Data General

Customer Documentation

Using TCP/IP on the DG/UX™ System

AViiON™
PRODUCT LINE

Using TCP/IP on the DG/UX™ System

093-701023-02

For the latest enhancements, cautions, documentation changes, and other information on this product, please see the Release Notice (085-series) supplied with the software.

Chapter 4 documents the Background File Transfer Program (BFTP), a program developed and implemented by the University of Southern California, Information Sciences Institute.

Using TCP/IP on the DG/UX™ System

093-701023-02

093-701058-02 (Japan Only)

Revision History:	Effective with:
Original Release — August 1986	DG/UX™ TCP/IP Rev. 1.0
First Revision — November 1989	DG/UX™ TCP/IP Rev. 4.20
Second Revision —May 1990	DG/UX™ TCP/IP Rev. 4.30

Preface

This manual introduces Data General's TCP/IP for AViiON™ Systems package. It describes how to use the package on AViiON™ 5000 and 6000 series systems and AViiON 300 and 400 series stations that run the DG/UX operating system. Technical changes from the previous release of this document are marked by vertical bars in the right column.

This manual assumes that you are familiar with the Bourne or C shell and the DG/UX file structure and process hierarchy. If you are unfamiliar with any of these subjects, please see *Using the DG/UX™ System* (093-701035) before using this manual.

Who Should Read This Manual?

This manual is for readers who want to use the following user-level commands provided with DG/UX TCP/IP.

telnet This command implements the TELNET protocol, which allows a user on one host to interact with a remote host as if the terminal is directly connected to the remote host.

ftp This command implements the File Transfer Protocol (FTP), which allows you to transfer files from one host to another.

bftp This command uses the FTP to transfer files from one host to another in the background. You do not have to be directly involved when the file transfer takes place.

tftp This command implements the Trivial File Transfer Protocol (TFTP), which allows file transfer with minimal capability and overhead.

Remote commands
These commands allow you to get information from, log in to, and execute commands on a remote host.

How This Manual Is Organized

This manual contains six chapters an appendix, a glossary, and manual pages.

Chapter 1	Introduces networking in general and the DG/UX TCP/IP product.
Chapter 2	Describes how to use the **telnet** program, which allows you to log in to a remote host from a local terminal.
Chapter 3	Describes how to use the file transfer program (**ftp**), which allows you to manipulate files across a TCP/IP network.
Chapter 4	Describes how to use the Background File Transfer Program (BFTP), which allows you to transfer files back and forth across a TCP/IP network in the background. This is a new chapter; there are no revision bars.
Chapter 5	Describes how to use the Trivial File Transfer Program (**tftp**), which allows you to transfer files back and forth across a TCP/IP network.
Chapter 6	Lists and describes the Remote commands, which allow you to log in to, execute a command on, copy files to, or receive information from any system on the network that is also running R commands.
Appendix A	Lists and describes error messages generated by the FTP server program.
Glossary	Provides a glossary of technical terms used in this manual.
Manual Pages	Provides manual pages to those who use the user-level commands and programs provided with the TCP/IP package.
Documentation Set	Lists and describes all documents currently available for AViiON computer systems.

Related Documents

For a list of all documents currently available for AViiON computer systems, see the "Documentation Set" that appears after the index. For information beyond the scope of this manual, consult the following documents.

Installing and Managing the DG/UX™ System (093-701052). Shows how to install and manage the DG/UX operating system on AViiON hosts that will run as stand-alone, server, or client systems. Aimed at system administrators who are familiar with the UNIX operating system.

User's Reference for the DG/UX™ System (093-701054). Contains an alphabetical listing of manual pages for commands relating to general system operation.

Using the DG/UX™ System (093-701035). Describes the DG/UX system and its major features, including **mailx** the C shell, the Bourne shell, and the filing system.

Using the DG/UX™ Editors (093-701036). Describes the text editors **vi** and **ed**, the batch editor **sed**, and the command line editor **editread**.

System Manager's Reference for the DG/UX™ System (093-701050). Contains an alphabetical listing of manual pages for commands relating to general system operation.

Setting Up and Managing TCP/IP on the DG/UX™ System (093-701051). Explains how to prepare for the installation of Data General's TCP/IP (DG/UX) package on AViiON computer systems. Contains information on tailoring the software for your site, managing the system, and troubleshooting system problems.

Programming with TCP/IP on the DG/UX™ System (093-701024). Describes how to use the socket system calls to access TCP, UDP, and IP protocol software.

Managing NFS® and Its Facilities on the DG/UX™ System (093-701049). Shows how to install, manage, and use the DG/UX ONC™/NFS® product. Contains information on the Network File System (NFS), the Yellow Pages (YP), Remote Procedure Calls (RPÇ), and External Data Representation (XDR). (NFS is a U.S. registered trademark of Sun Microsystems, Inc. ONC is a trademark of Sun Microsystems, Inc.)

Programmer's Reference for the DG/UX™ System (093-701055 and 093-701056). Alphabetical listing of manual pages for programming commands on the DG/UX system. This two-volume set includes information on system calls, file formats, subroutines, and libraries.

Using the DG/UX™ Software Development Tools (093-701078). Discusses programming support tools (**awk**, **nawk**, **lex**, **yacc**, **ld**, **lint**, and **as**), archiving, the C language, and SCCS.

Reader, Please Note

Data General manuals use certain symbols and styles of type to indicate different meanings. The symbol and typeface conventions used in this manual are defined in the following list. You should familiarize yourself with these conventions before reading the manual.

This manual also presumes the following meanings for the terms "command line," "format line," and "syntax line." A command line is an example of a command string that you should type verbatim; it is preceded by a system prompt and followed by a delimiter such as the curved arrow symbol for the New Line key. A format line shows how to structure a command; it shows the variables that must be supplied and the available options. A syntax line is a fragment of program code that shows how to use a particular routine; some syntax lines contain variables.

Convention	**Meaning**
boldface	In command lines and format lines, boldface indicates text (including punctuation) that you type verbatim from your keyboard. All DG/UX commands, pathnames, and names of files, directories, and manual pages also use this typeface.
`constant width/monospace`	Represents a system response on your screen. Syntax lines and examples of code also use this font.
italic	In format lines: Represents variables for which you supply values; for example, the names of your directories and files, your username and password, and possible arguments to commands. In text: Indicates a term that is defined in the manual's glossary.
[*optional*]	In format lines: These brackets surround an optional argument. Don't type the brackets; they only set off what is optional. The brackets are in regular type and should not be confused with the boldface brackets shown below.

Convention	Meaning
[ ]	In format lines: Indicates literal brackets that you should type. These brackets are in boldface type and should not be confused with the regular type brackets shown above.
...	In format lines and syntax lines: Means you can repeat the preceding argument as many times as desired.
$ and %	In command lines and other examples: Represent the system command prompt symbols used for the Bourne and C shells, respectively. Note that your system might use different symbols for the command prompts.
↵	In command lines and other examples: Represents the New Line key, which is the name of the key used to generate a new line. (Note that on some keyboards this key might be called Enter or Return instead of New Line.) Throughout this manual, a space precedes the New Line symbol; this space is used only to improve readability — you can ignore it.
< >	In command lines and other examples: Angle brackets distinguish a command sequence or a keystroke (such as <**Ctrl-D**>, <**Esc**>, and <**3dw**>) from surrounding text. Note that these angle brackets are in regular type and that you do not type them.

Contacting Data General

Data General wants to assist you in any way it can to help you use its products. Please feel free to contact the company as outlined below.

Manuals

If you require additional manuals, please use the enclosed TIPS order form (United States only) or contact your local Data General sales representative.

If you have comments on this manual, please use the prepaid Comment Form that appears at the back. We want to know what you like and dislike about this manual.

Telephone Assistance

If you are unable to solve a problem using any manual you received with your system, and you are within the United States or Canada, contact the Data General Service Center by calling 1-800-DG-HELPS for toll-free telephone support. The center will put you in touch with a member of Data General's telephone assistance staff who can answer your questions.

Free telephone assistance is available with your warranty and with most Data General service options. Lines are open from 8:30 a.m. to 8:30 p.m., Eastern Time, Monday through Friday.

For telephone assistance outside the United States or Canada, ask your Data General sales representative for the appropriate telephone number.

Joining Our Users Group

Please consider joining the largest independent organization of Data General users, the North American Data General Users Group (NADGUG). In addition to making valuable contacts, members receive FOCUS monthly magazine, a conference discount, access to the Software Library and Electronic Bulletin Board, an annual Member Directory, Regional and Special Interest Groups, and much more. For more information about membership in the North American Data General Users Group, call 1-800-877-4787 or 1-512-345-5316.

End of Preface

Contents

Documentation Set

Tables

Figures

Chapter 1
Understanding Networking and DG/UX TCP/IP

This chapter provides an overview of networking and of the DG/UX TCP/IP package. It tells you what you need to know to use the DG/UX TCP/IP family of protocols.

Reviewing Basic Terms

Before covering the components of the DG/UX TCP/IP package, it may be helpful to review some basic networking terms. If you do not need such a review, you can skip ahead to the next section.

First of all, TCP/IP stands for Transmission Control Protocol and Internet Protocol. These two protocols are described at length later in this chapter.

A *network* enables two or more computer systems to communicate. It allows the computers to share information and have easy access to other machines. The rules that govern network communications are called *protocols*.

A network consists of three major physical components:

The local host

Links

One or more remote hosts

The *local host* is the computer to which your terminal is connected. It can send to and receive information from the remote host.

The *link* is a medium connecting a local host with one or more remote hosts. A link can be a coaxial cable, a microwave signal, a twisted pair phone line, or any physical communications medium that transmits a signal.

Remote hosts are the other computers on the network that you wish to access. They can also send to and receive information or requests from your local host.

A local area network (LAN) is a network within a small area, such as within a building. The *Internet network* is a collection of local networks and gateways that use TCP/IP to function as a wide area network (WAN).

An *OS server* is a host that provides disk space for operating system software over the network. An *OS client* is a host that gets its system files from a disk that is physically connected to an OS server. A *servnet* refers to the collective unit formed by an OS server, its clients, and its *releases*, which are sets of software intended for a specific machine architecture and version of the operating system. To run a servnet, install DG/UX, set up DG/UX TCP/IP, and set up DG/UX ONC™/NFS® on the OS server for itself and for its clients. See the Glossary for a definition of these terms. For a more detailed discussion of these terms, see Chapter 1 of *Installing and Managing the DG/UX™ System.*

Understanding Network Structure

Networks can be complex. To help simplify them, designers organize networks into layers. The number of layers and each layer's function varies from network to network. In all networks, though, each layer provides a service to a higher layer, without the higher layer knowing the details of how the service is provided.

Each network system has its layers set up hierarchically. An *interface* consists of the types and forms of messages each layer uses to communicate with the layer above or below it. A protocol consists of the rules comparable layers in local and remote systems use to communicate. This set of layers, interfaces, and protocols is called the network's architecture. Figure 1-1 shows a four-layer network architecture.

```
Host A                              Host B

Layer 4 <-------Layer 4 protocol-------> Layer 4

Interface

Layer 3 <-------Layer 3 protocol-------> Layer 3

Interface

Layer 2 <-------Layer 2 protocol-------> Layer 2

Interface

Layer 1 <-------Layer 1 protocol-------> Layer 1
```

Figure 1-1 *Layers, Interfaces, and Protocols*

Data and control information are sent from the highest layer (layer 4) down to the lowest layer (layer 1). The highest layer contains the *user interface programs*. The user interface programs allow a user to communicate with the network. The lowest layer is the physical layer, where the two systems actually connect.

What Is DG/UX TCP/IP?

DG/UX TCP/IP is a package of communications software designed to run with DG/UX operating systems in the AViiON™ environment. The DG/UX system provides a device driver for the VME-based LAN controller for Data General's AViiON computer system, and for the Integrated Ethernet controller for Data General's AViiON workstation. DG/UX TCP/IP provides support for the Reverse Address Resolution Protocol (RARP), which is described later. DG/UX TCP/IP is a layered product that depends on DG/UX.

Figure 1-2 shows the DG/UX TCP/IP network architecture.

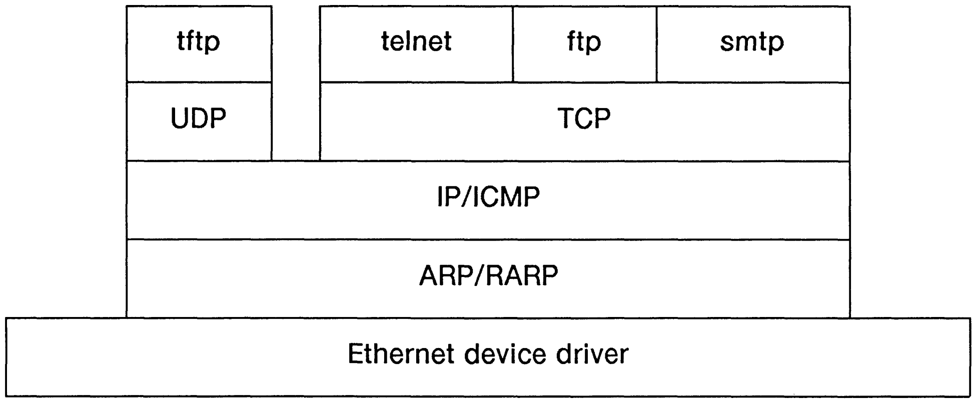

Figure 1-2 *DG/UX TCP/IP Network Architecture*

At the lowest layer of the DG/UX TCP/IP architecture is the Ethernet device driver. This software acts as an interface between the Ethernet controller and the computer's CPU.

At the next layer is the Address Resolution Protocol (ARP) and the Reverse Address Resolution Protocol (RARP). This layer provides a way to map hardware (Ethernet) addresses to protocol (Internet) addresses.

At the next layer is the Internet Protocol, or IP for short, and the Internet Control Message Protocol, or ICMP. This layer provides for the connectionless delivery, which means that data is transferred in well-defined bundles called packets, and each packet is treated independently of all the others.

At the next layer are two transport protocols, the Transmission Control Protocol, or TCP, and the User Datagram Protocol, or UDP. Transport protocols define how the destination host communicates with the source host. TCP, UDP, and IP are kernel-level protocols; that is, their operation executes kernel code.

Finally, there are four user-level programs shown in the diagram: **tftp**, **telnet**, **ftp**, and **smtp**. These allow users on different computer systems to interact with one another and with remote systems.

The *Defense Advanced Research Project Agency (DARPA)* developed the Internet protocols for the ARPANET network project. The University of California at Berkeley developed the 4.2 *Berkeley Software Distribution (BSD)* release of UNIX® based on the DARPA work. Data General developed the DG/UX TCP/IP software package from the BSD release, substantially revising it to comply with the *Defense Data Network (DDN)* specifications. Many BSD 4.3 features subsequently have been added to the DG/UX TCP/IP package.

Programmers can create applications for use on a TCP/IP network using TCP, UDP, or IP. For more information about how to do this, see *Programming With TCP/IP on the DG/UX™ System.*

DG/UX TCP/IP consists of several kernel-level protocols, servers to start daemons, administrative utilities, user commands, and user-level protocols.

Kernel-Level Protocols

DG/UX TCP/IP contains the following kernel-level protocols:

ARP – Address Resolution Protocol
: Used to map an Internet address to a physical hardware address (Ethernet address). ARP runs only across a single physical network, and runs only over networks that support hardware broadcast, such as Ethernet.

RARP – Reverse Address Resolution Protocol
: Used by a diskless system at startup to find its Internet address. A diskless client broadcasts a request that contains its Ethernet address, and the server responds by sending the client's Internet address to that Ethernet address.

IP – Internet Protocol
: A protocol that provides connectionless delivery of datagrams between hosts. Connectionless service means that the protocol treats each datagram as a separate entity. Each IP datagram contains the addresses of its source and destination, some control information, and the data transmitted. The protocol can deliver packets out of sequence, may drop packets, or may duplicate packets, but IP makes an earnest attempt to deliver packets. IP defines the exact format of data as it travels through a network, but delivery of data is not guaranteed.

ICMP – Internet Control Message Protocol
: A partner to IP that handles error and control messages. Gateways and hosts use ICMP to tell the other hosts about problems in delivering the datagrams. ICMP also allows a host to test whether a destination can be reached and whether it is responding.

TCP – Transmission Control Protocol
: A protocol that defines reliable, stream-oriented, process-to-process communication. TCP is a connection-based protocol; it requires a connection between communicating hosts before it transmits data. After a connection is established, TCP provides a two-way byte stream between communicating processes. Its messages include a protocol port number that allows the sender to distinguish between multiple programs on the remote host. TCP provides a checksum mechanism to guarantee that data has arrived intact. TCP uses IP to transmit information across a network.

UDP – User Datagram Protocol
: A protocol that defines datagram-based communication between a process on one host and a process on another host. UDP is a connectionless transport protocol. Its messages include a protocol port number that allows the sender to distinguish between multiple programs on the remote host. Data General's UDP provides a checksum mechanism to guarantee that data has arrived intact. UDP uses IP to transmit information across a network.

Servers to Start Daemons

DG/UX TCP/IP contains servers to initiate *daemons*, which are background processes that perform a system-wide public function. Each of these daemons operates at a specified port and provides service for a user protocol. You specify the port and services in the file **/etc/services**. User protocols are briefly described later in the chapter.

inetd The **inetd** server invokes network servers, or daemons, on demand. It also provides simple TCP-based services of its own. The following daemons are started by **inetd**. For more information, see the **inetd**(1M) manual page.

- **ftpd** The **ftpd** program, which is the File Transfer Protocol (FTP) server, is invoked by **inetd** when an incoming connection is detected on the specified port. FTP is briefly described later in this chapter. For more information, see the **ftpd**(1M) manual page.

- **telnetd** The **telnetd** program, which is the TELNET server, is invoked by **inetd** when an incoming connection is detected on the specified port. TELNET is described later in this chapter. For more information, see the **telnetd**(1M) manual page.

- **tftpd** The **tftpd** program, which is the Trivial File Transfer Protocol (TFTP) server, is invoked by **inetd** when an incoming connection is detected on the specified port. TFTP is briefly described later in this chapter. For more information, see the **tftpd**(1M) manual page.

- **rshd, rexecd, rlogind** These are servers or daemons for **rsh, rexec**, and **rlogin**. The **rsh** command is named **remsh** if you choose compliance with the *System V Interface Definition* (SVID); see *Setting Up and Managing TCP/IP on the DG/UX™ System* for details. These servers are invoked by **inetd** when an incoming connection is detected on the specified port. For more information, see the following manual pages: **rshd**(1M), **rexecd**(1M), and **rlogind**(1M).

named This is the name server for the domain name system, which is a distributed database that allows hosts on the Internet to share information. For details, see *Setting Up and Managing TCP/IP on the DG/UX™ System.*

pmtd This is the server for the magnetic tape pseudo device. The **pmtd** server handles local requests to do tape I/O operations on a tape device on a remote host. For more information, see the **pmtd**(1M) manual page.

routed The **routed** server manages network routing tables using the Routing Information Protocol (RIP). For more information, see the **routed**(1M) manual page.

rwhod This is the server for **rwho** and **ruptime**. For more information, see the **rwhod**(1M) manual page.

smtp This is the mail server program that must be started for **sendmail** to receive mail from other computers.

Administrative Utilities

DG/UX TCP/IP contains the following administrative utilities:

arp	Use the **arp** command to examine and change kernel ARP tables. (See Address Resolution Protocol (ARP) under the section "Kernel-Level Protocols" earlier in this chapter.) For more information, see the **arp**(1M) manual page.
hostid	The superuser can use the **hostid** command to set the hostid. Anyone other than the superuser can use the command to display the current hostid in hexadecimal. For more information, see the **hostid**(1C) manual page and *Setting Up and Managing TCP/IP on the DG/UX™ System.*
hostname	The superuser can use the **hostname** command to set the hostname. Anyone other than the superuser can use the command to display the current hostname. For more information, see the **hostname**(1C) manual page and *Setting Up and Managing TCP/IP on the DG/UX™ System.*
ifconfig	The **ifconfig** command assigns an address to a network interface, configures the network interface parameters, and stops and restarts an interface. For more information, see the **ifconfig**(1M) manual page. For examples of how to use this command, see *Setting Up and Managing TCP/IP on the DG/UX™ System.*
initrarp	The **initrarp** command initializes an OS server's ARP table. ARP and RARP use the ARP table to maintain Ethernet-to-Internet address translation information for all diskless clients. Use this command to initialize the ARP/RARP databases that contain information necessary for a remote boot (client Internet addresses). For more information, see *Setting Up and Managing TCP/IP on the DG/UX™ System* and the **initrarp**(1M) manual page.
netstat	The **netstat** command displays the contents of various data structures related to network activity. For example, you could use **netstat** to display the state of all sockets, to show the routing tables, or to display information about communication interfaces. For more information, see the **netstat**(1C) manual page. For examples of how to use this command, see *Setting Up and Managing TCP/IP on the DG/UX™ System.*

ping	The **ping** command tests whether a node on a network is up and working. It sends an ICMP echo packet to the specified host and expects the required ICMP response. If the ICMP packet is sent and received correctly, **ping** prints a message saying that the specified host is "alive." For more information, see the **ping**(1C) manual page.
route	The **route** command allows you to manipulate network routing tables. For more information, see **route**(1M). For examples of how to use this command, see *Setting Up and Managing TCP/IP on the DG/UX™ System.*
mailq	The **mailq** command prints the contents of the mail queue. It lists the queue IDs, the size of the message text in bytes, the date the message entered the queue, the sender of the message, any error messages generated, and the message recipients. For examples of how to use this command, see *Setting Up and Managing TCP/IP on the DG/UX™ System.*
newaliases	The **newaliases** command rebuilds the **sendmail** aliases database. It turns **/etc/aliases** entries into the databases **/etc/aliases.pag** and **/etc/aliases.dir**.

User Commands and User-Level Protocols

DG/UX TCP/IP contains the following user commands and user-level protocols:

bftp
The **bftp** command uses FTP to transfer files from one host to another in the background. You do not have to be directly involved when the file transfer takes place. For more information, see Chapter 4.

ftp
The **ftp** command implements the File Transfer Protocol (FTP). FTP allows you to transfer files from one host to another. FTP uses TCP as the transport level protocol. TCP was discussed earlier in the chapter. For more information about FTP, see Chapter 3.

nslookup
The **nslookup** command allows you to query domain name servers directly. For more information, see *Installing and Managing the DG/UX™ System.*

R commands
The R commands allow you to obtain information from, to log in to, and to execute commands on a remote host. DG/UX TCP/IP includes the following R commands:

rcp	Allows you to copy files between hosts on the network.
rlogin	Allows you to login to another host over the network.
rsh (remsh)	Connects to a specified host and executes a specified command. If you choose SVID compliance during setup of DG/UX TCP/IP, the command is **remsh**. If you choose not to comply with the SVID during setup, the command is **rsh**.

rwho	Produces a list of all users logged in to all hosts on the local network, as long as the hosts are running **rwhod**.
ruptime	Shows the status of each machine that is on the local network and running **rwhod**.

Some of the R commands use TCP as the transport level protocol, and some use UDP. For more information about R commands, see Chapter 6.

sendmail

The **sendmail** command implements the Simple Mail Transfer Protocol (SMTP), which allows the transmission of mail messages. The **sendmail** program uses TCP as the transport level protocol. It is not a command you will run directly. You will use it while sending mail with **mailx**. The **sendmail** command has associated programs called **mailq**, **newaliases**, and **smtp**. For information about how to configure and use **sendmail**, see *Setting Up and Managing TCP/IP on the DG/UX™ System*.

telnet

The **telnet** command implements the TELNET protocol. TELNET allows a user on one host to interact with a remote host as if the terminal is directly connected to the remote host. TELNET uses TCP as the transport level protocol. For more information about TELNET, see Chapter 2.

tftp

The **tftp** command implements the Trivial File Transfer Protocol (TFTP). TFTP allows file transfer with minimal capability and overhead. The **tftp** command depends on the UDP protocol, which was discussed earlier in this chapter.

TFTP is also used during a first stage boot with Data General's AViiON computer systems. The boot program, once it determines its Internet address, uses TFTP to transfer a file that contains the executable image of a second stage boot program. For more information about TFTP, see Chapter 5.

The next chapter discusses TELNET.

End of Chapter

Chapter 2 Understanding and Using the TELNET Protocol

The TELNET protocol provides virtual terminal service on remote systems. That is, it lets you log in to a remote host from a terminal that is connected to your local host. TELNET allows your terminal to perform as if it were connected directly to the remote host. It does this by using a Transmission Control Protocol (TCP) connection between your local host and a remote host to transmit data and control information.

You access the TELNET protocol through the **telnet** command. (TELNET in uppercase letters refers to the protocol, while **telnet** in boldface lowercase letters refers to the program you invoke on the command line.) There are two modes of operation in **telnet**: local (command) mode and remote mode. You use local mode to enter any of the **telnet** commands. These commands allow you to make or break network connections with other systems, display information about a TELNET session, create a child process, and terminate local mode. For more information on **telnet** commands, see "Using the telnet Commands" later in this chapter.

Once you make a network connection with another system, you enter remote mode. You interact with a remote host in remote mode. While in remote mode, **telnet** allows you to return to local mode to execute **telnet** commands if you so choose. For more information about moving between remote and local modes, see "Leaving Remote Mode" later in this chapter.

Understanding the Network Connection Through TELNET

Using the TELNET protocol requires running two programs: **telnet** and **telnetd**. When you want to log in to a remote host, you invoke the **telnet** program on the local host. This puts you in **telnet** local mode.

In local mode, you can enter commands to establish a connection with a remote host. When you enter such a command, the **inetd** server on the remote host hears a request for network service, invokes a **telnetd** server, and establishes a TCP network connection between a local and remote host. (For more information about **inetd**(1M), see the manual page.) TELNET uses this network connection to transmit data and control information. The **inetd** server passes control of the connection to **telnetd**, and continues to listen for other requests for network service.

Once you have formed a network connection, **telnet** enters remote mode. In **telnet** remote mode, your terminal performs as if it were connected directly to the remote host. (You can use different types of terminals to connect to the same remote port, but a particular port may not be configured for your type of terminal.) The **telnetd** server assigns the connection a *pseudo-terminal* to act as your terminal on the remote host. For details about pseudo-terminals, see the **pty**(7) manual page. Figure 2-1 shows this sequence of events.

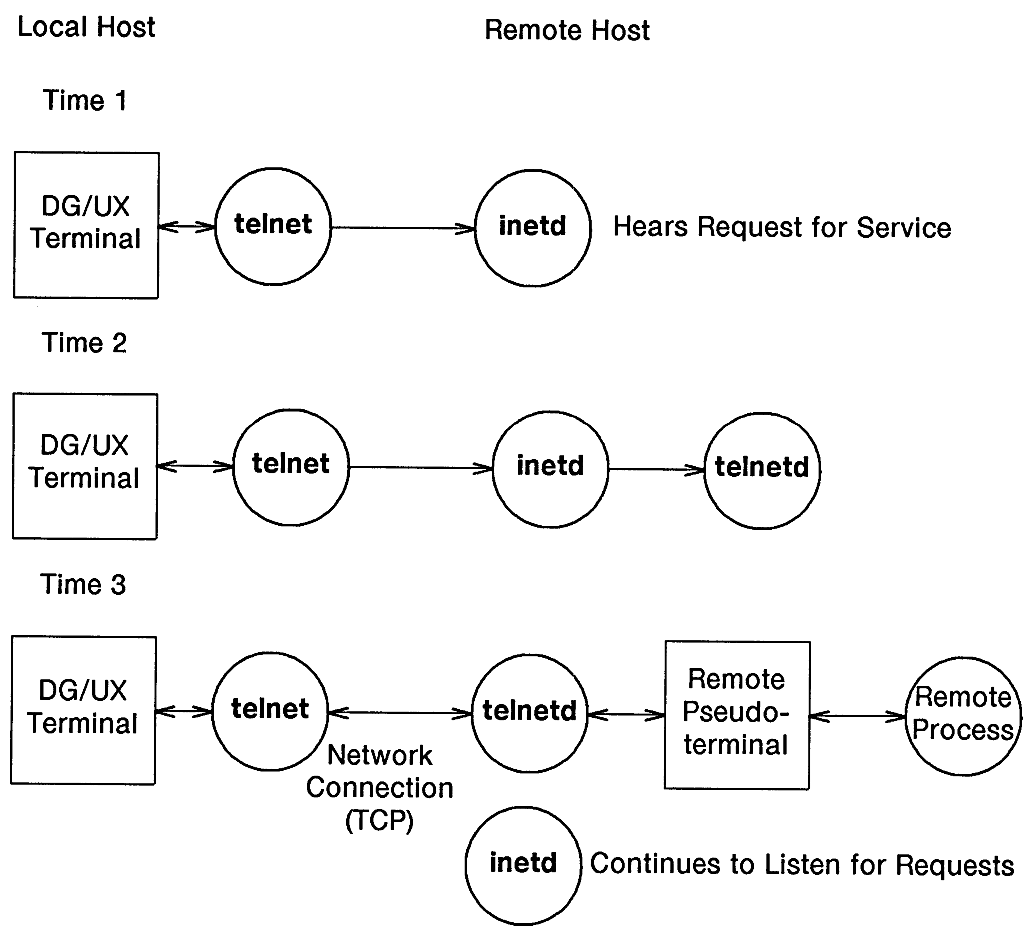

Figure 2-1 *Establishing a Connection Between telnet and telnetd*

Forming a Network Connection with telnet

You know how a connection is established, so now you need to know how to initiate this sequence of events. To form a network connection, first execute the **telnet** command. To execute **telnet** from the shell, type it with no argument and then press New Line.

```
$ telnet ↵
```

This puts you in **telnet** local mode. The local mode prompt appears:

```
telnet>
```

You can now enter any of the **telnet** commands. See "Using the telnet Commands" for a complete overview of the commands available, and the "telnet Command Dictionary" for a complete description of each command.

