.

`host.equiv` See `/etc/hosts.equiv`.

`hostid` A command that prints the identifier of the current host in hexadecimal. The administrator also uses it to set the ID.

`hostname` A command that prints the name of the current host system. The system administrator also uses the `hostname` command to set the name of the host system.

`hosts` See `/etc/hosts`.

HSX High-speed external communications channel. A Cray Research proprietary, 64-bit, parallel data channel that operates at speeds of up to 100 Mbyte/s.

HYPERchannel A networking medium connected by Network Systems Corporation (NSC) HYPERchannel hardware.

immediate mode Mode in which file transfer is executed immediately after a file transfer command is typed.

internet Abbreviation for an internetwork, an interconnected set of local area networks.

Internet dot notation A 32-bit logical address usually expressed as four 8-bit integers separated by periods (dots).

LAN Local area network. A LAN is a computer network that covers a relatively small (or local) area.

local host The computer from which you are originating a networking command.

login name The brief, unique name chosen by a user for identification at login time. The user's account on a computer system is usually set up under the user's login name, so the login name also is referred to as the *account name*.

`mail` A utility for sending mail on the network by using TCP/IP's `sendmail`(8) daemon.

`mailx` Same as `mail`.

network Two or more computers that are connected by wires, satellite, or other media. The connection between the computers lets individual computers share and exchange information.

node Device that can connect to a network.

packet A single unit of information that is transmitted over the network. The length of a packet varies. A single message can be carried by one packet or by a series of packets.

path name The name that specifies the directory location of a file.

protocol A standardized set of rules for transmission of data that allows communication between different types of hosts on a network.

`rcp` A file transfer utility used to copy files between hosts over a TCP/IP network.

remote host Any host computer system, other than the local host, on a network.

`remsh` A synonym for `rsh`.

`.rhosts` A file in a user's home directory that controls autologin to the user's account.

`rlogin` A remote login utility that lets you connect to a remote host on a TCP/IP network.

router Network hardware that has connections to more than one network.

`rsh` A command that provides automatic login to a remote host over a TCP/IP network and execution of a single command, then automatically returns you to the current environment of your local host.

standard output The place to which program output is directed. The default standard output device is the terminal screen.

TCP/IP Transmission Control Protocol/Internet Protocol; a set of computer networking protocols that allow two or more hosts to communicate.

`telnet` A TCP/IP utility that provides remote login between hosts on a network. This command invokes no operating system requirement.

TFTP Abbreviation for trivial file transfer protocol. The `tftp`(1B) program is a TCP/IP utility that copies files between a local and remote host.

virtual connection A connection in which you can use the resources of the remote host as though you were directly connected to it.

WAN A wide-area network. A WAN is a computer network located over a large geographic area.