To form a network connection with a remote host, use the **telnet open** command. Enter the **open** command with the hostname as an argument. For example, enter:

```
telnet> open remote3 ↵
```

to form a network connection with a host named **remote3**.

If your **open** command succeeds, your terminal displays a message confirming your connection, a message identifying the default escape character and the login banner of the remote host's operating system. You are in remote mode while working on the remote host's operating system. Everything that happened between **telnet, inetd**, and **telnetd** was transparent to you.

The discussion up to now has implied that the only way you can open a network connection to a remote host is to enter local mode and use the **open** command. This is not true. You can execute **telnet** and the **open** command in one step. For example:

```
$ telnet remote3 ↵
```

displays the login banner of the remote operating system. You are instantly put into remote mode.

NOTE: To work on the remote host, you must have user privileges on the remote system (see the remote host's system administrator to obtain these privileges).

Now, you can use the remote host as if it were your local host. Refer to the remote host's operating system manuals on how to use the system.

Terminating a Network Connection

This section shows you how to terminate a TELNET network connection. It shows you how to return to **telnet** local mode or the shell.

Leaving Local Mode

To leave **telnet** when in local mode and return to the shell, use either the **quit** or **bye** command, which are identical. For example:

```
telnet> quit ↲
```

closes any connection you have and returns you to the shell.

Leaving Remote Mode

There are two methods of leaving remote mode. Table 2-1 lists the methods of leaving remote mode and the results.

Table 2-1 Leaving telnet Remote Mode

Method	Result
Log out of the remote system	Most remote hosts will close the connection when you log out. You return to the **telnet** local mode. Some versions of **telnetd** terminate the present remote mode and begin prompting you for your username to begin a new remote mode process. If so, use the escape character and then enter **quit**.
Escape character	This character returns you to local mode and lets you execute a command. You can then use the **quit** command to return to the shell on the local machine.

To leave remote mode and return to the shell, log out of the remote system. Enter **exit** to log out. You will see the following on your screen:

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

You are now in local mode. Use **bye** or **quit** to return to the shell on your local host.

As an alternative, while in remote mode, you can return to the local mode by using the escape character (in the DG/UX system, the default escape character is CTRL-]). For example, hold down the Ctrl key and press the right square bracket (]). You will see the following on your screen:

```
telnet>
```

You are now in local mode. Use **quit** to return to the shell on your local host.

You may see the following error message while in local mode:

```
illegal command: syntax error at foo
```

where *foo* is the first word **telnet** did not recognize. This error message appears when you type a command incorrectly or enter a command that **telnet** does not support.

Displaying Information About TELNET

You can use the **help** and **status** commands, executed in **telnet** local mode, to display information about **telnet** commands and parameters respectively.

Using the telnet help Command

While in **telnet** local mode, you can check the syntax or meaning of any **telnet** command. For a list of the available commands, enter the **help** command. For example, enter the following to display all available **telnet** commands:

```
telnet> help ⤶
```

The ? command is identical to the **help** command. Entering the ? command displays a list of all available **telnet** commands.

If you want information on a particular command, enter the **help** or ? command with the command as the argument. For example, the following command displays the syntax line and the definition of the **escape** command:

```
telnet> help escape ⤶
```

Using the telnet status Command

The **status** command displays information about your TELNET network connection. It lists the following information:

- The current escape character
- Whether or not an outstanding connection exists
- Whether or not debugging mode is on
- Whether or not option negotiation messages between **telnet** and **telnetd** are displayed (see the **options** command later in this chapter)
- A list of terminators
- The current representation of the NVT keyboard
- The current state of negotiated options

For example, enter the following to find out if an outstanding connection exists:

```
telnet> status ↵
```

Using the telnet Commands

While in **telnet** local mode, you can use any of the **telnet** commands. The following two tables list the commands. Table 2-2 includes *basic* commands. Table 2-3 includes special *customizing* commands.

The *basic* commands allow you to form a connection with a remote host, display information about your TELNET session, terminate one network connection to form another, create a child process from local mode, and terminate the session to return to your local host (see "Using the Basic Commands," later in this chapter).

The special *customizing* commands allow you to send special characters untranslated, change special characters, and change modes (see "Using the Customizing Commands" later in this chapter).

Abbreviating telnet Commands

The **telnet** command will accept the first three characters of any of its commands. For example, **telnet** will accept **sta** for **status** and **clo** for **close**.

NOTE: **telnet** is case-sensitive. You must type commands in lowercase. **telnet** does not recognize commands in uppercase.

Providing Arguments to telnet Commands

The following list describes the arguments you will be using with the **telnet** commands. Brackets ([ ]) around an argument indicate that it is optional.

host	The name of the remote system you want to use. It is a sequence of characters that does not contain any of the following: a blank space, a tab, a new line, or double quotation marks (" ").
port	An integer representing a port number to which you want to be connected.
option	Any one of the following communications features: suppress-go-aheads (SGA), foreign echoing (EC), binary input (BI), binary output (BO), status (ST), timing mark (TM), or extended option (EX). For more information, see "Requesting Negotiation" later in this chapter.
type	Any one of the following values: on, off, always, or never. For more information see "Requesting Negotiation" later in this chapter.
NVT_char	NVT special characters include the following: interrupt process (IP), abort process (AP), are you there (AYT), break (BRK), erase character (EC), erase line (EL), end of record (EOR), and the synch signal. For more information on NVT characters, see "Understanding the NVT Characters" later in this chapter.
string	A sequence of characters enclosed in double quotation marks (" "). Strings follow the same syntax as string constants in the C programming language. They can be any length up to 200 characters. You can insert a new line character as text in a string with the **\n** sequence. You can insert a tab character with the **\t** sequence.

Table 2-3 telnet Customizing Commands

Command	Default	Meaning
crmod	Terminal's CR value	Turns on or off carriage return mode. In this mode, carriage return characters received from the remote host are mapped into a carriage return and a line feed.
debug	Off	Turns debug mode on or off. Certain commands are only available in this mode. These commands are **mode, map**, and **listen**. These commands are defined in this table.
escape *string*	^]	Lets you set the **telnet** escape sequence. Be careful about your choice of escape sequences. For example, do not set your escape sequence to the Esc key if you plan to use either the **vi** or EMACS editor.
listen [*port*]	Not applicable	Can only be used in debug mode. The **listen** command instructs **telnet** to wait for an incoming connection on the port number indicated. If you don't enter a port number, **telnet** assigns one.
map *string NVT_char*	Not applicable	Can only be used in debug mode. **Map** lets you substitute a string of your choice for any NVT character (see "How telnet Establishes Network Connections" later in this chapter for an explanation of NVT characters).
mode *option type*	Not applicable	Can only be used in debug mode. This command changes the mode regardless of the negotiated option.
negotiate *option type*	Not applicable	Can only be used in in debug mode. This command lets you request negotiation to change an option (see "How TELNET Establishes Connections" later in the chapter for an explanation of option negotiation).

(continued)

Using the Basic Commands

Table 2-2 lists basic **telnet** commands.

Table 2-2 Basic telnet Commands

Command	Meaning
open *host* [*port*]	Form a connection with a remote host.
close	Terminate the immediate TELNET session and return to local mode.
quit or **bye**	Terminate an open TELNET session and exit from **telnet**.
z	If using C shell, suspends **telnet** and returns you to the shell. In the Bourne shell, this command will do nothing.
help or **?**	Prints a summary of **telnet** commands with their definitions.
resume	Returns to an outstanding TELNET network connection from local mode.
status	Displays your TELNET connection's parameters.
shell or **!**	Create a shell process from **telnet**.

Using the Customizing Commands

The special customizing commands allow you to change characters and modes to suit your tastes. These commands have default values, so you may skip this section if you do not want to change them. Table 2-3 lists the special **telnet** commands. For more information, see "Summarizing telnet Commands" later in this chapter.

Table 2-3 telnet Customizing Commands

Command	Default	Meaning
options	Off	Turns on or off the display of any option negotiation between **telnet** and **telnetd** over the network.
prompt *string*	`telnet>`	Lets you change the command prompt.
send *NVT_char*	Not applicable	Lets you send an NVT character (for example, AO) through the network without being interpreted.
terminator *string*	NL, IP, AO, AYT, BRK, EOR, synch	Lets you specify additional characters as terminators. In line mode, terminators determine when to ship characters through the network.
unterm *string*	Not applicable	Lets you cancel terminator status on any terminator defined.

(concluded)

Using telnet Command Line Options

You can invoke some of the **telnet** commands above by using a **telnet** command line option. You can invoke commands without having to enter **telnet** local mode first. Table 2-4 shows the available command line options, their corresponding **telnet** commands, and their definitions.

Table 2-4 telnet Command Line Options

Option	Command	Definition
−i*s_type*	**negotiate**	Negotiate binary input option.
−o*s_type*	**negotiate**	Negotiate binary output option.
−d	**debug**	Turns debug mode on.
−e*s_type*	**negotiate**	Negotiate remote echo option.
−s*s_type*	**negotiate**	Negotiate remote side suppress-go-ahead option.
−l[*port*]	**listen**	Listen for connections on the given port number.

The argument *s_type* indicates whether or not you want the option. You must substitute either the letter **a** for always or the letter **n** for never. (See "Negotiating Options" later in this chapter for an explanation of types.)

For the argument *port*, substitute the port number of the connection you are monitoring. See your local system administrator for port numbers.

For example, to negotiate with the remote system to always suppress-go-aheads, type the following:

```
$ telnet −sa ↵
```

NOTE: Remember that **telnet** will not return any evidence of your request. You can check all options, however, with the **status** command after the connection has been established.

You can use more than one option on an invocation of **telnet**. Separate each option with a space. For example, to always suppress-go-aheads and always have local echoing, you would type:

```
$ telnet -sa -ea ↵
```

CAUTION: If you use the debug option (**−d**), it must appear before any other options on the command line.

How telnet Establishes Network Connections

The following sections briefly discuss how **telnet** makes a network connection.

The connection to the other system is defined in terms of a Network Virtual Terminal, or NVT. You can think of the NVT as an imaginary device that provides a standard intermediate representation of a terminal.

The NVT includes a way for the local host (TELNET user) and remote host (TELNET server) programs to negotiate *options* to achieve the best possible service between the two systems. Options represent a variety of communications features.

Option negotiation during a connection is invisible unless you have the **options** command turned on. You can inspect these options after negotiation with the **status** command (see "Displaying Information About TELNET" earlier in this chapter). If you want to change the value of an option, you can request negotiation on the option while in local mode (see "Negotiating Options" later in this chapter). Alternatively, you can request option negotiation when you invoke **telnet** with command line options. These options allows you to issue commands to **telnet** at invocation, before entering local mode (see "Using telnet Command Line Options" earlier in this chapter).

Understanding the NVT

The NVT is an imaginary character device that sends and receives characters from the remote and local hosts. This imaginary device provides standard definitions for characters on a terminal. Hosts from different machines can communicate by mapping their local device characteristics and conventions to match the NVT.

The NVT consists of a virtual printer and virtual keyboard. The virtual printer responds to incoming data from the remote host and the virtual keyboard produces outgoing data from the local host. The NVT provides standard representations for characters transferred between hosts.

By default, the NVT functions in *line mode*. In line mode, characters are buffered at the local terminal until a terminator character, such as new line, is pressed. Once the terminator is pressed, the entire buffer of data is transferred to the remote end.

Some options, such as suppress-go-ahead (SGA), may change line mode to *character mode*. In *character mode,* characters are transferred as they are typed (see "Negotiating Options" later in this chapter).

Understanding NVT Printer Codes

The NVT printer can represent all 95 USASCII codes (codes 32-126). It ignores the other codes (0-31, 127 and the uncovered codes 128-255) except those defined in Table 2-5.

Table 2-5 NVT Printer Codes

NVT Code	DG/UX System	Meaning
NULL	0	No operation.
Line Feed (LF)	new line	Moves the printer to the next print line.
Carriage Return (CR) *CR-NL*	new line	Moves the printer to the left margin of the next line.
CR-NULL	CR	Moves the printer to the left margin of the current line.
Bell (BEL)	^G	Produce an audible or visible signal.
Back Space (BS)	Read user stty	Moves the printer left one space.
Horizontal Tab (HT)	Tab	Moves the printer to the next horizontal tab stop.
Vertical Tab (VT)	5 new lines	Moves the printer to the next vertical tab stop.
Form Feed (FF)	20 new lines	Moves the printer to the top of the next page.

Understanding the NVT Keyboard

The NVT keyboard contains keys for generating all 128 ASCII codes. Some of these codes, however, have no effect on the NVT printer. The keyboard can generate the following additional codes shown in Table 2-6.

Table 2-6 NVT Keyboard Codes

NVT Keyboard	DG/UX System	Meaning
Abort Output (AO)	Defined by user	Suppresses output to the user's terminal.
Are You There (AYT)	Defined by user	Provides evidence that the system is running.
Break (BRK)	ioctl break character	Sends the appropriate break character to the remote process.
Erase Character (EC)	ioctl erase character	Deletes user's preceding character or printed position.
Erase Line (EL)	ioctl kill character	Deletes all data on the current line of input.
End of Record (EOR)	Defined by user	Transmits input buffer as though a terminator character is read.
Interrupt Process (IP)	ioctl interrupt character	Suspends, interrupts, aborts, or terminates remote user's process.
Synch signal	Defined by user	A TCP urgent notification with the command Data Mark (DM).

Understanding the NVT Characters

Most remote systems provide the functions listed above. How you invoke these functions can vary from system to system. NVT uses the following standard definitions for these functions:

Abort Output (AO)
: Allows a process to run to completion, but does not send the output to the user's terminal. It clears the output buffer that has not yet been printed on the terminal.

Are You There (AYT)
: Provides the user with visible evidence that the system is still up and running.

Break Character (BRK)
: Sends the appropriate break character to the remote process.

Erase Character (EC)
: Deletes the last preceding character or *printed position* the user types. Printed position means several characters that are a result of overstrikes.

Erase Line (EL)
: Deletes all the data on the current line of input.

End of Record (EOR)
: Allows the user to flush the input buffer before a terminator character is encountered.

Interrupt Process (IP)
: Suspends, interrupts, aborts, or terminates a user process. Use this function, for example, to stop a process that is in an infinite loop.

Synch Signal
: Consists of a TCP urgent notification with the TELNET command Data Mark (DM). The urgent notification is not affected by the flow control of the TELNET connection. The data stream is immediately scanned for interesting signals and data that was sent between the urgent notification and the DM. Interesting signals include: IP, AO, and AYT; the local analogs of these standard signals; and all other TELNET commands. Once the DM occurs, the urgent data conditions are serviced and the receiver can return to normal operation. Every urgent notification must be followed (eventually) by a DM.

You can change the default key bindings of these functions. See "Using the telnet Commands" earlier in this chapter.

Using TELNET Option Negotiation

TELNET uses *option negotiation* to accommodate hosts that provide more services than those available on the NVT. Option negotiation is the way two hosts agree on a set of conventions for a given connection. The options allow hosts to negotiate conventions, such as terminal type, character sets, and echo mode. Negotiation occurs in a Do, Don't, Will, Won't structure.

A host begins option negotiation by sending a Will (*n*) or Do (*n*), where *n* is the chosen option. Will (*n*) indicates that the sender offers to begin performing option (*n*). The sending host must wait for a positive or negative acknowledgement from the receiver. A positive acknowledgement from the receiving host will be Do (*n*); a negative acknowledgement will be Don't (*n*).

When a host sends a request for a Do (*n*), the sending host wants the receiving host to perform the option. The receiving host sends either Will (*n*) for a positive acknowledgment or Won't (*n*) for a negative acknowledgement.

Negotiating Options

When a connection is first established, the user and server programs negotiate back and forth for the best possible service. You can see this negotiation if you use the **option** command. (See "Using the telnet Commands" earlier in this chapter.) If an option is rejected, the negotiation does not repeat.

Once the connection is established, options are negotiated when another program (e.g., text editor) begins. You can see this negotiation if you use the **option** command. (See "Using the telnet Commands" earlier in this chapter.) Options that were rejected during the initial connection may be renegotiated at this time.

Options are negotiated when:

- A connection is first established.
- A local or remote condition changes.
- A user requests negotiation.

You can request negotiation on options when invoking **telnet** or when you're in local mode. See "Requesting Negotiation" later in this chapter for details. The available options are:

Transmit-binary (**BI, BO**)
: Passes all characters through without interpretation or translation (except IP, AO, EC, EL, EOR, and AYT). The default settings are Won't transmit binary and Don't transmit binary.

Echo (**EC**)
: Displays characters typed from the user's keyboard on the user's screen. The default settings are Won't echo and Don't echo.

Suppress-go-ahead (**SGA**)
: Stop sending the Go Ahead (GA). The GA coordinates transmission between the terminal and the computer. The GA signals the computer and the terminal when one is finished and the other can transmit data. Suppress-go-ahead turns on character-by-character mode. The default settings are: Won't suppress-go-ahead and Don't suppress-go-ahead.

Status (**ST**)
: Requests the receiver to send the current values of the TELNET options. The default settings are Won't status and Don't status.

Timing-mark (**TM**)
: Measures the round-trip delay between two processes or between a process and a terminal; flushes all characters typed after an erroneous command; or throws away output from a remote process until it receives a Will or Won't timing-mark. The default settings are Won't timing mark and Don't timing-mark.

Extended-options-list (**EX**)
: Allows another 256 options to be added to the current options. The default settings are Won't extended-options-list and Don't extended-options-list.

Requesting Negotiation

You can request negotiation on any of the above options with the command **negotiate**. You can only *request* negotiation, not send an announcement of the current mode. Your request may be denied. **telnet** will not stop to give you a response to your request. If you want to see the results, you must use the command **status** or **options** (see "Using the telnet Commands" earlier in this chapter).

You use the command **negotiate** by typing:

```
telnet> negotiate option type
```

where *option* is the option you want to change and *type* is the value you want. Table 2-7 shows the values available for *option*.

Table 2-7 TELNET Negotiation Option Codes

Option Code	Meaning
BI	Binary Input
BO	Binary Output
EC	Foreign Echoing
SGA	Suppress-go-aheads
ST	Status
TM	Timing Mark
EX	Extended Option

Table 2-8 shows the values available for *type*.

Table 2-8 TELNET Negotiation Types

Type	Meaning
on	Try to negotiate an option on. All future requests to negotiate the option off will be honored.
off	Try to negotiate the option off. All future requests to negotiate the option on will be honored.
never	Demand an option be negotiated off and left off. All future requests to negotiate the option on will be refused.
always	Try to negotiate an option on. All future requests from the server program to negotiate the option off will be honored. However, the user program immediately sends one request to the server to negotiate the option back on.

For example, if you want to request binary input, you would type the following while in local mode:

```
telnet> negotiate BI on ↵
```

NOTE: **telnet** will not return information on your request. If you are connected to a remote host, you will return to remote mode. If you are not connected to a remote host, you will remain in local mode.

Summarizing telnet Commands

The following is an alphabetical listing of commands **telnet** recognizes. Along with a format line is a description of the command and an example of how to use the command. When using these commands, remember the following:

- You can only use these commands in local mode.
- **telnet** is case-sensitive. You must type commands in lowercase. **telnet** does not recognize commands in uppercase.
- You can abbreviate any of the commands by typing the command's first three letters.

bye

Terminate a network connection and local mode.

Format

bye

Description

The **bye** command terminates local mode and returns you to the shell. Any outstanding connections are terminated as well. The **quit** command is identical to the **bye** command.

Example

```
telnet> bye ↵
$
```

The TELNET network connection terminates and you return to the shell.

close

Terminate any connection to a remote host.

Format

close

Description

Use this command to terminate an outstanding remote connection, but remain in local mode.

Example

```
telnet> close ↵
Connection closed.
telnet>
```

Terminates the network connection.

Error Message

`Connection does not exist.`	You do not have a network connection to clear.

crmod

Turn carriage return mode on or off

Format

crmod

Description

Use this command if, while in remote mode, your terminal is displaying two or more lines of output on the same line during a TELNET network connection. This command adds a line feed to any carriage return received from the remote process. First, re-enter **telnet** local mode, then enter the **crmod** command.

A carriage return usually moves the cursor over to the left margin and then moves down one line. But, sometimes a carriage return will only move the cursor to the left margin and not move down one line. If this happens, use the **crmod** command.

NOTE: The value mapped to carriage return for your terminal in **stty**(1) is the default.

Example

```
telnet> crmod ↵
telnet>
```

Makes any carriage return move to the left margin, and move down one line.

debug

Turn your ability to use certain commands on or off

Format

debug

Description

This mode enables you to use the following commands: **listen, map,** and **mode.** These commands let you wait for an incoming connection, change character mappings, and change modes without negotiating options.

To prevent accidental use, we only offer these commands in debug mode. These commands can change the way characters are sent to or received from your terminal. Be careful when using them.

Beware when using debug mode. **telnet** commands can be unpredictable when used in this mode.

Example

```
telnet> debug ↵
Debug mode is on.
telnet>
```

telnet is now operating in debug mode. You can use the commands **listen, map,** and **mode.**

escape

Change the telnet escape character

Format

escape *string*

Description

Lets you substitute a string of your choice for the default **telnet** escape character. Be careful about your choice of sequence. For example, if you use the editor **vi**, do not set the escape character to the Esc key. The **vi** editor uses the Esc key to delimit the insert command. You may not be able to exit from insert mode.

Example

```
telnet> escape x ↵
```

Changes the **telnet** escape character to the string represented by **x**.

help

Display telnet command definitions.

Format

help [*command*]

Description

Use this command to display a list of **telnet** commands with their definitions. For more information on the syntax and meaning of a particular command, enter **help** with the command as an argument. The ? command is identical to the **help** command.

Example

```
telnet>   help ↵
```

Displays all available **telnet** commands. If you want information on a particular command, enter the **help** or ? command with the command as the argument.

```
telnet> help status ↵
```

Displays the syntax line and meaning of the **status** command.

Error Messages

`Syntax error` or `File does not exist`	You did not enter enough characters to make the command recognizable, or there is no such command.

listen

Listen on a port for an incoming connection.

Format

listen [*port*]

Description

You can only use this command in debug mode. This command monitors the specified port. If you do not specify a port, **telnet** will pick one and announce it. **telnet** waits until the connection is established. You can abort the process with a user-defined interrupt character before the connection is established. You can use the escape character to abort after the connection is established.

Example

```
telnet> listen ↵
listening on 3001
```

telnet chooses a port number and waits until the connection is established.

Error Messages

`No such port.`	The port number you specified is undefined.

log
Control output logging.

Format

log [*logfile*]

Description

Use this command to make **telnet** put the data that the remote host sends to your terminal into a file. Logging starts when you specify the **log** command with the name of a *logfile* into which **telnet** should put data. Specify **log** without an argument to stop logging data from the remote host.

If the *logfile* does not exist, it will be created. If the file already exists, **telnet** will append the data to the end of the file.

Example

```
telnet> log output_logfile ↵
telnet>
```

Here, **telnet** copies data from the remote host into **output_logfile**.

Example

```
telnet> log↵
telnet>
```

The above command closes an open *logfile*.

Output generated by **telnet**, such as the output of the **status** command or locally echoed characters, will not be written to the *logfile*. Only one *logfile* can be open at a time. An attempt to open a second *logfile* causes an error. An attempt to close the *logfile* when output is not being logged also causes an error.

map

Substitute a string for an NVT character

Format

map *string NVT_char*

Description

You can only use this command in debug mode. It substitutes a string of your choice to use in place of the default NVT character. When you type your chosen *string*, TELNET sends the special NVT character. The NVT characters are listed below. (See "Understanding the NVT Keyboard" earlier in the chapter for descriptions.)

- Abort output (AO)
- Are you there (AYT)
- Break (BRK)
- Erase character (EC)
- Erase line (EL)
- End of record (EOR)
- Interrupt process (IP)
- Synch Signal (sync)

Example

```
telnet> map x IP ↵
```

The string represented by *x* replaces the default IP character. **telnet** returns to remote mode if you have a connection pending. Otherwise, **telnet** remains in local mode.

mode

Change mode regardless of negotiated option.

Format

mode *option type*

Description

You can only use this command in debug mode. This command lets you change modes without negotiating the option. The following table shows you the valid values of *option*:

Option	Definition
BI	Binary input
BO	Binary output
EC	Local echoing
LI	Line mode

Substitute one of the following strings for *type*:

on Turns on the mode regardless of the option that normally controls it. However, if the option is negotiated after your change, the mode changes to correspond with the change in the option.

off Turns off the mode regardless of the option that normally controls it. However, if the option is negotiated after your change, the mode changes to correspond with the change in the option.

never
Turns the mode off and leaves it off regardless of the option. The mode changes when you invoke the change or when you close the connection. When you close the connection, the mode returns to the default setting.

always
Turns the mode on and leaves it on regardless of the option. The mode is changed when you invoke the change or when you close the connection. When you close the connection, the mode returns to the default setting.

CAUTION: When using the option types **never** and **always**, you could ask for a particular option that the server does not want. In such a case, you can expect unusual results during the connection.

Example

```
telnet> mode LI on ⤶
```

This command changes data transfer over the network to line mode. Data will be sent across the network when a terminator (for example, a new line) is read.

negotiate

Request negotiation on an option.

Format

negotiate *option type*

Description

This command lets you request negotiation to change an option. Negotiation takes place only if the option type you specify is different from the current one.

You can only request a negotiation; you cannot send an announcement of the current mode. If you have the command **options** turned on, you can see the negotiation take place (see **options** later in this section). Otherwise, **telnet** does not notify you whether the change has been made or not. Use the command **status** to see the results.

The table below shows the strings that you can substitute for *option* and their corresponding meanings:

Option Code	Meaning
BI	Binary Input
BO	Binary Output
EC	Foreign Echoing
EX	Extended Option
SGA	Suppress-go-aheads
ST	Status
TM	Timing Mark

Substitute one of the following strings for *type*:

on Try to negotiate an option on. All future requests to negotiate the option off will be honored.

off Negotiate an option off. All future requests to negotiate the option on will be honored.

never
Demand an option to be negotiated off and left off. All future requests to negotiate it on will be refused. You must know whether or not the server will abort when that option is refused.

always
Try to negotiate an option on. All future requests from the server program

to negotiate the option off will be honored. However, the user program immediately sends one request to the server to negotiate the option back on.

Example

```
telnet> negotiate SGA on ↵
```

telnet negotiates to turn on the option suppress-go-aheads. If successful, **telnet** transfers each character as you type it. Otherwise, **telnet** transfers characters in line mode.

open

Connect your terminal to the login process on a remote host.

Format

open *hostname* [*port*]
open *Internet_address* [*port*]

where:

hostname	is the name of the desired remote host.
Internet_address	is the address the network manager assigned to the remote system. The address is in the following format: **128.223.8.48** For more information on Internet addresses, see Chapter 2 in *Setting Up and Managing TCP/IP on the DG/UX™ System.*
port	is the port number you want to assign to this connection.

Description

Use this command when you want to form a network connection with a remote host.

Examples

```
telnet> open nic ↵
```

Forms a network connection using the remote host's name.

```
telnet> open 128.223.8.48 ↵
```

Forms a network connection by using the remote host's Internet address.

Error Messages

`?Already connected to` *hostname*	You have already formed a network connection. You must use the **close** command to close this connection before you can form a new one.
`telnet:connect: Can't assign requested address`	The *hostname* that you requested is not in your network and there is no kernel route entry to the network that contains the destination host.
`telnet:connect: Connection refused`	There is no server process (**inetd** or **telnetd**) running on the destination host.
`telnet:connect: Connection timed out.`	The destination host is down. However, you do have a route to the network on which the destination host is located and there is an entry for the destination host in the host database.
`hostname: Unknown host.`	The destination host does not have an entry in the host database.

options

Toggles display of messages between **telnet** and **telnetd.**

Format

options

Description

Use this command when you want the results of option negotiation displayed on your terminal. To turn off the display, enter the **options** command again. Use the **status** command to see if options will be displayed.

Example

```
telnet> options ↵
Will show option processing.
telnet>
```

Displays the sequence of messages between **telnet** and **telnetd.**

```
telnet> options ↵
Wont show options processing.
telnet>
```

Turns off the display of messages between **telnet** and **telnetd.**

prompt

Change the command prompt.

Format

prompt "*string*"

Description

This command lets you substitute a *string* of your choice in place of the normal command prompt. You must enclose the *string* in double quotation marks (" ").

Example

```
telnet> prompt "net47" ↵
net47
```

The command prompt changes from telnet> to net47.

quit

Terminate a network connection and telnet local mode.

Format

quit

Description

The **quit** command will terminate **telnet** local mode and return you to the shell. The **bye** command is identical to the **quit** command.

Example

```
telnet> quit ↵
$
```

The network connection closes and you return to the shell.

resume

Exit from local mode and continue any suspended remote mode.

Format

resume

Description

Use this command to return to any suspended network connection. Once back in remote mode, you will get a message notifying you that you are in remote mode.

Example

```
telnet> resume ↵
Returned to remote mode.
```

Returns to **telnet** remote mode.

send

Send NVT character to a remote process.

Format

send *NVT_char*

Description

Use this command when you want to send NVT special characters across your network connection. You can substitute any of the NVT special characters for *NVT_char*. The table below lists and defines the NVT characters:

NVT Character	Meaning
Abort Output (AO)	Suppresses output to the user's terminal.
Are You There (AYT)	Provides evidence that the system is running.
Break Character (BKR)	Sends the appropriate break character to the remote process.
Erase Character (EC)	Deletes user's preceding character or printed position.
Erase Line (EL)	Deletes all data on the current line of input.
End of Record (EOR)	Sends current contents of input buffer.
Interrupt Process (IP)	Suspends, interrupts, aborts, or terminates user's output.
Synch (sync)	A TCP urgent notification with a Data Mark (DM).

Example

```
telnet> send IP ↵
Returned to remote mode.
```

Sends the character represented by IP to the remote host, and continues remote mode.

Error Messages

`Connection does not exist.`	You do not have a network connection so there is no remote host to receive the argument.
`Illegal command: syntax error at` *word*	You entered an argument that is not an NVT character.

shell

Create a shell process as a child of **telnet**.

Format

shell [*command*]

Description

Use **shell** to invoke the shell on your local host without terminating **telnet**. If you have a network connection, use the escape character (^]) to return to **telnet** local mode before executing the **shell** command. When you terminate the child process, you will return to your network connection. If you do not have a network connection, you will return to local mode. Terminate the child process by entering the shell **exit** command. The **!** command is identical to the **shell** command.

Use the *command* option to execute a command on your local host without suspending **telnet**. If you have a network connection and execute the **shell** command with the *command* option, you return to your network connection when the command completes. If you do not have an outstanding network connection, you return to local mode when the command completes.

Example

```
telnet> shell ⮐
$
```

Creates a shell child process.

```
$ exit ⮐
telnet>
```

Entering the shell **exit** command returns you to local mode.

Error Message

`Could not create a shell`	**telnet** could not create a shell child process.

status

Print out the status of TELNET parameters.

Format

status

Description

Use this command when you want to check on **telnet** local mode parameters. This command will also display the state of TELNET parameters for any outstanding network connection.

Example

```
telnet> status ↵
Connected to localhost.
Escape sequence: ^]
Crmod:off     Debug: off     Options: off
Binary input:  on            Binary output: on
Line mode:     off           Local echoing: off
Output logfile: output_logfile

List of terminators:  04
Current state of negotiated option:
Will Binary
Do Binary
Do Echo
Do Suppress go ahead
NVT keyboard keys:
IP:  ^C
AO:
AYT:
EC:  ^?
EL:  ^U
EOR:
```

terminator

Add the following string to the list of terminators.

Format

terminator "*string*"

Description

Terminators determine when to ship characters in line mode. When **telnet** sees a terminator, it sends the input buffer to the remote host. Terminators will include the default characters (see below) and any other characters you specify. You must enclose terminators in double quotation marks (" "). The escape character cannot be a terminator. The default terminators are as follows:

- New Line
- Interrupt character
- End-of-file character
- Switch key (if defined)
- End-of-line character (if defined)

Each character in the *string* you specify will be a terminator.

Example

```
telnet> terminator "abc" ↲
```

Here, the letters **a**, **b**, and **c** are each mapped as a terminator.

unterm

Cancel terminator status for specified characters.

Format

unterm *string*

Description

This command lets you clear the terminator meaning associated with any *string* currently defined.

Example

```
telnet> unterm "abc" ↵
```

The letters **a**, **b**, and **c** no longer function as terminators.

z

Suspend telnet.

Format

```
z
```

Description

This command only works if you are using the DG/UX C shell. (See **csh**(1) for details.) The **z** command interacts with the C shell's job control facilities. First, you are placed in the C shell. **telnet** determines that you are using the C shell by looking at the SHELL environment variable which will be **/bin/csh** for the C shell. For more information on the SHELL environment variable, see the **environ**(5) manual page in the *System Manager's Reference for the DG/UX™ System*.

To return to the TELNET session from the C shell, type **fg job#**. To display the job number from the C shell, type **jobs**.

Example

```
telnet> z ↵

Stopped
%
```

telnet is stopped and you are put back in the C shell.

```
% jobs ↵
[1] + Stopped          telnet sys16
%
```

Displays the job number, status, and remote host of your suspended job.

```
% fg 1 ↵
telnet sys16
Returned to the remote host
```

Returns to the TELNET session you specify.

?

List telnet command definitions or define a specific command.

Format

? [*command*]

Description

Use this command to display a list of **telnet** commands with their definitions. For more information on the syntax and meaning of a command, enter ? with the command as an argument. The ? command is identical to the **help** command.

Example

```
telnet> ? ⤶
```

Displays all available **telnet** commands. If you want information on a particular **telnet** command, enter the ? or **help** command with the command as the argument.

```
telnet> ? status ⤶
```

Displays the meaning of the **status** command.

Error Messages

`Syntax error` or `File does not exist`	You did not enter enough characters to make the command recognizable or there is no such command.

!

Create a shell process as a child of telnet.

Format

! [*command*]

Description

Use this command to work in the shell on your local host, without terminating **telnet**. If you have a network connection, use the escape character (^]) to return to **telnet** local mode before executing the ! command. When you terminate the child process, you will return to your network connection. If you do not have a network connection, you will return to **telnet** local mode. Terminate the child process by using the shell **exit** command. The **shell** command is identical to the ! command.

Use the *command* option to execute a command on your local host without suspending **telnet**. If you have a network connection and execute the ! command with the *command* option, you return to your network connection when the command completes. If you do not have an outstanding network connection, you return to local mode when the command completes.

Example

```
telnet> ! ↵
$
```

Creates a shell child process.

```
$ exit ↵
telnet>
```

Entering the shell **exit** command returns you to local mode.

Error Message

`Could not create a shell`	**telnet** could not create a child process.

End of Chapter

Chapter 3
Understanding and Using the File Transfer Protocol

The File Transfer Protocol (FTP) allows you to transfer files across a TCP/IP network. It does this by establishing a Transmission Control Protocol (TCP) connection between your local host and a remote host. FTP uses this network connection to transmit data and control information.

The FTP implementation has two parts: the FTP user program (**ftp**) and the FTP server (daemon) program (**ftpd**). The *local* host is the system to which your terminal is connected. The *remote* host is the system that you want to transfer files to or from. The FTP user program on the local host and the FTP server program on the remote host are used to connect the two computer systems. Two types of connections link the two systems: a *command connection* and a *data connection*. The command connection transfers commands that describe the functions to be performed and the replies to these commands. The data connection transfers only files.

The FTP server process listens for a command connection from another machine, while the FTP user process initiates a command connection. Figure 3-1 is a diagram of an FTP service connection.

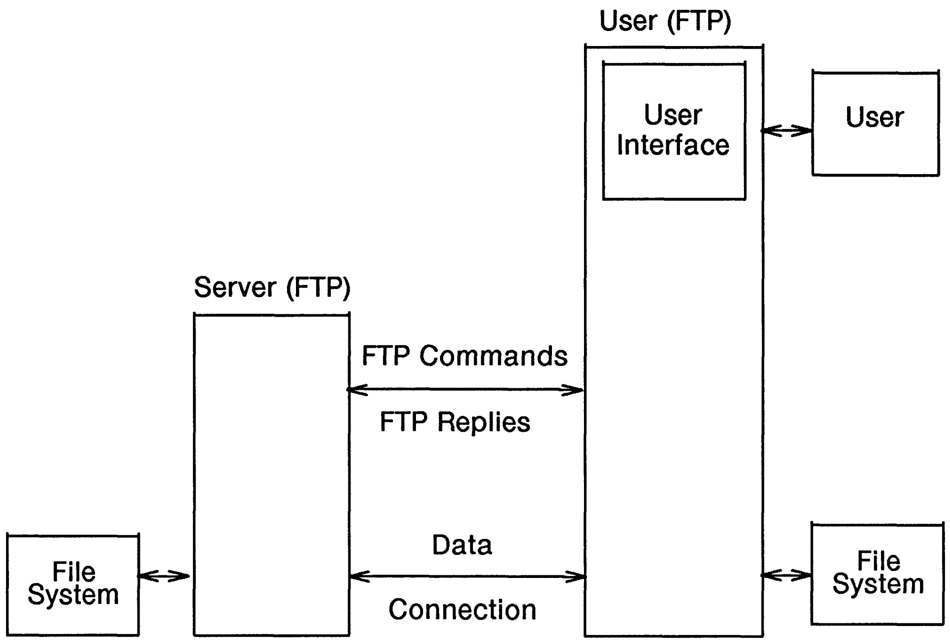

Figure 3-1 *An FTP Service Connection*

When you enter the **ftp** command, you invoke the FTP user program. (FTP in uppercase letters refers to the protocol, while **ftp** in lowercase bold letters refers to the command.) The FTP user program is your interface with the FTP server on the remote host. The user program works with the server program to transfer files. The FTP user program initiates a command connection to the **inetd** server, which in turn invokes the FTP server process **ftpd**.

The user program also issues FTP commands. It also listens on a data port for a connection from an FTP server, sets up parameters for transfer and storage, and transfers data on command.

The FTP server, **ftpd**, is initiated when the **inetd** server invokes it. The **inetd** server runs constantly, listening for a connection from a user program and invoking a server program when needed.

Once invoked, the **ftpd** server maintains a command communication connection. It receives standard ftp commands from and sends replies to the user program. The server program also establishes the data connection with the listening data port, sets up parameters for transfer and storage, and transfers data on command. It can also listen passively on a given data port.

What Is an FTP Command Connection?

An FTP command connection consists of a connection between an FTP local environment and an FTP remote environment. The local and remote environments consist of their respective operating systems and file structures.

You are in the FTP local environment once you execute **ftp** on the local host. Once in the local environment, you enter **ftp** commands to open a command connection with the remote **ftpd**. Then, the remote **ftpd** will ask you for your username, password (if required), and account (if required). If your username and password pair has user privileges on the remote host, your FTP command connection is completed and you have access to the FTP remote environment. You can now enter **ftp** commands to manipulate files back and forth across the FTP connection.

Figure 3-2 shows an FTP connection with a local and a remote FTP environment.

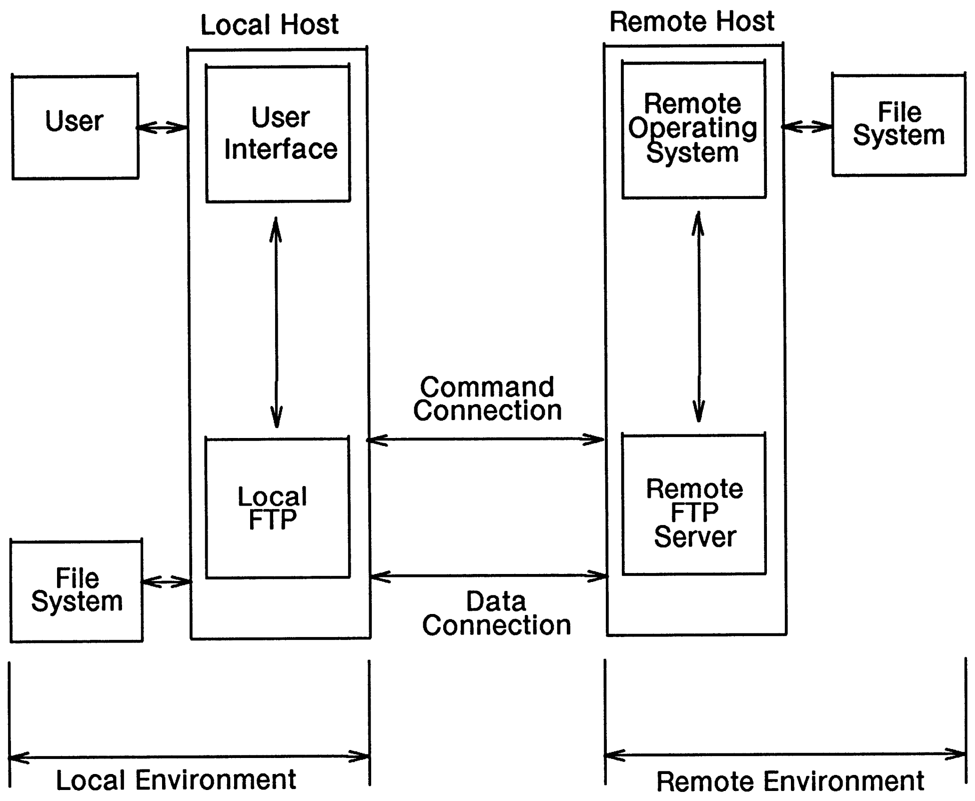

Figure 3-2 *An FTP Connection*

Forming an FTP Command Connection

You form an FTP connection by creating the FTP local environment on the local host and entering **ftp** commands to form a connection with the FTP remote environment.

Creating the FTP Local Environment

To create the FTP local environment, execute the FTP user program. For example, enter the following command:

```
$ ftp ↵
```

The **ftp** program returns the version of **ftp**, the date, and the **ftp** prompt to your terminal. For example:

```
FTP user (DG/UX TCP/IP Release 4.10 07/11/89) ready.
ftp>
```

You are now in your FTP local environment.

Forming a Command Connection with a Remote Host

Once in your FTP local environment, you can form a command connection with a remote host. To do this, enter the **open** command with the name of the remote host as an argument. Optionally, you can specify a port number. The default port number for FTP can be found in the **/etc/services** file.

In the following example, a network connection with a remote host named **remote3** is formed:

```
ftp> open remote3 ↵
Connected to remote3.
220 remote3 FTP server (DG/UX TCP/IP Release 4.10 07/11/89) ready.
Name (remote3:username):
```

If you have verbose mode on, the FTP server sends a reply that you can see at your terminal for every command entered (see **verbose** in the "ftp Command Dictionary later in this chapter"). Each response is preceded by a three-digit number.

You can form your local environment and establish the remote connection in one step. Simply execute **ftp** and enter the remote host's name as an argument on the same command line. By doing this, you do not stop in the FTP local environment before establishing the connection.

For example, the following single command creates the FTP environment and establishes a connection with remote host **remote3**.

```
$ ftp remote3 ↵
FTP user (DG/UX TCP/IP Release 4.10 07/11/89) ready.
Connected to remote3.
220 remote3 FTP server (DG/UX TCP/IP Release 4.10 07/11/89) ready.
Name (remote3:username):
```

To protect against unauthorized access to files, all users need an authorized username/password pair to access files in the remote environment. **ftp** may automatically prompt you for your username and password. If **ftp** does not prompt you, you must enter the **user** command.

If **ftp** automatically prompts you for your username and password, it might look like this:

```
Name (remote3:you):you ↵
331 Password required for you.
Password (remote3:you): type your password ↵
230 User you logged in.  No account needed.
ftp>
```

You have now created an FTP connection between your FTP local and remote environments.

NOTE: Your password does not appear on your terminal. This is so any unauthorized persons watching you work will not learn your username and password and then be able to use your FTP remote environment.

By default, FTP uses *auto-login* when forming a connection. Auto-login tries to log you on to a remote system automatically when you make the initial FTP command connection. To use auto-login, you should have a **.netrc** file in your home directory for often used machines. This file contains an entry for each remote machine you want to connect to and includes your username, password, and account (if one is required).

A typical **.netrc** file would have the following format:

```
machine remote2 login you password noprob
machine remote3 login you password noprob
machine remote5 login you password noprob
machine remote8 login you password noprob
machine remote9 login you password noprob
```

ftp checks in the **.netrc** file for a login entry. If **ftp** finds no entry for the machine you are connecting to, it prompts you for the login name, password, and account number to log you in to the remote system.

Because the **.netrc** file contains your username and passwords, **ftp** requires you to restrict access to this file. To restrict access to yourself only, set your access mode to 600.

For example, if you have a **.netrc** file with an entry for the remote system **remote3**, you can use auto-login by typing the following:

```
$ ftp remote3 ↵
FTP user (DG/UX TCP/IP Release 4.10 07/11/89) ready.
Connected to remote3.
220 remote3 FTP server (DG/UX TCP/IP Release 4.10 07/11/89) ready.
331 Password required for you.
230 User you logged in. No account needed.
ftp>
```

ftp connects to **remote3** and logs you in.

Understanding Error Messages

When you use **ftp**, you may see error messages generated from the FTP user and server programs, the DG/UX operating system, and lower layer protocols, such as TCP/IP. Error messages generated by the FTP user and server programs are documented in this manual.

This section contains error messages you could see when invoking **ftp**. It also includes two general error messages that you may get while using any of the FTP user commands. For information on error messages that are specific to FTP user commands, see the command dictionary at the end of this chapter. For error messages from the FTP server, see Appendix A.

Table 3-1 lists and explains errors that can occur when you invoke **ftp**.

Table 3-1 ftp Error Messages

Error	What It Means
`ftp: ftp/tcp: unknown service`	The protocol name (TCP) was not found in **/etc/services** file.
`ftp:` *option-name*`: unknown option`	FTP user program does not recognize the option you provided when invoking **ftp**.
`Unable to read from input`	End of File (EOF) encountered 10 times while reading from standard input.
value`: bad debugging value`	You typed in a negative number with the debug option.
hostname`: bad port number`	The default port number found in **/etc/services** is incorrect or you invoked **ftp** with an incorrect port number argument.
`Error – .netrc file not correct mode`	Change mode of **.netrc** file to 600.
`Unknown .netrc option` *opt-name*	The **.netrc** file should contain machine, username, password, and account fields only.

When issuing any of the **ftp** commands, you could see one of the following errors: `?Ambiguous command` or `?Invalid command.` `?Ambiguous command` means that you did not type enough of the command's name for the **ftp** program to identify it. `?Invalid command` indicates that the **ftp** either does not recognize or does not support the command you issued.

Terminating an FTP Connection

There are two ways of terminating an FTP connection: 1) you can terminate the connection and remain in your local FTP environment; or 2) you can exit from both your FTP remote and local environments, terminate the network connection, and return to the shell. Table 3-2 explains your two options for terminating an FTP network connection.

Table 3-2 Terminating an FTP Network Connection

Command	What It Does
close or **disconnect**	Terminates the network connection. You remain in your FTP local environment. You can now create another connection by using the **open** command.
bye, exit, or **quit**	Terminates the connection, and exits from your FTP local environment. You return to the shell. To form another connection, you must execute **ftp** to create your FTP local environment.

Understanding ftp File-Naming Conventions

Files specified as arguments to **ftp** commands are processed according to the following rules:

1. If a pathname begins with a slash (/), local filenames are interpreted from the absolute pathname. Otherwise, local filenames are interpreted from the working directory. For more information, see the **intro**(2) manual page in the *Programmer's Reference for the DG/UX™ System (Volume 1).*

2. If you use the dash (–) instead of a filename, the FTP user program uses standard input for reading and standard output for writing. For example, if you enter **get test1 -**, the FTP user program will send the contents of **test1** to your screen.

3. If the first character of the filename is a vertical line (|), the rest of the argument is interpreted as a shell command. The FTP user program then forks a shell, using **popen**(3) with the argument supplied, and reads from the standard input or writes to the standard output. If the shell command includes spaces, the argument must be enquoted; for example, "**| ls –lt**".

4. If the above tests fail and globbing is enabled, local filenames are expanded according to the rules used in the **csh** (1). Globbing is the method of processing local filenames for metacharacters. See the description of the **glob** command below for details.

Understanding FTP Transfer Parameters

Because different machines often use different ways to store data, FTP lets you transform data as well as transfer it. Several parameters control data transmission and its representation during transfer. These transfer parameters are mode, structure, and type. Mode defines how the data bits are transferred. Structure and type define how the data is represented as it is transferred. Table 3-3 lists the transfer parameters, the **ftp** commands to change them, and the options available for each parameter.

Table 3-3 FTP Transfer Parameters

Parameter	Command	Options
mode	**mode**	FTP supports **stream, block**, and **compress** modes. In **stream** mode (the default), data is transmitted across the connection as a stream of data bytes; there are no restrictions on the representation type. FTP does no processing, such as data compression or failure recovery, in this mode. In **block** mode, data is transmitted as a series of data blocks preceded by headers. Record structures are allowed in this mode and any representation type can be used. Also, restart procedures are supported. In **compress** mode, you can send regular data, which is sent in a byte string, compressed data, which consists of replications or filler, and control information, which is sent in a two-byte sequence. Also, restart procedures are supported.
structure	**struct**	FTP supports **file**, **page**, and **record** structures. In the **file** structure (the default), the file is considered to be a continuous sequence of data bytes without any internal structure. In the **page** structure, the file is made up of independent indexed pages. This structure is accepted only with the Local Byte type and is supported in the **stream** mode only. In **record** mode, the file is made up of sequential records. The DG/UX operating system does not support Record structured files for storage. By default, all EOR delimiters are replaced by a new line character.

(continued)

Table 3-3 FTP Transfer Parameters

Parameter	Command	Options
type	**type**	FTP supports four transfer types: **ascii, image** (**binary** is identical to **image**), **ebcdic**, and **local-byte**. The **ascii** transfer type is the default type and is accepted by all FTP implementations. It is used primarily to transfer text files. The **image** transfer type is used to transfer files in contiguous bits that are packed into eight-bit transfer bytes. The receiving site must store the data as contiguous bits. This type is used to transfer binary data. The **ebcdic** transfer type is used to transfer data between hosts that use ebcdic for their internal representation. It is used for text files. Data are represented as eight-bit ebcdic characters. The **local-byte** transfer type is used to transfer text files in **page** structure. Data are transferred in logical bytes specified by the parameter *byte size*.

(concluded)

If you want a transferred file to be identical to the original file, make sure that you retrieve the file with the same parameters that it was stored with.

NOTE: Transfer parameters must be the same on both the local and remote hosts. You can check transfer parameters with the **status** command (see "ftp Command Dictionary" later in this chapter).

Restrictions on File Transfer Parameters

The **ftp** command imposes the following restrictions on the file transfer parameters:

- The **local-byte** size must be a multiple of eight bits.
- The **page** structure requires **local-byte** type.
- The **page** structure is supported in the **stream** mode only.

For more information about the **ftp** commands that change the transfer parameters, see "ftp Command Dictionary" later in this chapter.

Displaying Information About FTP

The **help** and **status** commands display information about **ftp** commands and parameters both in your FTP local environment and your FTP remote environment. For information on remote commands, use **remotehelp**.

Using the ftp Help Command

While in your FTP local environment, you can check on the command line syntax or meaning of an **ftp** command. For a list of all available commands, enter either the **ftp help** or **?** command. The **?** command is identical to the **help** command. For example, enter

```
ftp> help ↵
```

to display all **ftp** commands.

If you want information about a particular command, enter the **help** or **?** command with the command in question as an argument. For example, enter

```
ftp> help append ↵
```

to display the meaning of the **append** command.

Using the ftp status Command

The **status** command displays information about your FTP connection. For example

```
ftp> status ↵
```

will display the following information about your connection:

With regard to the FTP user program, the **status** command will show the following:

- FTP Connection status
- **struct** parameter value
- **mode** parameter value
- **type** parameter value
- **verbose** mode status
- **bell** mode status

- **prompt** mode status
- **glob** status
- **hash** printing status
- **port** protocol command status
- **receive unique** status
- **store unique** status

With regard to the FTP server program, the **status** command will show the following:

- FTP Connection status
- **struct** parameter value
- **mode** parameter value
- **type** parameter value
- EOR delimiter
- Page size

NOTE: Server information depends on the FTP server, and may vary among various vendors' implementations of FTP.

Retrieving a File from a Remote System

Once you are connected to another system, you can retrieve a file from the remote system. To retrieve a remote file, use the **get** or the **recv** command. These commands are identical. For example, while in remote mode, you can retrieve a **file** by typing:

```
ftp> get file ↵
```

This retrieves the remote **file** and stores it in your current local working directory with the filename **file**.

NOTE: If a file already exists in your current local working directory with the same name as the file being retrieved, the existing file might be overwritten. See the "ftp Command Dictionary" later in the chapter for more information on the **get** and **recv** commands.

Taking Precautions

When you are transferring files, the DG/UX system file access permissions apply. For information on file access permissions in the DG/UX system, see "File Access Permissions" in **intro**(2) of the *System Manager's Reference for the DG/UX™ System*. Also, see the commands **chmod**(1), **chown**(1), and **chgrp**(1) in the *User's Reference for the DG/UX™ System*.

As a user of a remote system, you have to learn the remote system's file structure. It is especially important to learn the file-naming rules of the remote file system. Some remote operating systems limit their file names to 14 characters. DG/UX allows 255 characters. An example of a potential problem is as follows:

Suppose you have a DG/UX system directory containing ten files with filenames of the format **SYSTEM_PROBLEMS.identifier**. Also suppose that you try to move the entire directory of files to a remote system that allows filenames of only 14 characters, so that you end up with one file named **SYSTEM_PROBLEM**.

The reason you ended up with one file is that each time you move a file that has the same name as an existing file on the remote host, the new file overwrites the previous file. Any version of FTP that is compatible with Berkeley's 4.2 BSD release of the UNIX® operating system deletes a file if it already exists at its destination. In our example, the remote FTP will first create a file named **SYSTEM_PROBLEM**, using the first 14 characters of the first local filename. Since each of the files you are transferring begins with the same 14 characters, the system will overwrite the previous file each time a file is transferred. No files transferred, except for the last one, will exist on the remote system. The file transaction will be useless.

To avoid the problem, rename each file as you transfer it. Use a target filename that is suitable for the operating system to which you transfer the file. For more information see **get** in the "ftp Command Dictionary," which appears later in this chapter.

Using ftp Commands

You enter **ftp** commands from your FTP local environment. Some of these commands, such as the **remotehelp** command, display information about your FTP remote environment. The responses you receive will depend on your remote operating system's file structure and its implementation of the FTP server.

NOTE: The FTP user program is case-sensitive. You must type commands in lowercase. **ftp** will not recognize commands that are in uppercase.

Table 3-4 lists the **ftp** commands by their functions.

Table 3-4 ftp Commands Listed by Function

Function	Commands	Description
Setting or displaying transfer parameters	**bell**	Toggle beeping after completing some commands.
	glob	Toggle expanding local filenames using metacharacters.
	hash	Toggle printing "#" for each buffer transferred.
	mode	Set file transfer mode.
	prompt	Toggle prompting for multiple file transfers.
	quote	Send commands verbatim to remote server.
	runique	Toggle unique naming of files transferred from other systems.
	sendport	Toggle **port** command.
	status	Show current status of FTP (both sides of the connection).
	struct	Set file transfer structure.
	sunique	Toggle unique naming of files transferred to other systems.
	type	Set the character transfer type.
Manipulating directories	**cd**	Change remote directory.
	cdup	Change the working directory on the remote machine to the parent directory.
	dir	List the contents of the current remote directory.
	mdir	Display multiple remote directories.
	mls	Display abbreviated directory listing of multiple remote files.
	lcd	Change local directory.
	ls	List contents of a remote directory.
	mkdir	Create remote directory.
	pwd	Display name of current remote directory.
	rmdir	Delete remote directory.

(continued)

Table 3-4 ftp Commands Listed by Function

Function	Commands	Description
Manipulating files	**append**	Append to a remote file.
	delete	Delete a remote file.
	get - recv	Receive a remote file.
	mdelete	Delete multiple remote files.
	mget	Receive multiple remote files.
	mput	Send multiple files to remote system.
	put - send	Send a remote file.
	rename	Rename a remote file.
Opening or closing a local or remote FTP connection	**abort**	Abort current transfer.
	account	Send account number.
	bye - quit	Terminate FTP session.
	exit	Terminate FTP session.
	open	Connect to remote host.
	close	Terminate connection with a remote host.
	disconnect	A synonym for **close**.
	reinit	Reinitialize command connection.
	restart	Restart last aborted transfer.
	user	Send remote user information.
	!	Create a shell process on the local host.
Displaying information		
	debug	Toggle debug mode.
	help - ?	Display information on commands.
	remotehelp	Help from FTP server.
	site	Display remote information.
	verbose	Toggle verbose mode.

(concluded)

ftp allows you to execute a group of commands while a data transfer is in progress. To use these commands, you must enter the interrupt character first. The interrupt character is usually Ctrl-C; however, it can be different from system to system. Use the DG/UX command **stty**(1) to find out what your interrupt character is. The interrupt character suspends the data transfer and displays a menu on the screen. The menu lists the available commands. Table 3-5 lists and explains the available commands.

Table 3-5 Commands Available During FTP Data Transfer

Command	Function
interrupt character	Terminates the FTP-user process.
abort	Aborts data transfer, closes data connection, but leaves command connection open.
quit	Completes data transfer, closes data connection, terminates user, and closes command connection.
reinit	Completes data transfer, terminates user, resets all transfer parameters, but leaves the command connection open.
status	Displays status information and continues data transfer.
continue	Continues the data transfer.
help [*command*]	Displays available commands or information about a specified command.

Using ftp Command Line Options

Some of the user's commands are available as command line options, which allow you to issue commands when you invoke **ftp** without having to enter local mode first. Table 3-6 shows the available options, their corresponding commands, and definitions.

Table 3-6 ftp Command Line Options

Option	Command	Definition
–i	**prompt**	Turns off interactive prompting during multiple file transfers.
–d	**debug mode**	Turns debug mode on.
–v	**verbose**	Turns verbose mode on.
–n	**no-autologin**	Prevents **ftp** from attempting auto-login upon initial connection.
–g	**glob**	Turns filename globbing off.

For example, if you want to open a connection with **remote4** without using auto-login, you can type the following when you invoke **ftp**:

```
$ ftp –n remote4 ↵
```

ftp opens a connection with **remote4** and produces an **ftp** prompt. You must use the **user** command to start the login process.

ftp Command Dictionary

The following is an alphabetical listing of all **ftp** user commands. Along with a format line is a description of the command and an example of how to use the command. Most commands that require an argument will prompt you for the argument if you enter the command without it.

Responses to user commands depend on the remote server implementation. Examples in this document provide responses from the DG/UX TCP/IP implementation of the FTP server. Responses from other implementations of the FTP server may be different.

abort

Abort the previous ftp command.

Format

abort

Description

If no data transfer is in progress and if a data connection is open, **abort** closes the data connection and sends a reply indicating that the **abort** command was successfully processed.

If a data transfer is in progress, this command aborts the transfer in progress, closes the data connection, and returns a reply indicating that the transfer terminated abnormally. It then returns a reply indicating that the **abort** command was successfully processed.

Example

```
ftp> abort ↵
226 command okay.
ftp>
```

The data connection closes and you remain in command mode.

account

Send account number for a system login or a specific access.

Format

account *account-number*

Description

Use this command as part of the login procedure to a remote host or for access to a specific process, such as storing files.

The DG/UX FTP server program does not require any account information. If a foreign server requires account information during the login procedure, the FTP user program automatically prompts you for your account.

Example

```
ftp> account ↵
(account-number) 123 ↵
202 No account information necessary.
```

Entering the **account** command causes **ftp** to prompt you for your account number.

append

Append a local file to a remote file.

Format

append *local-filename* [*remote-filename*]

Description

Use this command when you want to append a local file to a remote file. The settings for type, mode, and structure remain the same.

If you omit the optional *remote-filename* argument, the remote host will create a new file with the *local-filename*. If a file already exists on the remote host with that name, then the local file will append to that remote file.

Example

```
ftp> append test_jog ⮐
```

The local file **test_jog** is appended to the file of the same name on the remote system. If **test_jog** does not exist on the remote system, the file **test_jog** is created.

bell

Sound a bell after certain commands.

Format

bell

Description

Use this command to turn on or off the sounding of a bell after certain commands. The bell sounds after commands that transfer files or commands that manipulate or list remote directories. By default, the bell is off.

Example

```
ftp> bell ↵
Bell mode on.

ftp>
```

Turns the sounding of a bell on.

```
ftp> bell ↵
Bell mode off.

ftp>
```

Turns the sounding of a bell off.

bye

Terminate the FTP connection and return to the shell.

Format

bye

Description

Use this command to terminate the FTP connection and to return to the shell. The **quit** command is identical to the **bye** command.

Example

```
ftp> bye ↵
220 Goodbye.
$
```

Terminates the FTP connection and returns you to the Bourne or C shell.

cd

Change the working directory on the remote host.

Format

cd *remote-directory*

Description

Use this command to change your working directory on the remote host.

Example

```
ftp> cd errors ↵
200 CWD command okay.

ftp> pwd ↵
251 "/bo.01/udd/errors" is current directory.

ftp>
```

Changes your remote working directory to a directory called **errors**. The *remote-directory* name must be in a form that the remote host can recognize. (See your remote host's operating system manuals to see how the remote system recognizes directory names.)

cdup

Change working directory on the remote host to the parent directory.

Format

cdup

Description

Use this command to change you working directory on the remote host to the parent directory.

Example

```
ftp> pwd ↵
251 "/usr/hank/doc" is current directory.

ftp> cdup ↵
200 command okay.

ftp> pwd ↵
251 "/usr/hank" is current directory.

ftp>
```

Changes your remote working directory from **/usr/hank/doc** to the parent directory **/usr/hank**.

close

Close the FTP connection.

Format

close

Description

Use this command to close the FTP connection, but remain in your FTP local environment. This command is identical to **disconnect**.

Example

```
ftp> close ↵
221 Goodbye.

ftp>
```

Closes the FTP connection. You remain in your FTP local environment.

debug

Turn debug mode on or off.

Format

debug [*debug-value*]

Description

This command toggles debug mode. Use this command if you want to see the commands that are sent over to the server. The option *debug-value* specifies the status of debugging. Use 1 to turn debug mode on and 0 to turn it off.

CAUTION: We cannot guarantee that all commands will function normally when debug mode is on. Expect some unusual results when operating in this mode.

Example

```
ftp> debug ↵
Debugging on (debug=1).

ftp>
```

Turns debug mode on.

```
ftp> debug ↵
Debugging off (debug=0).

ftp>
```

Turns debug mode off.

Error Message

value: `bad debugging value`	You typed in a negative number with the debug option.

delete

Delete a remote file.

Format

delete *remote-filename*

Description

Use this command to delete a file on the remote host.

Example

```
ftp> delete test ↵
200 DELE command okay.

ftp>
```

Deletes the remote file named **test**.

dir

List the contents of a remote directory.

Format

dir [*remote-directory* [*local-filename*]]

Description

Use this command to list the contents of a remote directory. If you enter the **dir** command without any arguments, the default is the current remote working directory.

If you specify a *remote-directory*, the contents of that directory are listed.

By using the option *local-filename*, you place a copy of the directory contents in a local file. If you omit the optional *local-filename*, the contents are displayed on your terminal.

We recommend that you use the ASCII transfer type when using this command.

Example

```
ftp> dir ↵
200 Port command okay.
150 Opening data connection for /bin/ls (89.0.0.47,2781)
(0 bytes).

total 1978
-rw-rw-rw-   1 other  993280 Jan 15 14:39 test
-rw-------   1 other    2178 Jan 14 09:02 mbox
-rw-rw-rw-   1 other    8115 Oct 29 16:35 list
226 Transfer complete.
165 bytes received in 1.00 seconds (0.16 Kbytes/s)

ftp>
```

This example shows how the **dir** command displays the contents of the remote working directory on the terminal.

disconnect

Terminate the FTP connection.

Format

disconnect

Description

Use this command to close the FTP connection but remain in your FTP local environment. This command is identical to **close**.

Example

```
ftp> disconnect ↵
221 Goodbye

ftp>
```

Closes the FTP connection. You remain in your FTP local environment.

exit

Abruptly terminate the FTP connection and return to shell.

Format

exit

Description

Use this command to terminate the FTP connection quickly and return to the shell. The **quit** and **bye** commands are similar to the **exit** command. The **exit** command, however, works a bit faster.

Example

```
ftp> exit↵
$
```

Terminates the FTP connection and returns you to the Bourne or C shell.

get

Retrieve a remote file and store it on a local system.

Format

get *remote-filename* [*local-filename*]

Description

Use this command to retrieve a remote file and store it on your local machine.

If you do not specify the *local-filename* option, **ftp** gives the file the same name it has on the remote host, within the limitations of the two file systems.

If there already is a local file with the same name as the *remote-filename* and you do not have **runique** set to true, you must create a new name for the *local-filename*. Otherwise, **ftp** will overwrite the local file.

If you specify a dash (-) for the argument *local-filename*, the **get** command copies the file to standard output.

The transfer uses the current settings for type, mode, and structure. This command is identical to the **recv** command.

Example

```
ftp> get test ↵
200 PORT command okay.
150 Opening data connection for test (127.0.0.1,1028) 8 bytes
226 Transfer complete.
Received 8 bytes.

ftp>
```

This example shows how the **get** command retrieves the remote file **test** and stores it in the current working directory with the filename of **test**.

Error Messages

`Invalid combination of transfer parameters`	The combination of transfer parameters (mode, type, and structure) you used is illegal.
filename`: not a plain file`	You tried to transfer a directory or device file.
`ftp: fdopen failed`	**fdopen** call for socket failed on the command connection end.

The error `ftp_recv_data with error: ftp-error` indicates that the data received contained one of the following errors:

(ftp-error	What It Means
`Error on read`	**ftp** detected an error while reading from the file or socket.
`Error on write`	**ftp** detected an error while writing to the file or socket.
`Unexpected EOF`	**ftp** found a premature End-of-File delimiter while reading from the socket.
`Wrong escape sequence`	**ftp** found an incorrect escape sequence while receiving a file in Record structure and Stream mode.
`Page is not multiple of logical byte`	The page size defined for Page structure is not a multiple of logical byte size.

glob

Turn filename globbing on or off.

Format

glob

Description

Use this command when you want to turn globbing on or off. When globbing is enabled, each local filename or pathname is processed for **csh**(1) metacharacters. These characters include * ? [ ]˜ { }. When globbing is disabled, all filenames and pathnames are treated literally.

NOTE: Only **mget** and **mput** allow globbing.

Example

```
ftp> glob ⤶
Globbing off (on).
```

Turns filename globbing off or on.

Error Messages

`Unknown username after˜`	The username specified after ˜ does not exist.
`Bad directory component`	The **open** or **opendir** call failed.
`Arguments too long`	Template expansion is longer than 340 characters.
`Pathname too long`	Directory pathname longer than 2048 characters.

hash

Turn hash sign (#) printing on or off.

Format

hash

Description

Use this command to turn on or off the printing of a hash sign (#) for each block of data transferred. The size of a data block is 2048 bytes. The hash signs are printed on your terminal as the data transfer is taking place. You use this command to receive some indication that a file transfer is actually going on; this is especially useful if you are transferring large files.

By default, hash sign printing is off. To turn on the printing of hash signs, enter the **hash** command.

Example

```
ftp> hash ⤶
Hash mark printing on (2048 bytes/hash mark).

ftp>
```

Turns hash printing on.

```
ftp> hash ⤶
Hash mark printing off.

ftp>
```

Turns hash printing off.

help

List ftp commands or information on a specific command.

Format

help [*command*]

Description

When no data transfer is in progress, use this command when you need information on a *command*. To display a list of all ftp commands, enter the **help** command without any arguments. To display the description of a particular *command*, enter the **help** command with the command as an argument. The **?** command is identical to the **help** command.

When data transfer is in progress, use this command when you need information on commands available when a data transfer has been interrupted. To display a list of all the available commands, enter the **help** command without any arguments. To display the description of a particular *command*, enter the **help** command with the *command* as an argument.

Example

```
ftp> help ↵
```

Displays all ftp commands.

```
ftp> help ! ↵
```

Displays the command description for the **!** command.

lcd

Change the working directory on the local host.

Format

lcd [*directory-name*]

Description

Use this command to change the working directory on the local host. If you omit *directory-name*, **ftp** will go to the user's home directory.

Example

```
ftp> lcd ↵
Local directory now /usr/udd/op

ftp>
```

Changes the local working directory to the user's home directory.

```
ftp> lcd .. ↵
Local directory now /usr/udd

ftp>
```

Changes the local working directory to the directory above your present working directory.

ls

Print an abbreviated listing of a remote directory's contents.

Format

ls [*remote-directory* [*local-filename*]]

Description

Use this command to list the contents of a *remote-directory*. This listing will contain only the filenames.

If you do not specify a *remote-directory*, **ftp** displays the filenames of the current remote working directory.

If you specify the option *local-filename*, the output will be placed in a local file with that name.

If you do not specify *local-filename*, **ftp** displays the output on your terminal.

Example

```
ftp> ls ↵
200 PORT command okay.
150 Opening data connection for /bin/ls (89.0.0.47,2795)
core
mbox
temp
226 Transfer complete.
18 bytes received in 0.00 seconds (17.57 Kbytes/s)

ftp>
```

Lists the contents of the current remote working directory on your terminal.

mdelete

Delete files on the remote machine.

Format

mdelete *remote-filenames*
mdelete *remote-filename-template*

Description

Use this command to delete several files from the working directory on the remote machine.

If you do not turn prompting mode off, **ftp** will prompt you with each file that matches the template. (See the **prompt** command later in this chapter.)

Example

```
ftp> mdelete test1 test2 test3↵

mdelete test1? y ↵
200 DELE command okay.
mdelete test2? y ↵
200 DELE command okay.
mdelete test3? y ↵
200 DELE command okay.
ftp>
```

Mdelete asks for confirmation on each file specified before deleting it.

```
ftp > mdelete test* ↵

mdelete test1? y ↵
200 DELE command okay.
mdelete test2? y ↵
200 DELE command okay.
mdelete test3? y ↵
200 DELE command okay.
ftp>
```

The **mdelete** command asks for confirmation on each file matching the template one by one before deleting it.

Error Message

`Can't find list of remote files`	File **/tmp/ftp***xxx* cannot be found.

mdir

Obtain a directory listing of multiple files.

Format

mdir *remote-filenames*
mdir *remote-filename-template local-file*

Description

Use this command to get a directory listing of several files in the working directory on the remote machine and place them in a *local-file*.

You can use filename templates as an argument to **mdir**. The **mdir** displays a directory listing of all the filenames matching the filename template. However, **mdir** does not display a listing for sub-directory names that match the template.

If you do not turn prompting mode off, **ftp** will prompt you with the name of each file that matches the template (see the **prompt** command later in this chapter.)

Example

```
ftp> mdir test1 test2 home↵

mdir test1? y ↵
200 PORT command okay.
150 Opening data connection for /bin/ls (0 bytes).

226 Transfer complete.

mdir test2? y ↵
200 PORT command okay.
150 Opening data connection for /bin/ls (0 bytes).

226 Transfer complete.
```

The **mdir** command asks confirmation on each directory specified before retrieving a directory listing and placing it in the file **home** on the local machine.

```
ftp> mdir test* home1↵

mdir test1? y ↵
200 PORT command okay.
150 Opening data connection for /bin/ls (0 bytes).

226 Transfer complete.

mdir test2? y ↵
200 PORT command okay.
150 Opening data connection for /bin/ls (0 bytes).

226 Transfer complete.
72 bytes received in 1.00 seconds (7.03e-02 Kbytes/s)
```

The **mdir** command asks confirmation on each filename matching the filename template before retrieving a directory listing and placing it in the file **home1** on the local machine.

CAUTION: When filenames contain meta-characters, this command can sometimes crash the remote FTP daemon.

Error Messages

`Can't find list of remote files`	File **/tmp/ftp***xxx* cannot be found.
`Invalid combination of transfer parameters`	The combination of transfer parameters (mode, type, and structure) you used is illegal.
filename`: not a plain file`	You tried to transfer a directory or device file.
`ftp: fdopen failed`	**fdopen** call for socket failed on the command connection end.

The error `ftp_recv_data with error: ftp-error` indicates that the data received contained one of the following errors:

ftp-error	**What It Means**
`Error on read`	**ftp** detected an error while reading from the file or socket.
`Error on write`	**ftp** detected an error while writing to the file or socket.
`Unexpected EOF`	**ftp** found a premature End-of-File delimiter while reading from the socket.
`Wrong escape sequence`	**ftp** found an incorrect escape sequence while receiving a file in Record structure and Stream mode.
`Page is not multiple of logical byte`	The page size defined for Page structure is not a multiple of logical byte size.

mget

Retrieve remote files and store them in a local directory.

Format

mget *remote-filenames*
mget *remote-filename-template*

Description

Use this command to retrieve remote files and store them in your local working directory. If you don't specify any filenames, **mget** prompts you for them.

The option remote-filename-template may contain a filename template character. The **mget** command will work on all files that match remote-filename-template.

The names of the retrieved local files are the same as the remote files, within the limitations of the two file systems. If there already is a local file with the same name as one of the files matching the template and you do not have **runique** set to true, **ftp** (if the user has write access) will overwrite the local file. If you do not have write access, you will receive an error indicating that permission to transfer the file has been denied.

The settings for type, mode, and structure remain the same.

If you do not turn prompting mode off, **ftp** will prompt you with each file that matches the template. (See the **prompt** command later in this chapter for details.)

Example

```
ftp> mget file1 file2 ↵

mget file1? y ↵
200 PORT command okay.
150 Opening data connection for file1 (89.0.047,1185)
(39778 bytes).

226 Transfer complete.
40843 bytes received in 3.00 seconds (13.29 Kbytes/s)
mget file2 y ↵
200 PORT command okay.
150 Opening data connection for file2 (89.0.047,1186)
(19724 bytes).
```

```
226 Transfer complete.
20450 bytes received in 1.00 seconds (19.97 Kbytes/s)

ftp>
```

Asks for confirmation on **file1** and **file2** before retrieving and storing them on the local file.

```
ftp> mget test* ↵
mget test1? y ↵
200 PORT command okay.
150 Opening data connection for test1 (89.0.0.47,1158)
(28412 bytes)

226 Transfer complete.
29356 bytes received in 2.00 seconds (14.33 Kbytes/s)
mget test2? y ↵
200 PORT command okay.
150 Opening data connection for test2 (89.0.0.47,1161)
(117160 bytes)

226 Transfer complete.
121168 bytes received in 10.00 seconds (11.83 Kbytes/s)

ftp>
```

Retrieves all remote files that start with the first four characters **test**, and gives them the same names in the local environment. In this example, only two remote files matched the four characters and template, so the only files transferred were **test1** and **test2**.

CAUTION: Be sure that the files you want to retrieve actually exist before using mget. If you use filename templates, be sure that no files have **csh**(1) metacharacters in their names.

Error Messages

`Can't find list of remote files`	File **/tmp/ftp***xxx* cannot be found.
`Invalid combination of transfer parameters`	The combination of transfer parameters (mode, type, and structure) you used is illegal.
filename`: not a plain file`	You tried to transfer a directory or device file.
`ftp: fdopen failed`	The **fdopen** call for socket failed on the command connection end.

The error `ftp_recv_data with error: ftp-error` indicates that the data received contained one of the following errors:

ftp-error	**What It Means**
`Error on read`	**ftp** detected an error while reading from the file or socket.
`Error on write`	**ftp** detected an error while writing to the file or socket.
`Unexpected EOF`	**ftp** found a premature End-of-File delimiter while reading from the socket.
`Wrong escape sequence`	**ftp** found an incorrect escape sequence while receiving a file in Record structure and Stream mode.
`Page is not multiple of logical byte`	The page size defined for page structure is not a multiple of logical byte size.

mkdir

Create a directory on the remote host.

Format

mkdir *directory_name*

Description

Use this command to create a directory on the remote host. If you do not specify the argument *directory_name*, **mkdir** prompts you for it. If the directory you specify already exists, **mkdir** tells you. The **mkdir** command will not overwrite an existing directory.

The **mkdir** command works only with a server that implements the **XMKD** command. Use the command **remotehelp** to see if **XMKD** is implemented.

Example

```
ftp> mkdir jgdir ↵
200 mkdir command okay.

ftp>
```

Creates a remote directory named **jgdir** in the current working directory.

mls

Obtain an abbreviated directory listing of multiple files.

Format

mls *remote-filenames*
mls *remote-filename-template local-file*

Description

Use this command to place an abbreviated directory listing of several files on the remote machine into *local-file*. If you do not specify *remote-filenames,* **mls** prompts you for them. If *local-file* does not exist, it is created.

You can use filename templates as an argument to **mls**. The **mls** command displays an abbreviated directory listing of all the filenames matching the filename template. However, **mls** does not display a listing for directory names that match the template.

If you do not turn prompting mode off, **ftp** will prompt you with each file that matches the template. (See the **prompt** command later in this chapter.)

Example

```
ftp> mls test1 test2 home↵

mls test1? y ↵
200 PORT command okay.
150 Opening data connection for /bin/ls (0 bytes).

226 Transfer complete.

mls test2? y ↵
200 PORT command okay.
150 Opening data connection for /bin/ls (0 bytes).

226 Transfer complete.
```

The **mls** command asks confirmation on each filename specified before retrieving a directory listing and placing it in the file **home** on the local machine.

```
ftp> mls test* home1↵

mls test1? y ↵
200 PORT command okay.
150 Opening data connection for /bin/ls (0 bytes).

226 Transfer complete.

mls test2? y ↵
200 PORT command okay.
150 Opening data connection for /bin/ls (0 bytes).

226 Transfer complete.
72 bytes received in 1.00 seconds (7.03e-02 Kbytes/s)
```

The **mls** command asks confirmation on each filename matching the filename template before retrieving a directory listing and placing it in the file **home1** on the local machine.

CAUTION: When filenames contain meta-characters, this command can sometimes crash the remote FTP daemon.

Error Messages

`Can't find list of remote files`	File **/tmp/ftp***xxx* cannot be found.
`Invalid combination of transfer parameters`	The combination of transfer parameters (mode, type, and structure) you used is illegal.
filename`: not a plain file`	You tried to transfer a directory or device file.
`ftp: fdopen failed`	**fdopen** call for socket failed on the command connection end.

The error `ftp_recv_data with error: ftp-error` indicates that the data received contained one of the following errors:

ftp-error	**What it Means**
`Error on read`	**ftp** detected an error while reading from file or socket.
`Error on write`	**ftp** detected an error while writing to the file or socket.
`Unexpected EOF`	**ftp** found a premature End-of-File delimiter while reading from the socket.
`Wrong escape sequence`	**ftp** found an incorrect escape sequence while receiving a file in Record structure and Stream mode.
`Page is not multiple of logical byte`	The page size defined for Page structure is not a multiple of logical byte size.

mode

Set the file transfer mode.

Format

mode [*mode-name*]

Description

Use this command to set the transfer mode to *mode-name*. If you do not specify an argument, **mode** displays the current working mode. The following modes are available:

Mode	Meaning
block	File is transmitted as a series of data blocks preceded by three header bytes. You can use record structures and any representation type in this mode. You can restart file transfers that have been aborted during a network failure or system crash. You can also restart file transfers that you aborted.
compress	Sends regular data, compressed data, and control information. Regular data is sent in a byte string, compressed data is sent in replications or fillers, and control information is sent in a two-byte sequence. This mode lets you restart file transfers that have been aborted during a system crash or network failure. You can restart file transfers that you aborted as well. If the network performance is slow, using this mode can transfer data faster than with other modes.
stream	Transmits data as a stream of bytes without any restrictions on the type used. This is the default mode. In **file** structure, the data connection is closed after each file transfer.

Example

```
ftp> mode block↵
200 Mode B okay.
Using Block mode to transfer files.

ftp>
```

Data will be transferred in **block** mode.

mput

Store multiple local files in a remote working directory.

Format

mput *local-filenames*
mput *local-filename-template*

Description

Use this command to transfer local files and store them in your remote working directory. If you do not specify *local-filenames*, **mput** prompts you for them. If a file does not exist, **mput** gives you an error message and continues.

The **mput** command will work on all files that match the *local-filename-template*.

The remote files are given the same names they had on the local system, within the limitations of the two file systems. If there already is a remote file with the same name as one of the files matching the template and **sunique** is not set to on, **ftp** (if the user has write access) will overwrite the remote file. If you do not have write access, you will receive an error indicating that permission to transfer the file has been denied.

If you don't turn prompting mode off, **ftp** will prompt you with each file that matches the *local-filename-template*. (See the **prompt** command in this chapter.)

Example

```
ftp> mput test1 test2 ↵
mput test1? y ↵
200 PORT command okay.
150 Opening data connection for test1 (127.0.0.1,1040)

226 Transfer complete.
833 bytes sent in 0.00 seconds (813.47 Kbytes/s)
mput test2? y ↵
200 PORT command okay.
150 Opening data connection for test2 (127.0.0.1,1040)

226 Transfer complete.
891 bytes sent in 0.00 seconds (870.11 Kbytes/s)

ftp>
```

This example shows use of **mput** to ask for confirmation on each file before sending it to the remote host. Each file transferred has the same name it had on the local host.

```
ftp> mput test* ↵
mput test1 ? y ↵
200 PORT command okay.
150 Opening data connection for test1 (127.0.0.1,1040)

226 Transfer complete.
833 bytes sent in 0.00 seconds (813.47 Kbytes/s)
mput test2 ? y ↵
200 PORT command okay.
150 Opening data connection for test2 (127.0.0.1,1040)

226 Transfer complete.
891 bytes sent in 0.00 seconds (870.11 Kbytes/s)

ftp>
```

This example shows use of **mput** to store all files that begin with the four characters **test** on the remote host and give them the same names as their local filenames. In the given example, two files on the local host start with the four characters **test**: **test1** and **test2**.

CAUTION: Be sure that the files you want to store on the remote system actually exist before using mput. If you use filename templates, be sure that no files have **csh**(1) metacharacters in their names.

Error Messages

`Can't find list of remote files`	File **/tmp/ftp***xxx* cannot be found.
`Invalid combination of transfer parameters`	The combination of transfer parameters (mode, type, and structure) you used is illegal.
filename`: not a plain file`	You tried to transfer a directory or device file.
`ftp: fdopen failed`	The **fdopen** call for socket failed on the command connection end.

The error `ftp_recv_data with error: ftp-error` indicates that the data received contained one of the following errors:

ftp-error	**What It Means**
`Error on read`	**ftp** detected an error while reading from file or socket.
`Error on write`	**ftp** detected an error while writing to the file or socket.
`Unexpected EOF`	**ftp** found a premature End-of-File delimiter while reading from the socket.
`Wrong escape sequence`	**ftp** found an incorrect escape sequence while receiving a file in Record structure and Stream mode.
`Page is not multiple of logical byte`	The page size defined for Page structure is not a multiple of logical byte size.

open

Establish an FTP connection with a remote host.

Format

open *hostname*

where

hostname is a hostname in your network.

Description

Use this command to establish an FTP connection with a remote host. Once this connection is formed, **ftp** will attempt to automatically log you on to the remote host unless the local **ftp** program disabled the automatic login facility.

If the automatic login facility is disabled, you have to enter the **user** command to begin the login process. (See the **user** command in this chapter for details.)

If the automatic login facility is enabled, you will be prompted for your username and password.

Example

```
ftp> open remote10 ↵
FTP user (DG/UX TCP/IP Release 4.10 07/11/89) ready.
Connected to remote10.
220 rtp47 FTP server (DG/UX TCP/IP Release 4.10 07/11/89) ready.
Name (remote10:jones): jones ↵
331 Password required for jones.
Password (remote10: jones): (Type jones's password) ↵
230 User jones logged in. No account needed.
ftp>
```

Establishes an FTP connection with the remote host named **remote10** and prompts for your username and password.

Error Messages

`Error - .netrc file not correct mode`	Change mode of **.netrc** file to 600.
`Unknown .netrc option` *opt-name*	The **.netrc** file should contain machine, username, password, and account fields only.
`Already connected to` *hostname*`, use close first`	You invoked the **open** command before closing your current connection.
hostname`: bad port number`	The default port number found in **/etc/services** is incorrect or you invoked **open** with an incorrect second argument.

prompt

Turn prompting on or off.

Format

prompt

Description

Use this command to turn **ftp** prompting on or off. The **mget, mdelete,** and **mput** commands are commands that work on more than one file. You might want to use prompting when you use these commands. (See the **mget, mdelete,** and **mput** commands in this chapter.)

The default for prompting is on.

Example

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

Turns prompting off.

```
ftp> prompt ↵
Interactive mode on.
```

Turns prompting on. Now if you enter an **mget** command to retrieve multiple remote files, **ftp** will prompt you with each file that is found to match the remote filename template. (See the **mget** command in this chapter.)

put

Store a local file in your remote working directory.

Format

put *local-filename* [*remote-filename*]

Description

Use this command to store a local file in your remote working directory. **ftp** uses the current settings for type, mode, and structure.

If you omit the option remote-filename, **ftp** will use the same name as the local-filename argument, within the limitations of the two file systems.

If there already is a remote file with the same name as the local-filename and **sunique** is not on, you will have to specify a new remote-filename. Otherwise, **ftp** will overwrite the remote file.

This command is identical to the **send** command.

Example

```
ftp> put test ↵
200 PORT command okay.
150 Opening data connection for test (127.0.0.1,1041)
226 Transfer complete.
Sent 10 bytes.

ftp>
```

Stores local file **test** on the remote host and names it **test.**

Error Messages

`Invalid combination of transfer parameters`	The combination of transfer parameters (mode, type, and structure) you used is illegal.
filename`: not a plain file`	You tried to transfer a directory or device file.
`ftp: fdopen failed`	The **fdopen** call for socket failed on the command connection end.

The error `ftp_send_data with error: ftp-error` indicates that the data sent contained one of the following errors:

ftp-error	**What It Means**
`Error on read`	**ftp** detected an error while reading from the file or socket.
`Error on write`	**ftp** detected an error while writing to the file or socket.
`Unexpected EOF`	**ftp** found a premature End of File delimiter while reading from the socket.
`Wrong escape sequence`	**ftp** found an incorrect escape sequence while sending a file in Record structure and Stream mode.
`Page is not multiple of logical byte`	The page size defined for Page structure is not a multiple of logical byte size.

pwd

Display the current remote working directory.

Format

pwd

Description

Use this command to display the current remote working directory.

Example

```
ftp> pwd ↵
251 "/udd/mth" is current directory.
```

Displays **/udd/mth,** which is the current remote working directory.

quit

Exit from FTP environments and return to the shell.

Format

quit

Description

If no data transfer is in progress, this command terminates any connection and returns to the shell. If you are in FTP local mode without any connection, this command will cause you to exit your FTP local environment and return to the shell. The **bye** command is identical to the **quit** command.

If a data transfer is in progress, this command allows the data transfer to complete. It terminates the user, closes the data and command connections, and returns to the shell.

Example

```
ftp> quit ↵
221 Goodbye.

$
```

Exits from your FTP local environment and returns you to the Bourne or C shell. If there is an FTP connection, **quit** will terminate that connection as well.

quote

Send commands verbatim to the FTP server.

Format

quote *arg1 arg2* ....

Description

Use this command to send FTP server commands verbatim to the remote FTP server. This command is useful when the remote server program provides **ftp** commands for which no user interface exists.

Example

```
ftp> quote xpwd ⮐
251 "/udd/thompson" is current directory.

ftp>
```

This example shows how the **quote** command sends the command **xpwd** to the remote FTP server.

recv

Retrieve a remote file and store it on the local system.

Format

recv *remote-filename* [*local-filename*]

Description

Use this command to retrieve a remote file and store it in your local working directory.

If you do not specify a *local-filename*, **ftp** gives the retrieved file the same name it has on the remote host, within the limitations of the two file systems.

If there already is a local file with the same name as the *remote-filename* and you don't want to overwrite it, specify a different *local-filename*. If **runique** is on, however, you don't have to specify a different *local-filename*.

The transfer uses the current settings for type, mode, and structure. This command is identical to the **get** command.

Example

```
ftp> recv test ⤶
200 PORT command okay.
150 Opening data connection for test (127.0.0.1,1042) 8 bytes
226 Transfer complete.
Received 8 bytes.

ftp>
```

Retrieves the remote file **test** and stores it on the local host with the filename of **test**.

Error Messages

`Invalid combination of transfer parameters`	The combination of transfer parameters (mode, type, and structure) you used is illegal.
filename`: not a plain file`	You tried to transfer a directory or device file.
`ftp: fdopen failed`	The **fdopen** call for socket failed on the command connection end.

The error `ftp_recv_data with error: ftp-error` indicates that the data received contained one of the following errors:

ftp-error	What It Means
`Error on read`	**ftp** detected an error while reading from the file or socket.
`Error on write`	**ftp** detected an error while writing to the file or socket.
`Unexpected EOF`	**ftp** found a premature End-of-File delimiter while reading from the socket.
`Wrong escape sequence`	**ftp** found an incorrect escape sequence while receiving a file in Record structure and Stream mode.
`Page is not multiple of logical byte`	The page size defined for Page structure is not a multiple of logical byte size.

reinit

Terminate user and reinitialize the command connection.

Format

reinit

Description

If no data transfer is in progress, this command terminates the user process and resets all parameters to their default state. The command connection remains open, but you will prompted to login with the **user** and **password** commands.

If data transfer is in progress, this command allows the transfer in progress to complete, but terminates the user process and flushes all other input and output. It leaves the command connection open and resets all transfer parameters to their default values.

Example

```
ftp> reinit ↵
200 REIN command okay.
```

Terminates user process, resets all transfer parameters to their default values, but leaves the command connection open.

remotehelp

Request help from the remote FTP server.

Format

remotehelp [*command*]

Description

Use this command when you want help from the remote FTP server. To list all the ftp commands supported by the foreign system, enter the **remotehelp** command without any arguments. To display the command description of a particular **ftp** command supported by the foreign system, enter the **remotehelp** command with the *command* as an argument.

Example

```
ftp> remotehelp ↵
```

The response to this command depends on the remote implementation of FTP.

rename

Rename a remote file.

Format

rename *old-filename new-filename*

Description

Use this command to rename a file on a remote system. If you do not specify *old-filename* and *new-filename*, **rename** prompts you for them. If *old-filename* does not exist, **rename** reports an error. The **rename** command displays a syntax description when you specify *old-filename* without *new-filename*. Consult your remote operating system's manuals for the operating system filenaming conventions.

Example

```
ftp> rename kmo test ↵
350 File exists, ready for destination name.
200 RNTO command okay.

ftp>
```

Renames the remote file **kmo** to **test**.

restart

Restart the last aborted transfer.

Format

restart

Description

Use this command to restart a transfer aborted by a system or network crash. You can also use this command to restart a transfer that you aborted. The recovery procedure works if: 1) the aborted transfer was done in **block** or **compress** mode; and 2) you made no attempt to do another file transfer (in **block** or **compress** mode) before invoking the **restart** command.

Example

```
ftp> restart ↵
```

The transfer in progress when the system crashed will continue from where it stopped.

Error Messages

`No restart information`	**ftp** cannot read file **/usr/spool/ftp/***user-name***/marker_cmd**, which contains the restart information.
`Restart - cannot change local directory`	The **restart** command cannot change to the directory containing the file you want to restart.
`Cannot position file` *filename* `for recovery`	File position from which the transfer should continue was specified incorrectly.

rmdir

Delete a remote directory.

Format

rmdir *remote-directory-name*

Description

Use this command to delete a directory on a remote host. Some remote operating systems do not use directories in their file structure. Check your remote host's manuals to see if its operating system uses directories.

Example

```
ftp> rmdir problems ↵
200 Port command okay
```

Deletes the remote directory named **problems**.

runique

Toggle unique naming of files received from other systems.

Format

runique

Description

Use this command to toggle unique naming of files transferred from other systems. When **runique** is on, if a file on the local machine has the same name as the file transferred, a number is appended to the filename of the transferred file. The numbers assigned per transfer run consecutively from 1-99. **runique** is off by default.

Example

```
ftp> runique ↵
Receive unique file names on.

ftp> status ↵

FTP-user:
Connected to leary25
Mode: stream; Type: ascii; Form: non-print; Structure:file
Verbose: on; Bell: off; Prompting: on; Globbing: on
Store unique: off; Receive unique: on
Hash mark printing: off; Use of PORT cmds: on

200-FTP-daemon:
Connected to farah10
Mode: stream; Type: ascii; Form: non-print: Structure:file
EOR delimiter: <NL>
Page size is: 2048

200 STAT command okay.

ftp>
```

Turns on unique naming of files received from other systems. The **status** command shows that the parameter **receive unique** is indeed on.

send

Store a local file in your remote working directory.

Format

send *local-filename* [*remote-filename*]

Description

Use this command to store a local file in your remote working directory. **ftp** uses the current settings for type, mode, and structure.

If you omit the option *remote-filename*, the file will be given the same name as the argument *local-filename*, within the limitations of the two file systems. If there already is a remote file with the same name as the *local-filename* and **sunique** is not on, **ftp** will overwrite the remote file.

This command is identical to the **put** command.

Example

```
ftp> send test ↵
200 PORT command okay.
150 Opening data connection for test (127.0.0.1,1030)
226 Transfer complete.
Sent 22 bytes.

ftp>
```

Stores local file **test** on the remote system and names it **test**.

Error Messages

`Invalid combination of transfer parameters`	The combination of transfer parameters (mode, type, and structure) you used is illegal.
filename`: not a plain file`	You tried to transfer a directory or device file.
`ftp: fdopen failed`	The **fdopen** call for socket failed on the command connection end

The error `ftp_send_data with error: ftp-error` indicates that the data sent contained one of the following errors:

ftp-error	**What it Means**
`Error on read`	**ftp** detected an error while reading from the file or socket.
`Error on write`	**ftp** detected an error while writing to the file or socket.
`Unexpected EOF`	**ftp** found a premature End-of-File delimiter while reading from the socket.
`Wrong escape sequence`	**ftp** found an incorrect escape sequence while sending a file in Record structure and Stream mode.
`Page is not multiple of logical byte`	The page size defined for Page structure is not a multiple of logical byte size.

sendport

Turn on or off the use of the PORT command.

Format

sendport

Description

Use this command to turn on or off the use of the **PORT** command. By default, **ftp** tries to send a **PORT** command when establishing a connection for each data transfer. If the **PORT** command fails, **ftp** uses the default data port. When **PORT** commands are disabled, no attempt is made to use them.

CAUTION: We recommend that you keep **sendport** on unless you know that the foreign server does not recognize the **port** command.

Example

```
ftp> sendport ↵
Use of PORT cmds off.
```

This example shows how to use **sendport** to turn off the use of the **PORT** command.

site

Display specific information about the remote system.

Format

site

Description

Use this command to display specific information about the remote system. This command works only if the remote server supports the **site** command. The following information is displayed: the operating system, the storage structure, the storage representation type, the storage filler, the acceptable local byte size, the default page size, and the default end-of-record delimiter.

Example

```
ftp> site ↵
200-Site information:

Operating system:             DG/UX
Storage structure:            File
Storage representation type:  ascii
Storage filler:               NULL
Acceptable local byte size:   multiple of 8 bits
Default page size:            2048 transfer bytes
        can be changed using PAGE command
Default EOR delimiter:        <NL>
        can be changed using SEOR command

200 SITE command okay.

ftp>
```

Displays specific information about the remote system.

status

Show the current status of FTP connection.

Format

status

Description

If no data transfer is in progress, this command displays the status of FTP local and remote environments. This command displays the current values for the transfer parameters, structure, mode, format, and type as well as for verbose, bell, prompt, hash, and sendport modes.

If a data transfer is in progress, this command displays the status of FTP local and remote environments and continues the data transfer.

Example

```
ftp> status ↵

FTP-user:
Connected to sys16.
Mode: stream; Type: ascii; Form: non-print; Structure:file
Verbose: on; Bell: off; Prompting: on
Store unique: off; Receive unique: off
Hash mark printing: off; Use of PORT cmds: on

200-FTP-daemon:
Connected to sys16
Mode: stream; Type: ascii; Form: non-print: Structure:file
EOR delimiter: <NL>
Page size is: 2048

200 STAT command okay.

ftp>
```

Displays the status of the FTP local and remote environments.

struct

Set the structure of the file transfer.

Format

struct [*structure*]

Description

Use this command to set the file transfer structure. If you do not specify an argument, **struct** displays the current structure. The default structure is file. The table below lists the file transfer structures and what they mean.

Structure	Meaning
file	There is no internal structure and the file is a continuous sequence of bytes.
page	The file is made up of independent indexed pages.
record	The file is made up of sequential records.

The DG/UX operating system does not support record-structured files. If you specify record structure, all EOR delimiters will be converted to a new line character for storage.

The **page** structure will only be accepted with the Local Byte type and is supported in the Stream mode only (see **type** and **mode** commands). We recommend that you set the local byte size to be 32.

Example

```
ftp> struct file ↵

200 STRU F ok.
Using file structure to transfer files.

ftp>
```

Sets file transfer structure to file.

sunique

Toggle unique naming of files sent to other systems.

Format

sunique

Description

Use this command to toggle unique naming of files transferred to other systems. When **sunique** is on, if a file on the remote machine has the same name of the file being transferred, a number is appended to the filename of the transferred file. The numbers assigned per transfer run consecutively from 1-99. **sunique** is off by default.

Example

```
ftp> sunique ↵
Send unique file names on.

ftp> status ↵

FTP-user:
Connected to leary25
Mode: stream; Type: ascii; Form: non-print; Structure:file
Verbose: on; Bell: off; Prompting: on; Globbing: on
Store unique: on; Receive unique: off
Hash mark printing: off; Use of PORT cmds: on

200-FTP-daemon:
Connected to farah10
Mode: stream; Type: ascii; Form: non-print: Structure:file
EOR delimiter: <NL>
Page size is: 2048

200 STAT command okay.

ftp>
```

Turns on unique naming of files being transferred to other systems. The **status** command shows that the parameter **Store unique** is indeed on.

type

Set the character transfer type.

Format

type [*t-name* [*vertical-format*]]

Description

Use this command to set the character transfer type. The character transfer type determines how the characters are represented as they are transferred.

The *t-name* is the character transfer type. If you omit the *t-name* argument, the current transfer type is displayed on your terminal. If you include *t-name*, the type is set to *t-name*.

Some transfer types have *vertical-format* options. If you do not specify a *vertical-format*, the format will be no-print (the default). See the table below for the transfer types and available *vertical-format* options.

Type	Vertical-format
ascii	[**no-print** | **telnet** | **carriage-control**]
ebcdic	[**no-print** | **telnet** | **carriage-control**]
binary	
image	
local-byte	[*byte_size*] Default size is 8 bits.

NOTE: The *byte_size* must be a multiple of 8 bits and a power of 2.

The **binary** transfer type is the same as the **image** transfer type. The **ascii** transfer type is the default.

The *vertical-format* options determine the vertical controls in how the information is represented on a printing device. The default vertical format is no-print. The following list defines the *vertical-format* options:

no-print
The file need not contain vertical format information. A printer process can assume standard values for spacing and margins. Typically, this format is used with files that will be stored or processed. This is the default *vertical-format*.

telnet
Use TELNET format controls. The file contains ASCII/EBCDIC vertical

format controls, such as CR, LF, and FF, which the printer process can interpret. The sequence CRLF denotes the EOL.

carriage-control
The file contains American National Standards Institute (ANSI) FORTRAN vertical format control characters. If lines and records are formatted according to the ANSI standard, vertical format controls are read in before the data is printed.

Example

```
ftp> type ebcdic↲
200 TYPE E N ok.
Using ebcdic type to transfer files.
Using non-print form.

ftp>
```

Sets the current transfer type to EBCDIC and the vertical-format to no-print.

```
ftp> type binary ↲
200 TYPE I ok.
Using binary type to transfer files.

ftp>
```

Changes the transfer type to BINARY.

```
ftp> type ascii ↲
200 TYPE A N ok.
Using ascii type to transfer files.
Using non-print form.

ftp>
```

Changes the transfer type back to ASCII.

user

Send new user information for login message or access to remote directory.

Format

user *username* [*password*] [*account*]

where

username	is the name of your user profile on the remote host.
password	is the password to your user profile on the remote host. This is an optional argument for the **user** command. If a password is required on the remote system and you omit this argument, **ftp** will prompt you for your password.
account	is the account number. For details, see the earlier description of the **account** command.

Description

Use this command for a login message, for access to a remote directory, or for changing the access control information.

If **ftp** has the automatic login facility enabled and you have a proper entry in your **.netrc** file, you will not be prompted for your *username* after you enter the **open** command. The **open** command is necessary for establishing an FTP connection from your FTP local environment.

Example

```
ftp> user ⤸
(username)
```

Entering the **user** command causes **ftp** to prompt you for your username.

Error Message

`Login failed`	One of the following server commands failed: **user**, **pass**, or **acct** (if it exists).

verbose

Turn verbose mode on or off.

Format

verbose

Description

Use this command to set verbose mode to on or off. In verbose mode, **ftp** displays all responses from the FTP server. By default, verbose mode is on, unless you do not invoke the FTP user program from a **tty**.

Example

```
ftp> verbose ↵
Verbose mode on.

ftp>
```

Turns verbose mode on.

```
ftp> verbose ↵
Verbose mode off.

ftp>
```

Turns verbose mode off.

?

Display a list of ftp commands or information on a specific command.

Format

? [*command*]

Description

Use this command when you need information on an **ftp** *command*. To display a list of all ftp commands, enter the ? command without any arguments. To display the meaning of a particular **ftp** command, enter the ? command with the *command* as an argument. The **help** command is identical to the ? command.

Example

```
ftp> ? ↵
```

Displays all **ftp** commands.

```
ftp> ? ! ↵
!       escape to the shell
ftp>
```

Displays the format line and command description of the ! command.

!

Create a shell process as a child of ftp process.

Format

! [*shell-command*]

Description

Use this command when you are using **ftp** and you want to work in the shell without terminating your FTP connection. Optionally specify an *shell-command* as an argument. If you specify a *shell-command,* the shell executes it and then returns you to command mode.

If you do not specify a shell-command, **ftp** creates a shell child on the local host. You remain in the shell until you enter the shell **exit** command.

Example

```
ftp> !⤶

$
```

Creates a shell process as a child of **ftp**. A DG/UX banner and the shell prompt will appear on your terminal.

End of Chapter

Chapter 4 Understanding and Using the Background File Transfer Program

This chapter is new to this document. It describes a feature that was unavailable in previous releases. Because the entire chapter is new, no revision bars are present.

The Background File Transfer Program (BFTP) is a service built upon the model of the File Transfer Protocol (FTP) that allows you to transfer files across a TCP/IP network. Unlike FTP, BFTP allows you to transfer files in the background, so you do not have to be directly involved when the actual file transfer takes place.

BFTP, because it can transfer files in the background, is particularly useful if your network becomes congested and has long delays during times of peak usage or if communication costs are lower during off-peak hours. In addition, as the world becomes more interconnected, planned and unplanned outages of hosts, gateways, and networks can make it difficult to successfully transfer files in the foreground.

BFTP is composed of two parts: a user interface program and a file transfer control (FTC) daemon. The user interface program collects parameters that describe the required transfer, and the FTC daemon transfers the files. These two programs run on the same host, called the BFTP control host. The user interface is implemented by **/usr/bin/bftp**. The FTC daemon is **/usr/bin/fts** (File Transfer Service).

Users interact with the user interface program to specify all the parameters necessary to transfer files from the source host to the destination host. The source and destination hosts can be on different hosts from the BFTP control host.

Background file transfer has a number of potential advantages for a user:

- No Waiting: You can request a large transfer and ignore it until a notification message arrives through electronic mail.
- End-to-End Reliability: The FTC daemon can try a transfer repeatedly until it either succeeds or fails permanently. This provides reliable end-to-end delivery of a file even if the source or destination host is down or there is poor network connectivity.
- Multiple File Delivery: You can transfer several files at once by setting the multiple flag and using the wildcard character (*).
- Deferred Delivery: You can delay transfer of large files until an off-peak period when the network is less congested.

One disadvantage of BFTP is that if you make a mistake while entering parameters to the user interface program, you may not recognize the problem before submitting the job. The user interface program verifies some parameters as you enter them, such as the source and destination hosts. Others are not easily verified. To check your parameters before submitting your request, use the **verify** command. For more information on the **verify** command, see the section "Using BFTP Commands" later in this chapter. If you detect a mistake after submitting a request but before the transfer begins, use the **cancel** command to cancel the request or the **find** command to correct and resubmit the request. For more information on the **cancel** and **find** commands, see "Using BFTP Commands" later in this chapter.

Understanding BFTP

The BFTP user interface program and its FTC daemon program must execute on the same host, called the BFTP control host. Through the user interface program, a BFTP user supplies all of the parameters needed to transfer a file from the source host to the destination host, where the source and destination hosts may be different from the BFTP control host. The parameters a user must supply include the following:

- source and destination hostnames
- login names and passwords on the source and destination hosts
- source and destination pathnames

In addition to the above parameters, you can specify a number of optional control parameters. These parameters include the following:

- Source file disposition -- Copy, move (copy and delete), or simply delete the source file. The default is copy.
- Destination file operation -- Create/Replace, append to, or create a unique destination file. The default is create/replace ("STOR").
- FTP Parameters -- Explicitly set any of the FTP type, mode, or structure parameters at the source and destination hosts.
- Multiple Transfers -- Enable "wildcard" matching to transfer several files at once.

After setting the above parameters, you can use the **submit** command to set your request for transfer in the background. To do so, you must specify the time of day for the first attempt of the transfer, specify a mailbox to which a completion notification message will be sent, and then submit the request to the FTC daemon queue. You can then exit the BFTP user interface program.

If the first attempt to transfer a file fails, BFTP attempts to transfer the file until it is successful, it fails permanently, or until it reaches the specified maximum number of attempts. If the transfer fails permanently, the FTC daemon sends a notification message to your mailbox. If there is a temporary failure (for example, a broken TCP connection), the FTC daemon logs the failure and retries the transfer after some timeout period. The retry cycles are repeated until the transfer succeeds or until some maximum number of tries specified has been reached. In either case, a notification message is sent to your mailbox.

You can check on the progress of the transfer by re-entering the BFTP user interface program and invoking the **status** command. You can get more detailed information on a transfer by using the **find** command. You supply the **find** command with the key that you defined with the request. It will then display the transfer parameters and the status of the request. You can then cancel the request or leave it in the queue.

Using the BFTP User Interface

The purpose of BFTP is to simplify the file transfer process and to place the burden of reliability on the BFTP control host. The BFTP user interface program allows you to submit the transfer request to the BFTP control host. This interface program provides the command mode that allows you to provide the information necessary to describe and control file transfers. The user interface provides the following features for ease of use:

- prompting for commands,
- defaults for many commands and subcommands, and
- on-line help information for each command and BFTP in general.

Setting the BFTP Working Directory

BFTP maintains a working directory for control files and request files. Usually the working directory is **$HOME** (as set by **sh** or **csh**), but you can use another by setting the environment variable **$BFTPDIR** to the desired directory. Using a separate directory for BFTP files is useful to avoid clutter and confusion with names. Also, keeping BFTP files in a directory that is not normally mounted by another host helps avoid problems caused by simultaneous access to the same directory through NFS. A suggested name for the BFTP working directory is **/var/spool/bftp/***user*. Permissions should allow *user* at least write and execute access.

During routine operation, BFTP creates temporary files that are associated with your request (for example **.req**, **.atjob**, **.msg**, **.cmd**, **.list**). Should something go wrong during the execution of your BFTP request, you may have to remove these files manually.

BFTP allows you to create request files that contain BFTP transfer parameters used by request commands (see "BFTP Request Commands" later in this chapter). BFTP maintains other files, such as **bftp_saved_info** and **bftp-save.*** as well.

Invoking the BFTP User Interface Program

To enter the BFTP user interface program, you only need to invoke BFTP. Invoke BFTP as follows:

```
% bftp ↵
```

BFTP returns the version of the program being used, a message on how to receive help, and a list and brief definition of special editing characters. Here is an example:

```
Background File Transfer: Version 2.01 5/31/89, with extensions

For help, type 'help', or 'explain'.

Special editing characters are as follows:

   <return>  Accept current command/field.
   <escape>  Complete current command/field, or display default.
   <space>   Complete and delimit current command.
   <delete>  Erase last character.
   control-L Refresh screen.
   control-R Refresh line.
   control-U Erase line.
   control-W Erase current token.
   ?         List legal options.

BFTP>
```

In addition to the information above, BFTP could give one of two error messages when you invoke the program. Here is one error message you might see:

```
There is already a bftp on bitstream using /var/spool/bftp/user:
     UID   PID  PPID  C    STIME     TTY   TIME COMMAND
      user  459  393   7    16:41:54 p5    0:00 bftp
Only one bftp per directory may run at one time.
You may use a different directory with $BFTPDIR.
```

This error indicates that BFTP has detected that another BFTP on the same host is operating in the same BFTP directory.

The second error message you could see when invoking BFTP is this:

```
Can't tell if a bftp on another host is already using /udd/polaris/user.
If it is, please quit now, or your bftp_saved_info file may become damaged.
```

This error message indicates this BFTP directory, which is mounted on NFS, does not have remote locking available. Therefore, your files could be damaged by other BFTP sessions running in the same directory. For remote locking to be enabled, **lockd** must be running on the NFS server for that directory and on all of the clients.

Exiting the BFTP User Interface Program

To exit the BFTP user interface program, you use the **quit** command from the BFTP main prompt (BFTP>). Exit BFTP as follows:

```
BFTP> quit ↵
Bye
%
```

BFTP responds with Bye to indicate that you are leaving the BFTP program and returns you to the shell.

Using the BFTP Help Facility

BFTP has a help facility that provides introductory information on BFTP and help messages for commands and subcommands. Invoke each of the help commands from the BFTP main prompt (BFTP>). The **?** command can also be used when you have invoked a command and are in the process of supplying values for parameters (for example, when using the **prompt** command). The help commands provided are as follows:

? Lists all BFTP commands or parameter values that are legal at the current time.

help ↵
Lists all available commands and special editing characters and briefly describes what they do.

help *command* ↵
Briefly explains what the specified command does. If the specified command has subcommands also, **help** provides a list and description of these as well.

explain ↵
Describes how BFTP works, including an introduction, background information, BFTP terminolgy, and descriptions of several commands and subcommands.

To exit the BFTP program, issue the **quit** command from the BFTP main prompt (BFTP>).

Transferring Files in BFTP

The typical procedure for transferring files in BFTP is to set up a set of parameters that define the transfer and then submit the request to the FTC daemon. There are three modes in which you can submit a file or files for transfer. These modes are as follows:

- Background Operation: In this mode, you set all of the transfer parameters that define the kind of transfer you want. You then request a reliable background file transfer, using the **submit** command to send the parameters to the FTC daemon.

- Foreground Verification, Background Operation: In this mode, you set all of the transfer parameters that define the transfer you want. You then use the **verify** command to check that all the file transfer parameters are valid. This command causes BFTP to attempt to connect to the FTP servers on both the source and the destination hosts, log into both, verify the FTP parameters, and verify that the specified source file is present.

  Once the **verify** command has successfully completed, you can use the **submit** command to schedule the actual file transfer.

- Foreground Operation: In this mode, once you have set all of the transfer parameters, you use the **transfer** command to perform the specified third-party transfer in foreground mode.

Controlling the Status of BFTP Transfers

BFTP provides a way for you to obtain status information about, cancel, and suspend earlier requests. To provide this service, BFTP has to have a means of distinguishing between different users' requests. BFTP uses two methods to distinguish requests. First, it uses a user-supplied character string, called a keyword, to distinguish requests. The **submit** command prompts you to supply a unique keyword before you send your request to the FTC daemon. The second method is a request ID that BFTP assigns transfers when they are submitted. After submitting the request, you can do the following:

- Use the **status** command to obtain the state of your requests (queued, transferring, or completed).

- Use the **find** command, with the keyword or request ID as an argument, to obtain status information and parameter settings on a particular request. You can then change or cancel the request.

- Use the **hold** command, with your request ID as an argument, to suspend a transfer that is in progress. You cannot use this command to suspend a transfer that is not actually running.

- Use the **unhold** command, with your request ID as an argument, to resume a transfer previously suspended by the **hold** command.

- Use the **cancel** command, with your request ID as an argument, to cancel the corresponding request.

Making a Series of Requests

To make a series of similar requests, you need only to change the individual parameters that differ from the preceding request and then issue a new **submit** command for each new request. You can set individual parameters using either of two procedures. (These procedures can also be used to correct a mistake made in entering a particular parameter.) The simpler (but lengthier) procedure is to use the **prompt** command to review the current set of parameters. While reviewing the current set, change the parameters necessary to set up the new request and use the Esc and New Line keys to retain the current values of the others.

The second procedure is a shortcut for BFTP experts; it involves using commands that set individual parameters usually set by the **prompt** and **set time** commands. These commands include parameters for both the source and destination hosts, such as **ddir**, **dfile**, **dhost**, **sdir**, **sfile**, and **shost**. For more information on these commands, see "BFTP Standard Transfer Commands" later in this chapter.

Displaying and Manipulating BFTP Parameters

You can display the current settings of all the BFTP parameters at any time with the **show** command. This command provides a complete list of the BFTP parameters, including those set only by the **set** command. To set all of these parameters to their initial values, use the **clear** command.

Finally, if you want to save the current set of parameters in a file or restore the parameters from a previously saved file, you can use the **request** command (see "BFTP Request Commands" later in this chapter).

Entering BFTP Commands

You can enter BFTP commands immediately after you invoke BFTP and receive the BFTP prompt (`BFTP>`). For convenience, you can abbreviate commands and subcommands by typing just enough characters to uniquely identify a command and pressing the space bar. You can also complete a partial command name by pressing the New Line key.

BFTP provides special editing characters to use on the command line while you are entering commands, subcommands, and parameters. Table 4-1 lists and describes these editing characters.

Table 4-1 BFTP Special Editing Characters

Command	What It Does
New Line key	Accepts the value displayed for this parameter, and continues to the next parameter, if any. If you have typed part of a command or parameter name, this command completes the name.
Esc key	Displays the default value (or last value set) for this parameter. You can accept this default by pressing the New Line key, or erase it with Ctrl-W and enter a different value for the parameter, followed by pressing the New Line key to accept the entered value.
Space bar	Completes and delimits the current command.
Del key	Erases the last character.
Ctrl-L	Refreshes the screen.
Ctrl-R	Refreshes the line.
Ctrl-U	Erases the current line.
Ctrl-W	Erases the value typed or displayed for the current parameter.
?	Lists the legal options available at the current time.

NOTE: If your BFTP editing keys do not work as documented, check that your $TERM variable, **stty** settings, and **editread** settings (if applicable) are correct. Verify that keyboard and screen behavior are correct on the shell command line and with other full-screen applications, such as **vi**.

Using BFTP Commands

This section lists and describes all of the **bftp** commands. The section is divided into the following five subsections:

- **BFTP Standard Transfer Commands**. This section lists and describes the commands (and parameters) that the **prompt** command uses to set up a typical transfer.
- **BFTP Information Commands**. This section lists and describes commands that allow you to get status information on transfers and help for commands.
- **BFTP Transfer Control Commands**. This section lists and describes the commands that allow you to clear transfer parameter settings, and find, change, and cancel transfers. These commands also allow you to put transfers on hold and leave BFTP.
- **BFTP Request Commands.** This section lists and describes the commands that allow you to save, delete, or read a file of current or previous transfer parameters.
- **BFTP Set Commands**. This section lists and describes the commands that allow you to set individual transfer parameters.

Each of the sections listed above describes the commands and includes examples. Most commands that require an argument will prompt you for the argument. Each command should be followed by pressing the New Line key.

BFTP Standard Transfer Commands

The commands in this section allow you to set the basic parameters necessary to transfer files and submit these parameters to the FTC daemon so that the transfer can take place later. They also allow you to start transfers, clear parameters, and exit BFTP. Table 4-2 lists and describes the commands included in this section.

Table 4-2 BFTP Standard Transfer Commands

Command	What It Does
clear	Returns all parameters to their default values.
ddir *directory_name*	Sets the destination directory. If **ddir** is not set and **dfile** is not a complete pathname, **dfile** will be relative to the user's home directory on the destination host.
dfile *file_name*	Sets the destination filename. Can be a full or a relative pathname. If **ddir** is not set and **dfile** is not a complete pathname, the pathname will be relative to **$HOME** on the destination host.
dhost *hostname/number user password*	Sets the destination host, user, and password. If the destination user does not have a password, the password argument is not required.
prompt	Prompts you for commonly-used parameters.
quit	Returns all parameters to their default values and exits the BFTP program.
sdir *directory_name*	Sets the source directory. If **sdir** is not set and **sfile** is not a complete pathname, **sfile** will be relative to the user's home directory on the source host.
sfile *file_name*	Sets the source filename. Can be a full or a relative pathname. If **sdir** is not set and **sfile** is not a complete pathname, the pathname will be relative to **$HOME** on the source host.
shost *hostname/number user password*	Set the source host, user, and password. If the source user does not have a password, the password argument is not required.
submit	Submits the current request for background FTP transfer.
transfer	Perform the current request in the foreground.

NOTE: A directory name should end with the directory delimiter for the host (for example, a dash (/)). Some hosts, however, may not require this.

The source and destination commands listed above, such as **ddir**, **dfile**, **sdir**, and **sfile,** are used to set their respective parameters separately. Though you can use these commands to set all the basic parameters for a transfer, the more efficient way to use these commands is to change one or two parameters from a previous transfer. The **prompt** command will prompt you for each of these parameters, allowing you to fill in values for these parameters without using the individual commands.

When you use the **prompt** command, BFTP prompts you for the following parameters:

```
copy/move/delete: [copy | move | delete]
ascii/ebcdic/image/local:
[ascii|ebcdic] [nonprint|telnet|carriage-control]
or
[image]
or
[local] byte size

Source --
 Host: hostname/number
 User: user
 Password: password
 Dir: directory
(either an absolute path, or relative to the login)
 File: filename

Destination --
 Host: hostname/number
 User: user
 Password: password
 Dir: directory
 File: filename
```

When you have finished providing values for the parameters, BFTP displays the current values for all parameters, including those the **prompt** command does not mention. Parameters not mentioned by the **prompt** command are initialized with default values. These parameters can be changed with the **set** commands (see "BFTP Set Commands" later in this chapter).

If you want to perform the transfer request in the foreground, use the **transfer** command. This command immediately tries to perform the request. However, if you want to place the request in the background, use the **submit** command.

NOTE: You must have permission to submit **at** jobs on your host. To get permission, your username must be in the file **/var/spool/cron/at.allow**. If you want to run a large number of simultaneous transfers, you may want to increase the **/var/spool/cron/queuedefs** limits. Refer to the **at**(1) manual page or the *User's Reference for the DG/UX™ System* for details.

When you use the **submit** command, BFTP prompts you for the following information:

```
StartTime: date and/or time
ReturnMailbox: user@host
RequestKeyword: an optional keyword of your choice
```

When supplying values for the parameters above, you can either use the Esc key to choose the default values, or supply your own values. The default time to start the transfer is now, while the default return mailbox is your (the user's) mailbox on the local machine. There is no default value for the keyword; you must choose a keyword if you want to use the **find** command later to locate this request by keyword. Otherwise, you can omit the keyword. Keywords need not be unique.

To change the default time, type in the time (and date, if different from the current day) you desire. You can enter the time either by typing in the hour and minute according to 12-hour clock time (for example, 9:00 pm) or according to 24-hour clock time (for example, 21:00). Specifying a time already passed is equivalent to specifying **now**. When specifying the start time as **now**, there is a 90-second delay.

To specify the date, you can type in the month, day, and, optionally, the year in one of three ways. You can write out the month and include the day and year (for example, July 29, 1990); use all-figure style (for example, 7/29/90); or use military style (for example, 29 July 1990).

Let's assume that you want to send a file from a directory you are using on a system named **toni58** to another system named **maya49**. A simple way to transfer this file is to use the **prompt** command, which prompts you for all the necessary parameters. If your user name is **wilson** and the file you want to transfer is **sula**, then you could use the **prompt** command as follows:

```
BFTP> prompt
copy/move/delete: copy
ascii/ebcdic/image/local: ascii: (format) nonprint

Source --
 Host: toni58
 User: wilson
 Password: (password will not be echoed)
 Dir: /usr/writes/books/
 File: sula

Destination --
 Host: maya49
 User: wilson
 Password: (password will not be echoed)
 Dir: usr/writes/poems/
 File: caged_bird
```

After providing the above parameters, BFTP will display values for all of the parameters, including those not mentioned by the **prompt** command. BFTP's display would appear as follows:

```
Source--
   Host: 'toni58'
   User: 'wilson'
```

```
      Pass: SET
      Acct: ''
      Dir: '/usr/writes/books/'
      File: 'sula'
      Port: 21

   Destination--
      Host: 'maya49'
      User: 'wilson'
      Pass: SET
      Acct: ''
      Dir: '/usr/writes/poems/'
      File: 'caged_bird'
      Port: 21

   Structure: file, Mode: stream, Type: ascii, Format: nonprint
   Multiple matching: FALSE, Return mailbox: 'wilson@toni58'
   Remaining tries: 5, Current retry interval: 15 minutes

   Start immediately.  Verbose mode=OFF
```

At this point, you can issue the **submit** command to set up the transfer in the background. To set up the transfer for 9:45 a.m. on July 29, 1990, you would use the **submit** command as follows:

```
BFTP> submit

Checking parameters...

StartTime: 29 July 1990 9:45 am
ReturnMailbox: wilson@toni58
RequestKeyword: (your keyword will not be echoed)
Request bftp636648150 submitted to run at Jul 29 09:46:30 1990 EST
```

After you enter your unique keyword, BFTP registers your request, provides a number to identify your request (for example, bftp636648150), and returns a message indicating that your job has been submitted. At this point, you can exit BFTP.

BFTP Information Commands

The commands in this section allow you to get status information on transfers and help for commands. Table 4-3 lists and describes the commands included in this section.

Table 4-3 BFTP Information Commands

Command	What It Does
?	Displays a list of commands available to you at the time.
explain	Displays a short explanation of how to use BFTP.
help *command*	Prints local help information. If a command is supplied as an argument, prints information only on that command.
status	Lists the transfers that are currently submitted and provides a summary of each transfer. Use the **find** command for more detailed information on a transfer.
verify	Makes the connections necessary to conduct the current transfer, using the specified parameters. Does not make the transfer, but checks the parameters.
show	Displays the current parameter values.

The **status** command provides information on transfers that have been submitted. An example of the information that the **status** command returns is as follows:

```
Request ID              Time to run               Status

bftp636826346           at 15:01 Mar 7,1990       Complete.

bftp636837322           at 17:01 Mar 7,1990       Waiting to run.
```

The **status** command provides a quick summary of all transfers. The information provided is broad and sometimes inexact. For example, the status message `Running...` applies once the first attempt begins. However, when you receive the message, the transfer may be currently awaiting a retry. Similarly, `Complete.` means that no more activity will occur for that request. The message does not mean that the transfer completed successfully. You can get a chronological activity log by using the **find** command, or by examining the notification message that is mailed when a request completes.

The **verify** command is useful to check parameters before you submit transfer requests. It checks all of the transfer parameters by logging in to the source and destination systems and setting up the transfer. The process is echoed to the screen so that you can see whether or not problems exist. For example, if, in the process of typing your parameters, you entered an incorrect destination password, the transfer would fail. The **verify** command would allow you to discover your error before submitting your transfer request. The following example shows how to use the **verify** command.

```
BFTP> verify ⤸
```

The system will display the following:

```
Checking parameters...


  Mar  7 17:01:01 1990 EST: starting...

    Request type: VERIFY
    Source: toni58,wilson,XXX,,21,/usr/writes/books/wilson,sula
    Destination: maya49,wilson,XXX,,21,/usr/writes/poems,caged_bird
    Stru: F, Mode: S, Type: A N, Creation: STOR
    Multiple matching: FALSE


Connect to: toni58, 21
Connect to: maya49, 21
Logging in: wilson, on toni58
Logging in: wilson, on maya49

maya49 -- Login failure
         Error: 530 Login incorrect.

Status: FAILED PERMANENTLY

  toni58 <== QUIT
  maya49 <== QUIT

  Mar  7 17:01:24 1990 EST: completed unsuccessfully.
```

You can use the **dhost** command to enter the correct password and then submit your request (see "BFTP Standard Transfer Commands" earlier in this chapter).

BFTP Transfer Control Commands

The commands in this section allow you to find and cancel transfers. These commands also allow you to put transfers on hold.

NOTE: Because of Data General-specific extensions, the **hold**, **unhold**, and **cancel** commands are available only when the source host is running the 4.30 (or later) release of **ftpd**. If you are not running the 4.30 release of **ftpd**, the **status** command will indicate that these commands work, but they do not. Table 4-4 lists and describes the commands included in this section.

Table 4-4 BFTP Transfer Control Commands

Command	What It Does
cancel	Prevents the specified transfer from taking place, or stops it after it has begun.
find	Finds and displays the parameters for a transfer request and a log summarizing transfer activity.
hold	Suspends a transfer that is currently active (`running` and not between retries). This command cannot be used for transfers that are not actually running.
unhold	Restarts a transfer that has been suspended by the **hold** command.

The **find** command displays the parameters for a specified transfer request. When you invoke this command, BFTP prompts first for the request ID and, if not supplied, then for the request keyword. The request ID is the number BFTP returns when you submit your transfer request (for example, bftp583101774) and the request keyword is the unique string you chose to name your transfer request. You must supply either the request ID or request keyword to identify a request. If you enter a request keyword, BFTP displays all requests that match your keyword.

Let's assume that you have submitted several transfer requests but want to check the parameters for a particular request that you gave the keyword "hanna". To check that transfer, you would use the **find** command as follows:

```
BFTP> find
RequestID (optional): ↵
RequestKeyword: hanna

Request: bftp636678943

   Request type:Copy
   Source: toni58, wilson, XXX,, 21 /usr/writes/books/, sula
   Destination: maya49, wilson, XXX, 21, /usr/writes/poems/, caged_bird
   Stru:F, Mode: S, Type:A N, Creation: STOR
   Multiple matching: False, Return mailbox: 'wilson@toni58'
   Remaining tries: 5, Current retry interval: 15 minutes

History:
Request bftp636678943 submitted to run at Mar 5 17:01:30 1990 EST.
```

After displaying the above summary, BFTP asks whether you would like to change or cancel the request. If you do not want to change or cancel the request, you would type **no** in response to the prompts, as shown below:

```
Do you wish to change this request? no
Do you wish to cancel this request? no
```

If you indicate **yes** to either of the above questions, BFTP cancels the request. If you indicate that you want to change the request, BFTP resets the current parameters to those of the request you want to change. At this point, you can make changes to the parameters and use the **submit** command to requeue the request. BFTP assigns and displays a new request ID.

Although extremely rare, a system crash (or the interruption of the BFTP program) at just the right moment can prevent a request from being queued. This condition also occurs while a transfer is active.
When the **find** command locates such a request, it displays this warning:

```
Your request is NOT currently queued.
```

If this happens, and the output of the **find** command appears abnormal, you can read in your request and resubmit your transfer as follows:

```
Your request is NOT currently queued.
Do you wish to change this request? yes
```

(BFTP displays the parameters that have been read in.)

```
Previous request cancelled.
Use the 'submit' command to submit a new request.
```

In addition to the **find** command, you can use the **status** command to get summary information on transfers that you have submitted.

BFTP Request Commands

The commands in this section allow you to save a set of BFTP transfer parameters in a file for future use. These files, called request files, are stored in your home directory or **$BFTPDIR** (see "Setting the BFTP Working Directory" earlier in this chapter). The subcommands provided with the request commands let you list, read, write or delete available request files. These files are particularly useful when you want to make similar transfer requests periodically. By using the request files, you can save a set of parameters once and reload them many times.

Table 4-5 lists and describes the commands included in this section.

Table 4-5 BFTP Request Commands

Command	What It Does
request delete *name*	Deletes request file **bftp-save.***name*.
request list	Lists all request files.
request load *name*	Reads **bftp-save.***name* in as the current request.
request store *name*	Saves the current request in a file named **bftp-save.***name*. Currently, name can consist of numbers and letters only.

You must supply a name for the request file; BFTP prepends **bftp-save.** to the name you supply. Consequently, request files are listed as follows: **bftp-save.***name*. Following is an example of how to save a set of parameters.

```
BFTP> request store ↵ (request name) books ↵

File name = /usr/wilson/bftp-save.books
BFTP>
```

After you have saved a set (or sets) of parameters, you can list, delete, or load that set of parameters by using the appropriate command above. For example, let's assume that you have saved two sets of parameters in files named **bftp-save.books** and **bftp-save.essays**. To use the parameters in one of those files, you could first list your saved files and then choose one. An example of how to do this follows.

```
BFTP> request list

  books   essays

BFTP> request load books

Reading /usr/wilson/bftp-save.books...

    Request type: COPY

Source--
    Host: 'toni58'
    User: 'wilson'
    Pass: SET
    Acct: ''
    Dir:  '/usr/writes/books/'
    File: 'sula'
    Port: 21

Destination--
    Host: 'maya49'
    User: 'wilson'
```

```
        Pass: SET
        Acct: ''
        Dir:  '/usr/writes/poems/'
        File: caged_bird
        Port: 21

    Structure: file, Mode: stream, Type: ascii, Format: nonprint
    Multiple matching: FALSE, Return mailbox: ''
    Remaining tries: 5, Current retry interval: 15 minutes

    Start immediately.  Verbose mode = OFF
```

At this point, you can change any parameters you wish and then use the **submit** command to log your new transfer request.

BFTP Set Commands

The commands in this section allow you to set many of the less commonly used BFTP/FTP parameters. The parameters controlled by these commands have default values, so you have to be concerned with them only if you want to change the defaults. For those commands that have subcommand structures, BFTP prompts you for the required values. For example, when you are setting the **type** parameter, BFTP prompts you to specify values for the **type** and **format** of the data you transfer.

Table 4-6 lists and describes the commands included in this section.

Table 4-6 BFTP Set Commands

Command	What It Does
set account	Sets the account for logging in to the source and destination hosts. Many hosts do not require this.
set append	Sets to true or false the request to append transferred files to destination files. If the destination file does not exist, the file is created. The default is false. Setting this command to true turns off the **set unique** command.
set copy	Source file will be copied to the destination filename. Copy is the default.
set delete	Source file will be deleted. Note that when delete is set, no connection is made to the destination host, so only source parameters are required.
set mailbox	Sets the mailbox where BFTP transfer results are returned. The mailbox should be in standard internet format, for example: **farah@doc**. The default is *logname@local_system*.
set mode	Sets the FTP transfer mode to stream, block, or compress. The default mode is stream.
set move	When **set copy** is the default, the source file will be deleted after it has been copied.
set multiple	Sets to true or false the request to transfer multiple files. To use wildcards in sourcefile names (for example, datafile*), **multiple** must be set to true. Note that you can't use the question mark (?) as a wildcard. Question mark is the command that lists the current legal options. When **multiple** is set to true, BFTP ignores the destination file (dfile) parameter; therefore, you don't need to set this parameter. The default is false.

(continued)

Table 4-6 BFTP Set Commands

Command	What It Does
set port	Sets the port for the source or destination system of the FTP connection. The default is 21 for both source and destination.
set structure	Sets the FTP structure to file, record, or page. The default is file. If you specify page for the structure, BFTP sets the mode to stream, the type to local, and the byte size to 36.
set time	Sets the start time, the starting retry interval, and the maximum number of tries for a transfer. The default time is **now**, the default retry interval is 15 minutes, and the default number of tries is 5. When a job is set to run **now**, there is a 90-second delay before the job starts.
set type	Sets the FTP type and format and byte size parameters. Note that a normal text file is usually ASCII, and a binary file is often the same as an image file. The default is ASCII and nonprint.
set unique	Sets to true or false the request to use the STOU command. If the STOU command is supported by the destination host, the file will be stored into a file having a unique filename. The default is false. Setting this command to true turns off the **set append** command.
set verbose	Sets to true or false the request to display full FTP conversations for the **verify** and **transfer** commands. The default is false. Transfers run by the **submit** command always run as if **verbose** is true.

(concluded)

All of the commands above are used to set parameters. Some of them are quite simple to use. For example, you can set the copy, delete, and move parameters by entering the **set** command along with the parameter and pressing the New Line key. Other commands, however, require you to enter additional values. BFTP prompts you for the additional values. An example follows.

```
BFTP> set mode ↲ (stream/block/compress) block ↲
BFTP>
```

The above example sets the FTP mode parameter to block. Note that BFTP prompts you with the modes possible; you only have to choose the one you want.

The parameter values for other set commands are either on or off, determined by the values true and false. True turns the value on; false turns it off. When you are using them, BFTP prompts you with the following string: (t/f). An example of how to use these commands follows.

```
BFTP> set append↵ (t/f) true↵
BFTP>
```

This example sets the **append** parameter to true. When a file exists on the destination host with the same name as a file transferred with BFTP, the contents of the file being transferred are appended to the existing file.

NOTE: Setting the **set append** command to true turns off the **set unique** command.

Another type of set commands prompts you to identify whether you want to set the transfer parameter for the source or destination host. These commands prompt with the following string: (s/d). An example of how to use these commands follows.

```
BFTP> account ↵ (s/d) source ↵ doc ↵
BFTP>
```

This example sets the account for the source host to **doc**.

A Sample BFTP User Session

This section contains a complete BFTP session. It shows how to invoke BFTP from the shell, use the **prompt** command to set up transfer parameters, use the **submit** command to log a request, and exit BFTP.

```
% bftp

Background File Transfer: Version 2.01 5/31/89, with extensions

For help, type 'help', or 'explain'.

Special editing characters are as follows:

   <return>  Accept current command/field.
   <escape>  Complete current command/field, or display default.
   <space>   Complete and delimit current command.
   <delete>  Erase last character.
   control-L Refresh screen.
   control-R Refresh line.
   control-U Erase line.
   control-W Erase current token.
   ?         List legal options.
```

```
BFTP> prompt

copy/move/delete: copy
ascii/ebcdic/image/local: ascii (format) nonprint

Source --
 Host: toni58
 User: wilson
 Password:
 Dir: /usr/writes/books/
 File: sula

Destination --
 Host: maya49
 User: wilson
 Password:
 Dir: /usr/writes/poems/
 File: caged_bird

Request type: COPY

Source --
    Host: 'toni58'
    User: 'wilson'
    Pass: SET
    Acct:  ''
    Dir:  '/usr/writes/books/'
    File: 'sula'
    Port: 21

Destination --
    Host: 'maya49'
    User: 'wilson'
    Pass: SET
    Acct:  ''
    Dir:  '/usr/writes/poems/'
    File: 'myfile'
    Port: 21

Structured: file, Mode:stream, Type: ascii, Format: nonprint
Multiple matching: FALSE, Return  mailbox: ''
Remaining tries: 5, Current retry interval: 15 minutes

Starts immediately.  Verbose mode = OFF

BFTP> submit

Checking parameters...


StartTime: 9:00 am
ReturnMailbox: wilson@toni58
RequestPassword:
```

```
Request bftp636648155 submitted to run at Mar 13 09:01:30 1990 EST
BFTP> quit
bye
%
```

Sample BFTP Notification Message

This section contains a sample BFTP notification message. You would receive this kind of message in your mailbox after a successful BFTP transfer. This message details the entire process of transferring a file, including the login processes, opening of the data connections, and the actual file transfer.

```
From croot Tue Mar  6 13:16 EST 1990
Received: from toni58 (toni58.ARPA) by toni58
(1.00/4.7)
                      id AA20209; Tue, 6 Mar 90 13:16:09 est
Received: by toni58 (4.20/4.7)
                      id AA00804; Tue, 6 Mar 90 13:16:27 est
Date: Tue, 6 Mar 90 13:16:27 est
From: wilson@toni58 (Henry Wilson)
Message-Id: <9003061816.AA00804@toni58>
To: wilson@toni58
Subject: BFTP Results: caged_bird -- Succeeded
Status: RO

    Request type: COPY
    Source: toni58,wilson,XXX,,21,/usr/writes/books/,sula
    Destination: maya49,wilson,XXX,,21,/usr/writes/poems/,caged_bird
    Stru: F, Mode: S, Type: A N, Creation: STOR
    Multiple matching: FALSE, Return mailbox: 'wilson@toni58'
    Remaining tries: 5, Current retry interval: 15 minutes


Request bftp636747209 submitted to run at Mar  6 13:16:30 1990 EST.

  Mar  6 13:16:01 1990 EST: starting...

    Request type: COPY
    Source: toni58,wilson,XXX,,21,/usr/writes/books/,sula
    Destination: maya49,wilson,XXX,,21,/usr/writes/poems/,caged_bird
    Stru: F, Mode: S, Type: A N, Creation: STOR
    Multiple matching: FALSE, Return mailbox: 'wilson@toni58'
    Remaining tries: 4, Current retry interval: 15 minutes

Connect to: toni58, 21
  toni58 ==> 220 toni58 FTP server (DG/UX TCP/IP Release 4.30 May 90) ready.
Connect to: maya49, 21
  maya49 ==> 220 maya49 FTP server (DG/UX TCP/IP Release 4.30 May 90) ready.
Logging in: wilson, on toni58
```

```
  toni58 <== USER wilson
  toni58 ==> 331 Password required for wilson.
  toni58 <== PASS XXX
  toni58 ==> 230-User wilson: working directory set to /udd/doc/wilson.
  toni58 ==> 230 User wilson logged in. No account needed.
Logging in: wilson, on maya49
  maya49 <== USER wilson
  maya49 ==> 331 Password required for wilson.
  maya49 <== PASS XXX
  maya49 ==> 230-User wilson: working directory set to /udd/doc/wilson.
  maya49 ==> 230 User wilson logged in. No account needed.
  toni58 <== CWD /usr/writes/books/
  toni58 ==> 200 CWD command okay.
  maya49 <== CWD /usr/writes/poems/
  maya49 ==> 200 CWD command okay.
Using '/udd/doc/wilson/bftp636747209.list' list-file
  toni58 <== PASV
  toni58 ==> 227 Entering Passive Mode (128,223,2,20,4,18)
  maya49 <== PORT 128,223,2,20,4,18
  maya49 ==> 200 PORT command okay.
Transferring sula.
  maya49 <== STOR caged_bird
  toni58 <== RETR sula
  toni58 ==> 150 Opening data connection for sula (128.223.2.15,20)
(7927 bytes).
  toni58 ==> 226 Transfer complete.
  maya49 ==> 150 Opening data connection for caged_bird (128.223.2.20,1042).
  maya49 ==> 226 Transfer complete.
  toni58 <== QUIT
  maya49 <== QUIT

  Mar  6 13:16:25 1990 EST: completed successfully.
```

How BFTP Transfers Files

BFTP uses the "third party" or "Server-Server" model incorporated in the Internet File Transfer Protocol (FTP). In this model, the FTC daemon opens FTP control connections to the existing FTP servers on the source and destination hosts and instructs them to transfer the specified file(s) from source to destination host. The source and destination hosts can be any two internet hosts supporting FTP servers (but at least one of them must support the FTP PASV command). The PASV command requests that the server data transfer process (server-DTP) listen on a data port that is not its default data port and wait for a connection. This approach allows the implementation of a background file transfer capability between diverse machines by using existing facilities.

NOTE: The PASV command is officially listed as an optional command. While many FTP server implementations do support this command, some (in particular, the 4.2BSD FTP) do not. Some implementations claim to support PASV, but do not do so correctly. This generally results in a `Connection refused` error when BFTP attempts to create a data connection.

An ECLIPSE® system or an FTP daemon running the 4.10 release of the DG/UX system cannot be a source host. These daemons use a nonstandard PASV response format. These systems can be a destination host if the source host supports the PASV command.

Figure 4-1 illustrates the BFTP model of operation. It traces the steps of a file transfer from the local user through the BFTP user interface through the FTC daemon to the source and destination hosts. This model also shows the additional steps necessary for a remote user (a user on a system not running BFTP) to use BFTP. The remote user uses TELNET to log in to a system running BFTP.

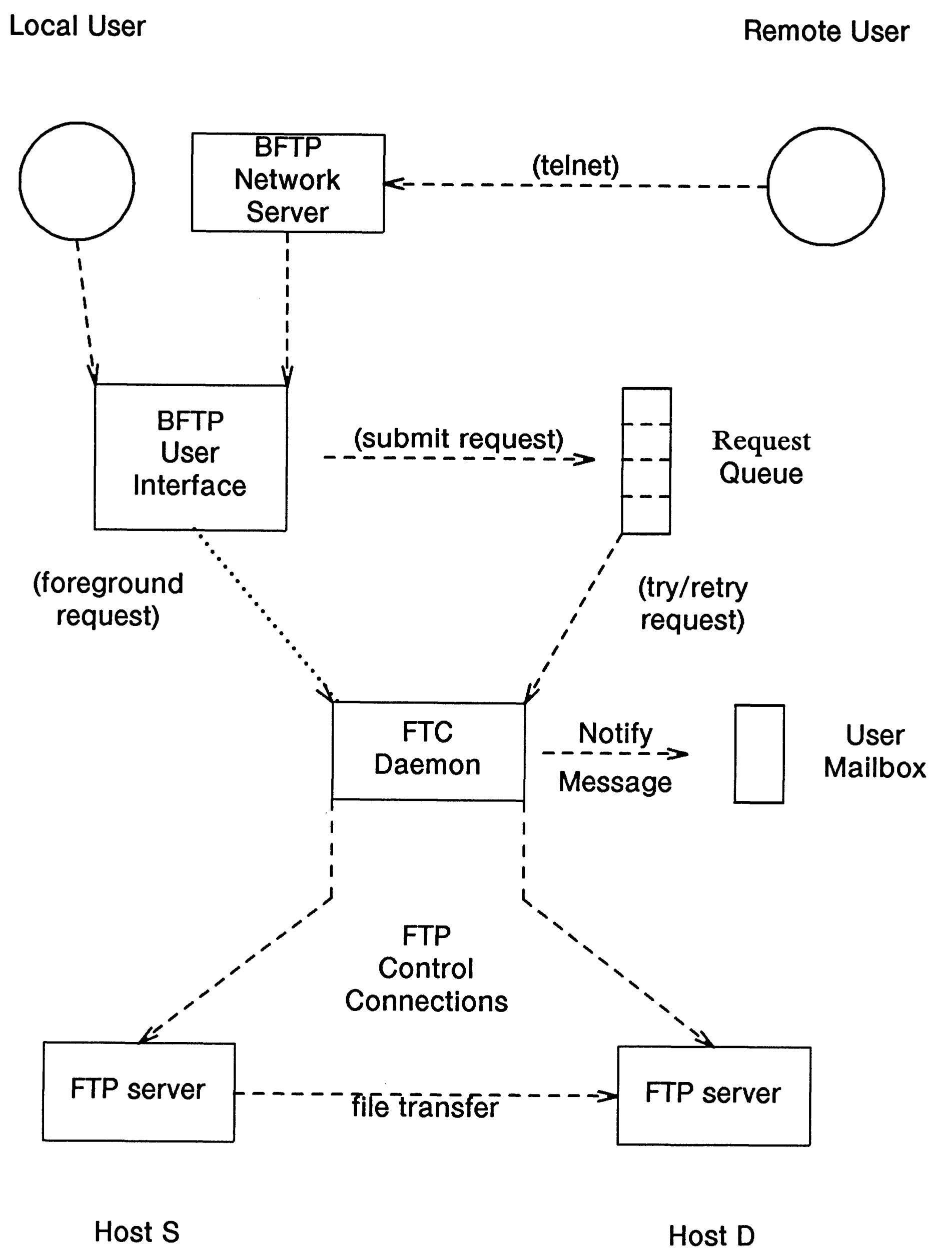

Figure 4-1 *Server-Server File Transfer*

Figure 4-2 illustrates the FTP command interchange used in a typical Server-Server file transfer operation. BFTP currently uses the following Server-FTP commands: USER, PASS, ACCT, PASV, PORT, RETR, STOR, STOU, CWD, NLST, MODE, STRU, TYPE, and QUIT. BFTP may also use some Data General extensions to support the **hold**, **unhold**, and **cancel** commands. For more information on FTP server commands, see Chapter 3, "Understanding and Using the File Transfer Protocol."

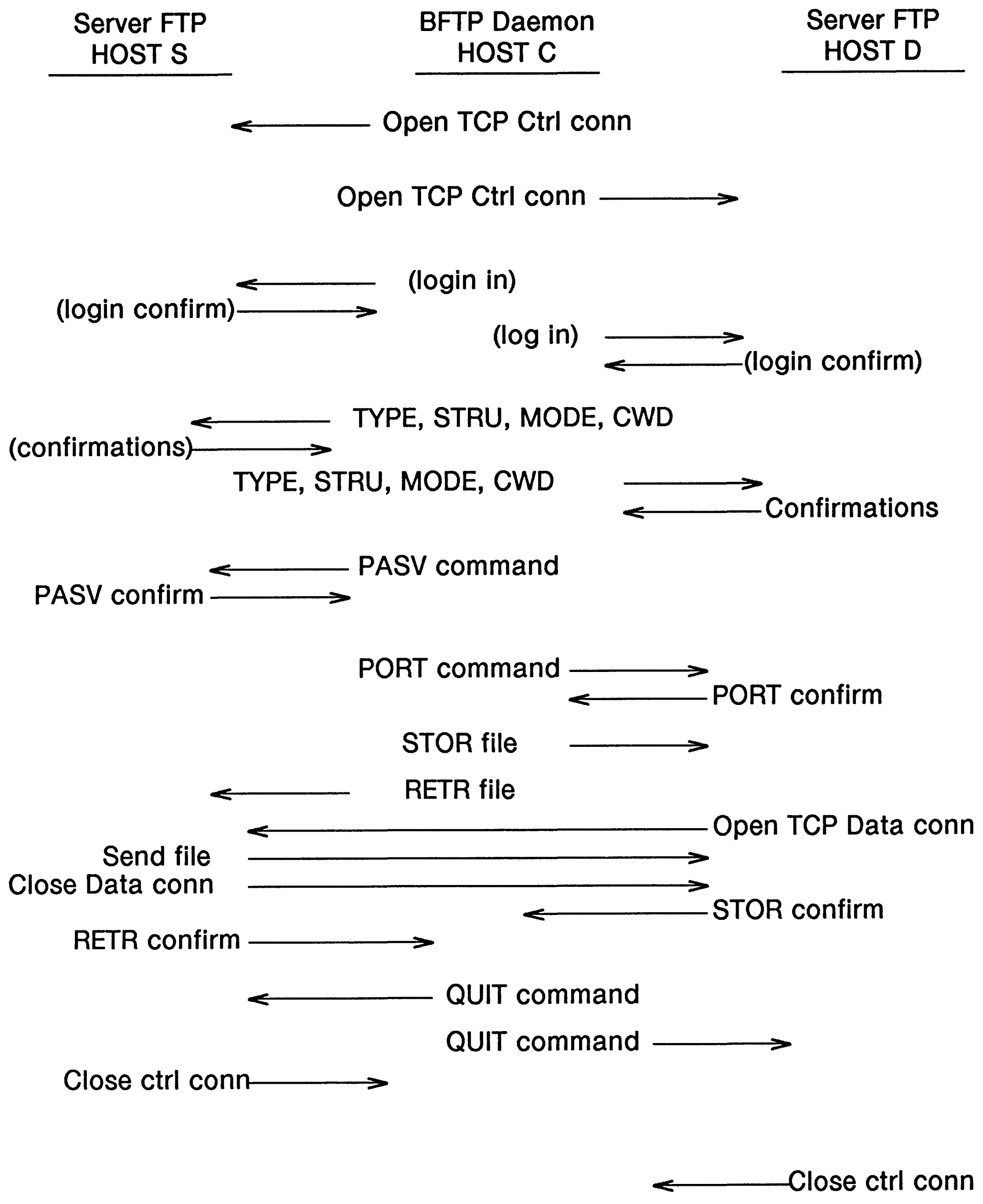

Figure 4-2 *FTP Command Interchange*

The FTC daemon attempts to work around FTP servers that do not support commands that are considered optional. For example, if a server does not support the optional command CWD, the FTC daemon attempts to construct a complete pathname using the source directory name and the source filename. However, at least one of the two hosts must support the FTP passive (PASV) command.

Because BFTP could be asked to transfer files between any two hosts in the internet, it understands all the types, structures, type, and mode parameters defined for FTP (see RFC-959). However, the ability to actually perform a transfer with the chosen settings also requires support by the underlying **ftpd** processes.

BFTP supports the transfer of a set of files in a single request, using the standard technique:

1. Send an NLST command to the source host, specifying a pathname containing "wildcard" characters that use a syntax (for example, an asterisk (*)) understood by that host. The reply will contain a list of matching source filenames.

2. Execute a separate transfer operation for each file in this list. The destination filename in each case is assumed to be the same as the source filename; this requires that these names be compatible with the naming conventions of the destination host.

It is typically necessary to specify working directories for the transfers at the source and destination hosts, so the filenames will be simple, unstructured names on each system. This approach depends upon the wildcard matching capability of the source host FTP daemon.

Understanding the Reliable Delivery Function

BFTP's reliable delivery function is analogous to reliable delivery in a transport protocol like TCP. BFTP, like TCP, repeatedly tries to deliver files until it succeeds. In both cases, determining the interval between tries requires a careful balance between overhead and responsiveness.

BFTP uses the Ethernet's exponential backoff algorithm to determine the interval between tries. That is, BFTP's FTC daemon starts each transfer request with a short retry interval (for example, 15 minutes) and then doubles this interval for successive retries until it reaches a maximum interval (for example, 4 hours). This algorithm is also used by transport protocols such as TCP.

Multiple transfer requests require a clear definition of the meaning of reliable transmission. For example, because files can be created or deleted while a transfer request is pending, the set of files selected by wildcard characters in a pathname when a transfer is first submitted can be different from the set of files selected at the actual time of the transfer. Similarly, if a file transfer fails before the entire set of files has been transferred, BFTP must be able to determine which files were transferred and which were not. BFTP, therefore, must be able to determine what it considers to be the stable list of files to be transferred and keep track of which files have been transferred during each attempt.

BFTP addresses these issues as follows:

- For a multiple file operation, the FTC daemon saves the file name list returned by the first successful NLST command in the request queue entry. This name list determines the set of source files for the transfer; there can be no later additions to the set.

- The FTC daemon maintains a transfer status pointer. On each retry cycle, it tries to transfer only those files that have not already been successfully transferred.
- The request is complete when all the individual file transfers have been successful, a permanent failure has occurred, or when the retry limit is reached.
- The notification message to the user lists the status of each of the multiple files.

End of Chapter

Chapter 5 Understanding and Using the Trivial File Transfer Protocol

This chapter describes how to use the Trivial File Transfer Protocol (TFTP). TFTP allows you to transfer files back and forth across a network.

TFTP is similar to FTP. It is, however, a much simpler program than FTP and uses the User Datagram Protocol (UDP) to communicate with remote systems rather than the Transmission Control Protocol (TCP). TFTP does not let you display a remote directory or do other kinds of file and directory manipulations. In general, you should use this protocol when communicating with a remote host that does not support FTP.

TFTP does not require you to have an account or password on the remote system to transfer files. File access is determined by the "others" section of file permission. You can transfer only files that are publicly accessible.

Understanding TFTP File Transfer

TFTP transfers files between a local TFTP environment and a remote TFTP environment. The local and remote environments consist of the operating system and file structure of the local and remote hosts, respectively.

You identify hosts by hostname or dot-format internet addresses. Identify ports with small integers. If you do not specify a port, TFTP uses a default port.

You are in the TFTP local environment once you execute **tftp** on the local host. You can now enter TFTP commands to move files to and from the local and remote environments. When you do so, the **inetd** on the remote host invokes the TFTP daemon (**tftpd**) that will service requests sent by the TFTP user program. Then, the user program and the daemon open communication to one another.

NOTE: TFTP is case-sensitive. You must type commands in lowercase. TFTP does not recognize commands in uppercase.

Figure 5-1 shows a local and a remote TFTP environment.

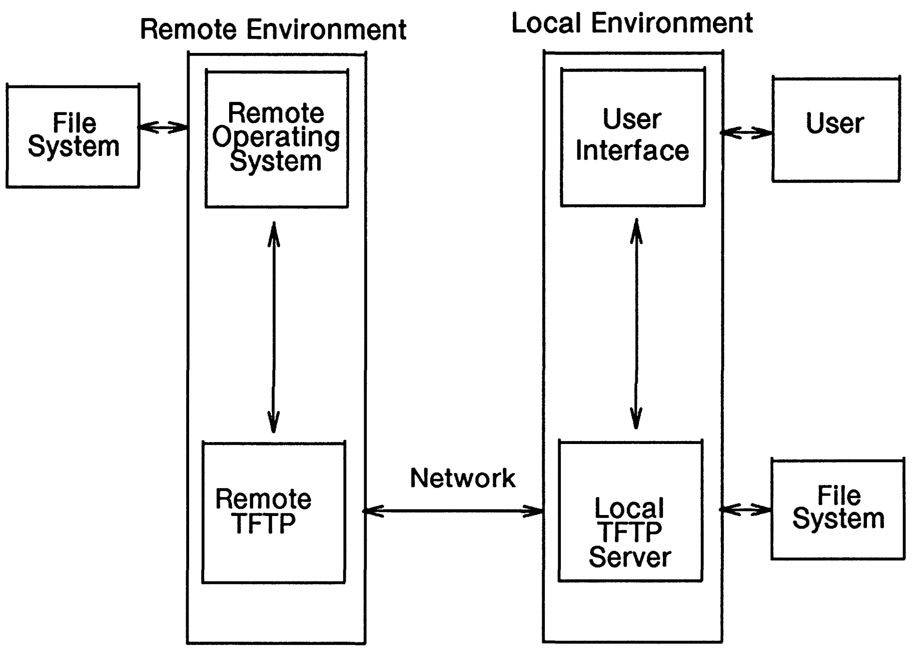

Figure 5-1 *TFTP Local and Remote Environments*

Forming the TFTP Local and Remote Environments

You form the TFTP local and remote environments by creating the TFTP local environment on the local host and entering TFTP commands to form the TFTP remote environment. These steps are described below.

Creating the TFTP Local Environment

The first step to creating the TFTP local environment is to execute the **tftp** command. (TFTP in uppercase letters refers to the protocol, while **tftp** in bold lowercase letters refers to the command.) For example, enter the following:

```
$ tftp ⤶
```

The TFTP program will return the **tftp** prompt to your terminal. You will then be in your TFTP local environment.

Specifying the Remote Environment

Once in your TFTP local environment, you can specify a remote host by entering the **connect** command with either the name of the remote host or the remote host's Internet address as an argument. Optionally, you can specify a port number on which to assign the connection. The default port number for TFTP is specified in **/etc/services**. For example:

```
tftp> connect remote3 ↵
tftp>
```

This command specifies **remote3** as the remote host.

You can combine the two commands that form your local and remote environments by entering **tftp** and the remote host's name or address as an argument on the same command line. By doing this, you do not stop in the TFTP local environment. For example:

```
$ tftp remote3 ↵
tftp>
```

executes the TFTP program and specifies **remote3** as the remote host.

Terminating a TFTP Connection

Terminate TFTP with the **quit** command. For example:

```
tftp> quit ↵
```

terminates TFTP and returns you to the shell.

Understanding TFTP Transfer Parameters

Mode is the only TFTP transfer parameter that controls the transmission of data as the data is transferred.

Table 5-1 describes the mode parameter, the DG/UX command to change it, and what the mode means.

Table 5-1 TFTP Transfer Parameters

Parameter	Command	What It Means
mode	**mode**	TFTP supports two transfer types: **ascii** and **binary**. The **ascii** type is the default and is used to transfer text files from host to host. The **binary** type is used to transfer binary data, such as compiled programs, from host to host.

For example, to set the mode to **binary**, type the following:

```
tftp> mode binary ↵
```

Mode is set to **binary** and the **tftp** prompt returns.

Displaying Information About TFTP

The ? and **status** commands display information about TFTP commands and parameters.

Using the TFTP ? Command

While in your TFTP local environment, you can check the meaning of a TFTP command. For a list of all available commands, enter the TFTP help (?) command. For example, enter:

```
tftp> ? ↵
```

to display all TFTP commands. If you want information about a particular command, enter the ? command with the command in question as an argument. For example, enter:

```
        tftp> ? get ↵
```

to display the command line syntax and meaning of the TFTP **get** command.

Using the TFTP status Command

The **status** command displays information about your TFTP parameters. For example:

```
        tftp> status ↵
```

will display the following information:

- TFTP connection status
- Mode parameter value
- Verbose mode status
- Tracing mode status
- Rexmt-interval parameter value
- Max-timeout parameter value

Transferring Files in TFTP

TFTP has two commands to transfer files: **get** and **put**. You can use these commands either in your TFTP local or remote environment.

Using the TFTP get Command

The **get** command transfers a file from the remote host to a local file or directory. If you have not named the remote host, then you must specify the host explicitly. You can invoke the **get** command with the following formats:

get Prompts for files. Copies the remote file *file1* from the remote *host* into a file named *file* in the current local directory. If **tftp** is not connected, you must specify the host.

get [*host*:] */rdirpath/file1*

Copies the remote file */rdirpath/file1* from the remote *host* into a file named *file1* in the current local directory. You must specify the host explicitly if **tftp** is not connected. If there is a file in the local directory named *file1*, then the contents of that file are replaced with the contents of the file you are transferring.

get [*host*:]*/rdirpath/file1 /ldirpath/file2*

Copies remote file */rdirpath/file1* into local file */ldirpath/file2*. You must specify the host explicitly if **tftp** is not connected.

get [*host*:]*/rdirpath/file1* ... [*/ldirpath/*]*dir*

Copies the remote file */rdirpath/file1* into the specified local directory *dir*. More than one remote file can be specified; TFTP assumes that the last name given is the appropriately specified local directory that you want the remote files moved to. You must specify the host explicitly if **tftp** is not connected.

In the preceding formats, the following variables are specified.

Variable	Definition
host:	Hostname or dot-format Internet address. If you specify a *host*, the colon character (:) is required.
rdirpath	Remote directory pathname specified from the root directory.
ldirpath	The local directory pathname (can be absolute or relative pathname).

If the local directory is not specified, TFTP assumes you want the current local directory. For example:

```
tftp> get /pdd/owen/test2 ↵
tftp>
```

Copies the file **test2** into a file of the same name in the current directory on the local host.

Using TFTP's put Command

The **put** command transfers a file from the local host to the remote host. If you have not named the remote host, then you must specify *host*: explicitly. Use the **put** command with the following formats:

put

Prompts for files. Copies local file *file1* into a file of the same name in the specified remote directory. If **tftp** is not connected, you must specify the host.

put [*host*:]**/rdirpath/file1**

Copies local file *file1* into a file of the same name in the specified remote directory. If **tftp** is not connected, you must specify the host.

put [*/ldirpath/*]*file1* [*host*:]*/rdirpath/file2*

Copies local file *file1* into remote file */rdirpath/file2*. If **tftp** is not connected, you must specify the host.

put [*/ldirpath/*]*dir/file1...* [*host*:]*/rdirpath*

Copies local file *file1* into the specified remote directory. You can specify several local files; TFTP assumes that the last name given is the appropriately specified remote directory. If **tftp** is not connected, you must specify the host.

In the formats above, the following variables are specified.

Variable	Definition
host:	Hostname or dot-format Internet address.
rdirpath	Remote directory pathname specified from the root directory.
ldirpath	The local directory pathname (can be absolute or relative pathname).

If the local directory is not specified, TFTP assumes you want the current local directory. For example:

```
tftp> put /udd/owen/test3 /pdd1/owen/file1 ⤶

tftp>
```

Copies the local file **test3** into the remote file **file1**.

Using Miscellaneous TFTP Commands

This section describes TFTP commands that either display information about or change parameters that rarely need changing. Unless you have special need for these commands, skip this section. Table 5-2 lists the TFTP commands.

Table 5-2 Miscellaneous TFTP Commands

Command	Function
rexmt	Sets the amount of time in seconds to wait before retransmitting a packet. The default is 5 seconds.
timeout	Sets the amount of time in seconds to wait before giving up on a file transfer. The default is 25 seconds. If your network is slow, you may need to increase the default.
trace	Toggles trace mode on or off. When on, packet transfers are displayed on the screen. When sending or getting files, each packet transfer prints the packet header on the screen. The packet header contains information, such as type of packet and the packet number. Retries show up as multiple lines with the same packet number.
verbose	Toggles verbose mode on or off. Verbose mode has no meaning in the DG/UX environment.

To change the default values of **rexmt** and **timeout**, you can type the commands with or without arguments. If you type them without arguments, these commands prompt you for a value. For example:

```
tftp> timeout ↵
(value) 10 ↵
tftp>
```

changes the value of **timeout** to 10 seconds. You can, however, reset the value of these commands by specifying the new value as an argument (for example, **timeout 50**).

Understanding Error Messages

Table 5-3 describes the error messages you might see when using TFTP.

Table 5-3 TFTP Error Messages

Error	What It Means
`Access violation`	Specified file doesn't exist or can't be accessed.
`?Ambiguous command`	Command abbreviation is not unique.
`?Ambiguous help command`	Argument given to the **?** command is not unique.
`bad port number`	Specified port number does not exist.
`bad value`	Value specified for **rexmt** or **timeout** command is invalid.
`Error code` *code*`:` *message*	Error message from the server.
`Illegal TFTP operation`	Server does not recognize local **tftp** request.
`?Invalid command`	Unknown command.
`File not found`	Specified file does not exist.
`No target machine specified; see connect.`	Host has not been specified.
`tftp:` *system-error-message*	Failed trying to access a local resource (for example, a network or file).
`tftp: udp/tftp: unknown service`	TFTP can't find the file **/etc/services**.
`Transfer timed out`	Allotted time to wait for transfer expired.
`unknown host`	Specified host does not exist.

End of Chapter

Chapter 6
Understanding and Using the Remote Commands

The remote commands, or R commands, allow you to form a connection with one or more remote hosts and use services on those hosts. These commands allow you to log in to, to execute a command on, to copy files, or to receive information from any operating system on the network that is running the R command servers.

R commands consist of program pairs: the R command user process and the R command server (daemon) process. Either or both programs may run on the *local* and *remote* systems. The local host is the system the user's terminal is connected to. The remote host is the operating system whose services are requested. The user and server programs are necessary to connect the two systems and then use the services on a remote host. The user programs initiate a connection to another system.

There are two schemes that may be used to allow connections. The first is referred to as 4.2 BSD compatible and the second is 4.3 BSD compatible. Beginning with the 4.10 release for AViiON Systems, the DG/UX system became (and remains) 4.3 BSD compatible. On 4.2 BSD compatible systems, the server programs (for example, **rshd**) are invoked at the time the system is brought up and run constantly, listening for connections from other systems. On 4.3 BSD compatible systems, the **inetd** server program listens for connections from other systems and invokes a server when a connection is requested. The services that **inetd** provides are determined by the configuration file **/etc/inetd.conf**.

The execution of an R command causes the local user programs to run. Together with the remote server, these commands are the user interface with services on the remote host. Figure 6-1 shows how the R commands function between two hosts.

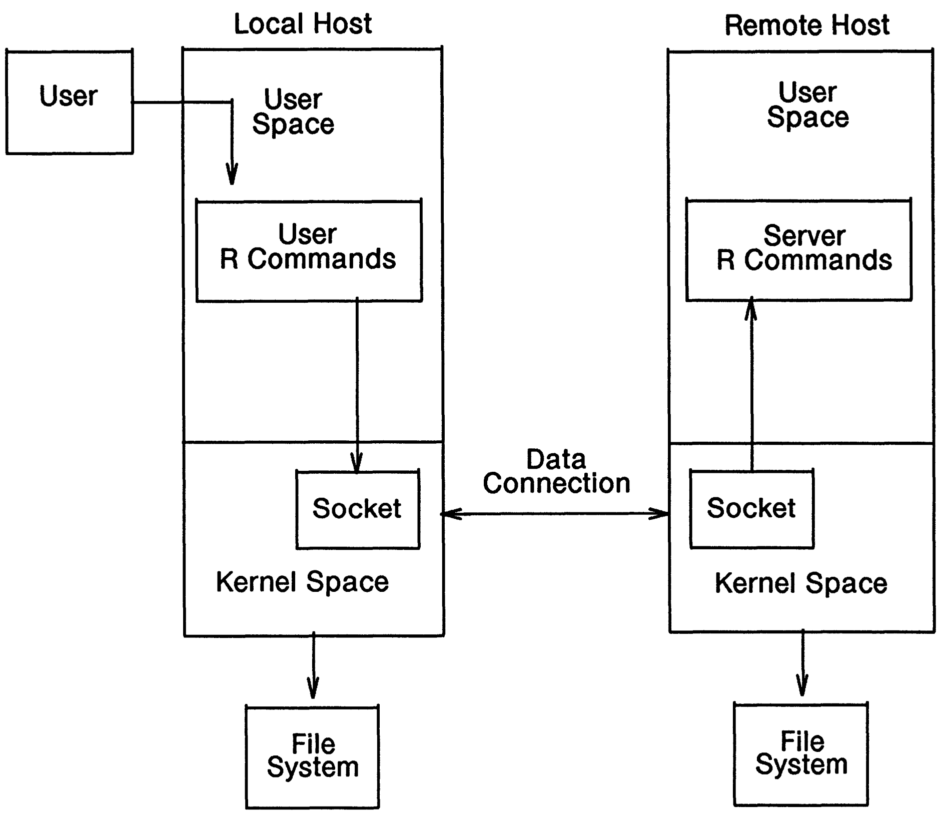

Figure 6-1 *R Commands Service Connection*

Using the R Commands

To use the R commands, the following are required:

- A local host running the R command user program.
- A remote host running the R command server program (or an **inetd** that provides access to the R command server program).
- Access to an account on both the local and remote machines. (Users, however, can run **rwho** and **ruptime** without having an account on the remote system.)

If a user needs remote login privileges, the system administrator must either set up an account on the remote system or set up another person's account to give the user remote login privileges.

A local user can give remote users access to the local account by placing a **.rhosts** file in that user's local home directory. An **.rhosts** file lists the users (and their respective systems) that may have access to the account. A sample **.rhosts** file would have the following format:

```
sys41           owen
sys14           day
sys22           hill
sys12           brown
```

The owner of this file gives the users listed access to his account. The users in this file can use the commands **rsh** and **rcp** to execute a command on or copy files on the owner's system.

Hostnames are given in the file **/etc/hosts**. Each host has one standard name (the first name given in the file) and optionally one or more nicknames or aliases. If this directory is in a user's search path, **rsh** or **rlogin** can be omitted from the command line.

The **/etc/hosts.equiv** file allows users with accounts on two systems to use R commands between systems without creating a **.rhosts** file. A sample **/etc/hosts.equiv** file would have the following format:

```
sys41
sys14
sys22
sys12
```

This file gives all users from the systems listed above that have accounts on the local system access to their accounts. These users must have the same username on both the local and remote systems. Users in this file can use the commands **rsh** and **rcp** to execute a command or copy files on the local system.

The **/etc/hosts.equiv** file can also be used to give individual users access to the R commands between systems. A sample **/etc/hosts.equiv** file providing this function would have the following format:

```
sys41           owen
sys14           day
sys22           hill
sys12           brown
```

This file gives only the specified users access to their accounts through the R commands.

CAUTION: If two different users from foreign systems listed in the **/etc/hosts.equiv** file have the same username, then the two users will have access to each other's accounts.

Summarizing the R Commands

The following is an alphabetical listing of the R commands. For more a detailed description of each of these commands, see the appropriate man page.

rcp	Allows you to copy files between systems on the network.
rlogin	Allows you to log in to another system over the network.
rsh (remsh)	Connects to a specified host and executes a specified command. If you choose SVID compliance during setup of TCP/IP for AViiON™ Systems, the command is **remsh**. If you choose not to comply with the SVID during setup, the command is **rsh**.
rwho	Produces a list of all users logged in to all systems on the local network, as long as the systems are running **rwhod**.
ruptime	Shows the status of each machine that is on the local network and running **rwhod**.

End of Chapter

Appendix A Understanding FTP Server Error Messages

This appendix lists and discusses the error messages that the DG/UX TCP/IP FTP server program returns. All FTP commands generate at least one message, sometimes more than one. These messages can be either reports of success or error messages. All FTP server replies are three digits followed by text. The digits indicate whether the reply is an error or a report of success.

The FTP user program interprets the three-digit number to determine how it should respond. The text message is displayed for the user. Because the message is server-dependent, it may be different for different servers. The numbers, however, should have consistent meaning for all servers.

Each digit in the number provides special information. The first digit indicates whether the response is an error message, a positive acknowledgement, or a request for more information. If the first digit indicates an error message, the second and third digits describe the type of error that occurred.

The DG/UX TCP/IP FTP user program displays the number and text sent from the server program. It recognizes and responds to the first digit only. The first digit can be one of the following:

1*yz*	Indicates a positive preliminary reply. The requested command is being initiated and you should wait for another reply before issuing a new command.
2*yz*	Indicates a positive completion. The requested command has been successfully completed. You can issue a new command.
3*yz*	Indicates a positive intermediate reply. The command has been accepted but is waiting for more information. You should send another command providing the information needed.
4*yz*	Indicates a transient negative completion reply. The command was not accepted, but the error is temporary and you can re-issue the command in exactly the same form.

5*yz*	Indicates a permanent negative completion reply. The command was not accepted and the requested action did not take place. You should not re-issue the command until you correct the problem (for example, correct the spelling of an invalid command).

If the first digit indicates an error, you can check the second and third digits for more specific information. These digits can be helpful if you are not using the DG/UX TCP/IP server. The second digit indicates one of the following categories:

*x***0***z*	Indicates an error in syntax, a command that does not fit any functional category, an unimplemented command, or an unnecessary command.
*x***1***z*	Indicates a reply to a request for information, such as **help** or **status**.
*x***2***z*	Indicates a reply to the command and data connections.
*x***3***z*	Indicates a reply to the login process and accounting procedures.
*x***4***z*	Indicates an unspecified reply.
*x***5***z*	Indicates the status of the server file system as it relates to the requested transfer or other file system action.

The third digit fine tunes the meaning of the category indicated by the second digit. It distinguishes replies grouped in the same category.

Table A-1 lists and describes the error messages that the DG/UX TCP/IP server can return.

Table A-1 FTP Server Error Messages

Error	What It Means
`421 Timeout (`*number* `seconds): closing control connection`	The timeout parameter (**−t** option for ftp server) value is too small. Increase the parameter or set it to 0 – no timeout.
`425 Can't create data socket (`*dest-address*`,` *dest-port*`): relevant-message`	FTP is unable to open data connection for file transfer. Any of the socket related calls could have failed.
`451 Error in server: Out of memory`	The **malloc** function call failed.
`451 Error in server: Unknown state in scanner`	The **yylex** function has detected an unknown token. This indicates bad data on the command connection.
`500 Command not understood`	The wrong argument type or wrong number of arguments has been passed to the server.
`502` *command-name* `command not implemented`	The specified command is not implemented on the remote server.
`502 Invalid TYPE`	Only ASCII, EBCDIC, Image and Local Byte are allowed.
`502 Invalid STRU`	Only File, Record, and Page structures are allowed.
`502 Invalid MODE`	Only Stream, Block, and Compressed modes are allowed.
`503 Bad sequence of commands; RNFR ignored`	The **RNFR** (rename from) server command was not followed by an **RNTO** (rename to) server command.
`503 RNFR failed -- RNTO not accepted`	The **RNTO** (rename to) command is not accepted because the **RNFR** (rename from) command failed.
`503 Login with USER first`	You tried to invoke the PASS command before invoking the USER command.

(continued)

Table A-1 FTP Server Error Messages

Error	What It Means
`504 STAT command does not accept parameters`	Foreign server implementation does not allow any parameters to be sent to **STAT** (status) command.
`530 Login incorrect`	You didn't give a password or you gave an incorrect password.
`530 User` *username* `unknown`	The *username* provided as an argument to **user** command was not found in **/etc/passwd** file.
`530 Please login with USER and PASS`	You tried to access the remote system without executing the login sequence first.
`540 Command` *cmd-name* not accepted during transfer	The FTP server received a command that cannot be executed in the middle of an interrupted data transfer.

(concluded)

Table A-2 FTP Server Error Messages

Error	What it Means
`541 Invalid combination of transfer parameters`	The transfer parameters are set in one of the following combinations: type ascii structure page type ebcdic structure page type binary structure page mode block structure page mode compress structure page
`542 Cannot position file for recovery`	The file position from which the transfer should continue was specified incorrectly.
`543 Error on write`	FTP server detected an error while writing to a file or socket.

(continued)

Table A-2 FTP Server Error Messages

Error	What it Means
`543 Unexpected EOF`	Server found a premature End of File delimiter while reading from the socket.
`543 Wrong escape sequence`	Server found an incorrect escape sequence while receiving a file in Record structure and Stream mode.
`543 Page is not multiple of logical byte`	The page size defined for Page structure is not a multiple of the logical byte size.
`544 EOR not changed, string too long`	The string chosen as the End of Record delimiter for file storage exceeds 10 characters.
`544 Byte size must be multiple of 8`	Logical byte size does not accept values that are not multiples of 8.
`550` *filename*`: not a plain file`	You tried to transfer a character or block device file.
`550` *filename*	Either the file cannot be opened or cannot be accessed.
`550 Can't set guest privileges`	The root directory cannot be changed to your home directory on the remote system.

(concluded)

End of Appendix

Glossary

Address Resolution Protocol (ARP)

A kernel-level protocol used to map an internet address to a physical address (Ethernet address). ARP can be used only across a single physical network and can be used only over networks that support broadcasts over the communications medium.

anchored connection

When peer processes on each end of a connection are anchored, you can transfer data without specifying the peer. In this context, anchored means that there is an actual connection between the processes (that is, one side has issued a **connect** call and the other has issued an **accept** call).

ARPANET

A wide-area network funded by the Defense Advanced Research Projects Agency. The ARPANET served as the basis for early networking research and as the backbone during the development of the Internet network.

association

Binds communicating processes to one another over a network. An association is composed of a local address bound to a local port and a remote address bound to a remote port.

Berkeley Software Distribution (BSD)

A general term for the version of UNIX created at the University of California at Berkeley. When DARPA first made TCP/IP widely available, they decided to encourage university researchers to use the protocols. Most university computer science departments were running BSD UNIX at that time. DARPA funded a company to implement TCP/IP under BSD UNIX and funded UC Berkeley to integrate them with its distribution. BSD UNIX offered more than the basic TCP/IP protocols; it also offered UNIX-like utilities to use the protocols, for example, **rcp**. BSD UNIX also provided the socket abstraction that allows application programmers to access the protocols. Data General's implementation of TCP/IP is based on the BSD implementation.

binding

Assigns a name to a socket so that a process can use the socket to communicate with another process. See the **bind**(2) manual page for details.

broadcast

To send the same message to all systems on a network at the same time.

Carrier Sense Multiple Access with Collision Detection (CSMA/CD)

A characteristic of network hardware that operates by allowing multiple stations to contend for access to a transmission medium by listening to see if it's idle, and which allows the hardware to detect when two stations simultaneously attempt transmission. Ethernet is an example of such a medium.

client

1. An operating system (OS) client.
2. An executing program that sends a request to a server for services and waits for a response. Thus, there are network clients, Yellow Pages clients, Network File System clients, and clients of the domain name system.

client-server relationship

Refers to a pattern of interaction among application programs that communicate over the network. The relationship describes which program initiates a connection, sends data first, and controls the communication link. *See also* client, server.

connection

The path between two processes that provides reliable, stream-oriented, process-to-process delivery service.

connection-based communication

Characteristic of the reliable, stream-oriented, process-to-process service offered by the Transmission Control Protocol. System calls are used to connect to and communicate with remote processes.

connectionless communication

Characteristic of the packet delivery service offered by the Internet Protocol or the User Datagram Protocol. Treats each packet or datagram as a separate entity that contains the source and destination address. Connectionless services may drop or duplicate packets or deliver them out of sequence.

daemon

An unattended background process, often perpetual, that performs a system-wide public function; for example, **inetd**. *See also* server.

DARPA

See Defense Advanced Research Projects Agency.

datagram

A self-contained package of data carrying the necessary information to route itself from source to destination. It is the unit of transmission in the IP protocol. To cross a particular network, a datagram is encapsulated inside a packet.

datagram socket

Sends datagrams and receives datagrams from both directions simultaneously, preserving logical breaks in the data. Data travels in complete packets rather than streams or bytes. The packets may arrive out of order or may fail to be delivered. This service is known as connectionless communication. *See also* socket, User Datagram Protocol.

Defense Advanced Research Projects Agency

The agency that developed the internet protocols (for example, UDP and TCP) for the ARPANET network project.

Defense Data Network

Used loosely to refer to the MILNET, ARPANET, and the TCP/IP protocols that the Defense Data Network uses.

Defense Data Network — Network Information Center (NIC)

The part of the Defense Data Network (DDN) that is the authority who assigns internet addresses.

device driver

A set of software used to manage a peripheral device. For example, **hken** is a device driver used to manage a V/Ethernet 3207 Hawk Local Area Network (LAN) Controller. *See also* controller.

diskless client

An operating system (OS) client with no disks of its own. Even when an OS client has its own local disk, it uses an OS server for its system software (the traditional UNIX **/** and **usr** directories). *See also* OS client.

Ethernet

A type of local area network developed by the Xerox Corporation. An Ethernet network consists of cable and interface hardware that connects hosts. Only one host can use the network at any instant. Hosts send out packets of information over the network whenever they detect that other hosts are not using it.

Ethernet address

A number that identifies a specific host on an Ethernet-based local area network. Ethernet addresses are set on a host during manufacture with hardware switches and are guaranteed to be unique.

file system

See logical disk-file system.

File Transfer Protocol (FTP)

A user-level protocol accessed through the **ftp** command. FTP allows you to transfer files from one host to another. The FTP uses TCP as the transport level protocol.

foreign port

See port, definition 2, remote port.

gateway

A computer (or, in many cases, special-purpose equipment designed for use as a gateway) that converts from one protocol family to another.

handshaking

The exchange of control information between two processes communicating over a network where each process takes turns transmitting data.

heterogeneous OS server

An OS server that provides many system releases to many hosts. Clients share release software and maintain private files the same way clients of homogeneous OS servers do; clients have the ability to boot other releases supported by the OS server if the other releases are compatible with client hardware. *See also* OS server.

host

A computer that is configured to share resources with other computers in a network. Refers to any computer: stand-alone, OS server, or OS client. *See also* local host, remote host.

host-dependent

The commands and files that are dependent or unique to an individual host. If a given file cannot or should not be shared between several hosts, then it is host-dependent. Every client host on the OS server needs its own copy of host-dependent files and needs to be able to write to its own data files. This class of files also contains the set of commands and data files required to boot a host. The **/etc/passwd** file is an example of a host-dependent file. Host-dependent files may or may not be release-dependent.

host ID

A unique number that identifies the host. In the DG/UX system, the host ID is the host's internet address. *See also* host number.

hostname

A string that represents a host. Hostnames are associated with internet addresses in the **/etc/hosts** file.

host number

The host portion of a computer system's internet address. *See also* address class, Internet address.

Internet

The collection of networks and gateways, including the ARPANET and the MILNET, that use the TCP/IP protocol suite and function as a single, cooperative virtual network.

Internet address

A unique 32-bit number that identifies a specific host on the Internet. Internet addresses are expressed in dot notation, and have the general form *a.b.c.d*, where each letter represents eight bits, or an octet. One part of the Internet address represents the network number, and one part represents the host number. There are three classes of Internet address: Class A, Class B, and Class C. The difference among classes depends on the length of the network number: Class A network numbers are one octet long, Class B network numbers are two octets long, and Class C network numbers are three octets long. Network numbers are assigned by the Defense Data Network — Network Information Center, or simply the NIC.

Internet Control Message Protocol (ICMP)

The part of IP that handles error and control messages. Gateways and hosts use ICMP to tell the source of datagrams about problems delivering the datagrams. ICMP also allows a host to test whether a destination is reachable and responding.

interface

A common boundary between two devices, programs, or systems. An interface gives two systems that handle information differently a way to interact.

Internet Protocol (IP)

A kernel-level protocol that defines unreliable, connectionless delivery of datagrams. An IP datagram contains the addresses of its source and destination, and the data transmitted. Connectionless service means that the protocol treats each datagram as a separate entity; the protocol can deliver packets out of sequence, or can drop packets. IP defines the exact format of data as it travels through a network, but delivery of data is not guaranteed.

internetwork

A technology that allows the interconnection of disparate physical networks into a coordinated functional unit. An internetwork (for example, the Internet) accommodates different networking hardware by adding physical connections and by implementing a standard set of protocol conventions.

interoperability

The ability of diverse computing systems to cooperate in solving computational problems.

kernel

The nucleus of the DG/UX system. It controls access to the computer, manages the computer's memory, maintains the file system, and allocates the computer's resources among users. The kernel is sometimes described as the DG/UX system proper; resident code that implements the system calls.

local port

See port.

logical disk-file system

A file system directly associated with a specific logical disk; one logical disk equals one file system.

logical network

A network that may consist of one or more physical networks or may be a subdivision of a single physical network. You set up a logical network through the addressing scheme you use.

mapping

Associating the elements of two different representations of a system (like a directory tree) so that a correspondence exists between the two systems. Every element in one system can be mapped to an element in the other system.

MILNET

Originally part of the ARPANET, the MILNET was partitioned in 1984 to give military installations reliable network service while the ARPANET continues to be a research network. The MILNET uses the same hardware and protocols as the ARPANET.

multiplexing

Using a device to handle several similar but separate operations simultaneously by alternating attention among them.

name server

Part of the domain name system. The name server runs as a daemon process called **named**, and responds to queries by consulting its database. If the answer is not in its database and the name server acts recursively, the name server forwards a query to other name servers. For more information, see *Setting Up and Managing TCP/IP on the DG/UX™ System*.

native release

A release supplied by Data General.

network

The hardware and software that constitute the interconnections between computer systems, permitting electronic communication between the systems and associated peripherals. Networking for computer systems means sending data from one system to another over some medium (such as coaxial cable or phone lines). Common networking services include file transfer, remote login, and remote execution.

Network File System (NFS)

A service that allows many users to share file systems over a network. For more information, see *Managing NFS® and Its Facilities on the DG/UX™ System.*

NIC

See Defense Data Network — Network Information Center.

node name

Name for the system; used as the official name of the machine in a network (typically a UNIX to UNIX Copy (UUCP)-based network). The node name resides in the NODE parameter.

office cluster

A system configured to server as a stand-alone.

Open Network Computing/Network File System (ONC™/NFS®)

A package that consists of the Yellow Pages, NFS, and other networking facilities. For more information, see *Managing NFS® and Its Facilities on the DG/UX™ System.*

OS client

A host that gets its system files from a disk connected to an OS server. *See also* client.

OS release

See release.

OS server

1. A host that is actively sharing resources with another host in an NFS environment.
2. The term also refers to a host that provides disk space for operating system software. OS servers can be stand-alone, homogeneous or heterogeneous.

See also server.

packet

Refers to the unit of data sent across a packet-switching network. The format of a packet is typically defined by the protocol.

peer processes

Processes on different computer systems that run at the same level in the communications hierarchy. That is, both processes run at the user-level or both run at the kernel-level. Peer processes agree through protocol negotiation to exchange data that only their peer can understand.

physical network

The hardware (computers, communication controllers, media) that makes up a network.

port

1. The point of connection between a device (such as a communications controller) and the CPU.
2. The number used to determine which process on a host receives information. Networking software uses ports to allow processes on different computers to communicate. A single process can use several ports, using each to communicate with the port of a different remote process. The *local port* exists on the local host. The *remote port* exists on the remote host.

primary release

The release of software that resides in the server's **/** (root) and **usr** logical disk-file systems. Other (secondary) releases may reside on the OS server in a separate directory space.

process

A program being executed. When a process is being executed by several people simultaneously, there are several processes, but only one program. Each process is cataloged in the system's process table.

protocol

A set of rules that governs the transfer of data and communication between two or more devices in a network. Includes rules for handshaking and the line discipline.

raw socket

Allows access to the underlying communication protocols (such as IP) that support higher level protocols. These sockets normally send information in datagrams, but their characteristics depend on the interface provided by the protocol. *See also* socket.

release

A set of software intended for a specific architecture and operating system. A release encompasses all software required for a host including the release-dependent and host-dependent files.

release-dependent

The system commands and files that are dependent on a release of software. A file that can't or shouldn't be shared between releases is release-dependent. If, for example, manual pages and certain ASCII data files can be shared by more than one release, then they are release-independent. The master files and most commands are examples of release-dependent files. Release-dependent files may or may not be host-dependent.

remote host

The other computer that a local host sends information to and gets information from though connection-based or connectionless communication.

remote port

See port.

Request for Comments (RFC)

A series of technical papers that contain surveys, measurements, techniques, specifications, and proposed and accepted Internet protocol standards. RFCs are available across the Internet.

Reverse Address Resolution Protocol (RARP)

A kernel-level protocol used by a diskless system at startup to find its Internet address. The diskless system broadcasts a request that contains its Ethernet address and the server responds by sending the machine its Internet address.

RFC

See Request for Comments (RFC).

route

The path that network traffic takes from its source to its destination.

sendmail

A command that implements the Simple Mail Transfer Protocol (SMTP), which allows the dispatch of mail messages. The **sendmail** command uses TCP as the transport level protocol.

server

1. An operating system (OS) server.
2. A server process that provides network services to a client process, for example, **telnetd**.
3. A Yellow Pages (YP) server, which provides YP database information to YP clients.
4. A Network File System (NFS) server, which provides file system access to remote NFS clients.

5. A name server for the domain name system (DNS). For details about DNS, see *Setting Up and Managing DG/UX™ TCP/IP.*

servnet

A Data General convention to refer to the collective unit formed by an operating system server, its clients, and its releases. For example, an AViiON computer system supplying releases for two AViiON workstations and one other workstation (for example, a Sun Workstation®) is a servnet.

shell

A command interpreter and programming language that acts as an interface to the UNIX® system. As a command interpreter, the shell accepts commands and acts on them. As a programming language, the shell's features include flow control and string-valued variable definition. When you log in to the system, you acquire a login shell. In this shell, you can run another shell program, which becomes a subshell to your login shell. The two most common shells are the Bourne shell and the C shell. For more information, see *Using the DG/UX™ System.*

socket

A mechanism in the kernel that acts as an interface between processes on different computers. There are three types of sockets available with DG/UX TCP/IP: stream sockets, datagram sockets, and raw sockets.

socket address

In the internet domain, the concatenation of an internet address with a port number. Also called a connection endpoint. *See also* socket.

stand-alone system

A machine with its own disks (typically a mini-computer) that supports dumb terminals in a traditional timeshare environment where all terminals are running the same release. A stand-alone does not need TCP/IP or NFS to service its terminals, and it has no OS clients. *See also* OS server.

stream

A full duplex, processing and data transfer path in the kernel. It implements a connection between a driver in kernel space and a process in user space, providing a general character I/O interface for the user processes.

stream socket

What TCP uses to send and receive data in continuous streams of bytes without logical breaks or duplication. Data can pass through the socket in both directions simultaneously, guaranteeing delivery in the original order in which the data is sent. *See also* socket.

tapeless

An OS server without a tape drive. To read a release tape, a tapeless server must access another host that does have a tape drive.

TELNET

A user-level protocol accessed through the **telnet** command. The TELNET protocol allows a user on one host to interact with a remote host as if the terminal is directly connected to the remote host. TELNET uses TCP as the transport level protocol.

Transmission Control Protocol (TCP)

A kernel-level protocol that defines reliable, end-to-end delivery of datagrams. TCP is connection-based because it establishes a connection between communicating hosts before transmitting data. TCP allows a process on one host to send data to a process on another through a byte stream. TCP uses IP to transmit information along an internet network. TCP messages include a protocol port number that allows the sender to distinguish multiple programs on the remote host.

Trivial File Transfer Protocol (TFTP)

A user-level protocol accessed through the **tftp** command — allows file transfer with minimal capability and overhead. The **tftp** command depends on the UDP protocol.

During first-stage boot with the AViiON station, the boot program, once it determines its internet address, uses TFTP to transfer a file that contains the executable image of a second-stage boot program.

User Datagram Protocol (UDP)

A kernel-level protocol that allows a process on one host to send a datagram to a process on another. UDP is a connectionless transport protocol. UDP messages include a protocol port number that allows the sender to distinguish multiple programs on the remote host.

workstation

A system with its own processor, its own graphics terminal, and graphics software (shared or host-dependent). A workstation could be an OS server, a diskless client, or a client with a disk.

Yellow Pages (YP)

A service that maintains a set of databases about hosts, networks, and services for an entire network. For more information, see *Managing NFS® and Its Facilities on the DG/UX™ System.*

YP domain

1. A named set of YP maps, which are set of keys and associated values.
2. A logical grouping of hosts in a YP environment. Each host in a domain relies on the same server(s) for certain resource sharing and security services. Each domain has one primary and zero or more secondary servers.

For more information, see *Managing NFS® and Its Facilities on the DG/UX™ System.*

YP server

A computer that creates and maintains the following information for hosts in a YP domain: advertised resources, hostnames and passwords, names and addresses for name servers of other domains (optional), host user and group information used for ID mapping (optional). For more information, see *Managing NFS® and Its Facilities on the DG/UX™ System.*

End of Glossary

DG/UX TCP/IP Manual Pages

Here are manual pages for persons who use the DG/UX TCP/IP package. The manual pages describe some DG/UX TCP/IP commands, files, and protocols. These manual pages are also available online through the **man**(1) command.

The following manual pages are included:

Table man-1 List of TCP/IP Manual Pages

Name	Description
bftp(1C)	The command interface to the Background File Transfer Program.
ftp(1C)	The command interface to the File Transfer Protocol.
hostid(1C)	Sets or prints the identifier of the current host system.
hostname(1C)	Sets of prints the name of the current host system.
rcp(1C)	Copies files between hosts.
remsh(1C)	Execute a command on a remote host
rlogin(1C)	Log in to another host on the network.
ruptime(1C)	Show host status of local machines.
rwho(1C)	Show who is logged in to hosts on the local network.
telnet(1C)	Log in to another host on the network.
tftp(1C)	Command interface to the Trivial File Transfer Program.

NAME

bftp – Background File Transfer Program

SYNOPSIS

bftp

DESCRIPTION

bftp is the user interface to the Background File Transfer Program (BFTP). **bftp** may be used to submit a request to have a file transferred at some time in the future via the standard internet File Transfer Protocol (FTP), which is described in RFC-959.

BFTP makes use of third party FTP, so the source and the destination hosts do not have to be operational at the time the request is submitted. At least one of the hosts must correctly support the PASV command of the FTP protocol. Transfers are scheduled locally via the system batch processor, **at**.

For more information on BFTP see *Using TCP/IP on the DG/UX™ System*, *Setting Up and Managing TCP/IP on the DG/UX™ System*, and RFC-1068 (BFTP).

BFTP STANDARD TRANSFER COMMANDS

ddir *directory_name*
: Sets the destination directory. If **ddir** is not set and **dfile** is not a complete pathname, **dfile** will be relative to the user's home directory on the destination host.

dfile *destination-filename*
: Sets the destination filename. Can be a full or a relative pathname. If **ddir** is not set and **dfile** is not a complete pathname, the pathname will be relative to **$HOME** on the destination host.

dhost *destination-hostname user password*
: Sets the destination host, user, and password. If the destination user does not have a password, the password argument is not required.

prompt
: Prompts you for all commonly-used parameters. This combines **shost, sdir, sfile, dhost, ddir, dfile, dhost, ddir, dfile, set type**, and **set copy | move | delete.**

sdir *directory_name*
: Sets the source directory. If **sdir** is not set and **sfile** is not a complete pathname, **sfile** will be relative to the user's home directory on the source host.

sfile *file_name*
: Sets the source filename. Can be a full or a relative pathname. If **sdir** is not set and **sfile** is not a complete pathname, the pathname will be relative to **$HOME** on the source host.

shost *hostname/number user password*
: Sets the source host, user and password. If the source user does not have a password, the password argument is not required.

submit Submits the current request for background FTP transfer. **bftp** will prompt for the *StartTime*, *ReturnMailbox*, and *RequestKeyword*.

transfer
: Perform the current request in the foreground.

BFTP INFORMATION COMMANDS

? List the legal options.

explain
Displays a short explanation of how to use BFTP.

help [*command*] Prints local help information. If a command is supplied as an argument, prints information only on that command.

status Lists the transfers that are currently submitted and provides a summary of each transfer. Use the **find** command for more detailed information on a transfer.

verify Makes the connections necessary to conduct the current transfer, using the specified parameters. Does not make the transfer, but checks the parameters.

show Displays the current parameter values.

BFTP TRANSFER CONTROL COMMANDS

cancel Prevents the specified transfer from taking place. Unlike the **find** command, **cancel** also works after the transfer has begun. This command requires that the source host be running DG/UX 4.30 or higher. To check the version you are running, invoke the **verify** command with **set verbose** set to true.

clear Returns all parameters to their default values.

find Finds and displays the parameters for a transfer request and a log summarizing transfer activity. **bftp** will prompt for the (optional) *RequestID* and the *RequestKeyword*. Once a request has been located and displayed, it can be changed and resubmitted, or cancelled.

hold Suspends a transfer that is currently active (`Running` and not between retries). This may be used to ease congestion on a slow data link between the two hosts. This command requires that the source host be running DG/UX 4.30 or higher. To check the version you are running, invoke the **verify** command with **set verbose** set to true.

quit Returns all parameters to their default values and exits the BFTP program.

unhold Restarts a transfer that has been suspended by the **hold** command. This command requires that the source host be running DG/UX 4.30 or higher. To check the version you are running, invoke the **verify** command with **set verbose** set to true.

BFTP REQUEST COMMANDS

request delete *name*
Deletes request file **bftp-save.***name*.

request list
Lists all request files.

request load *name*
Reads **bftp-save.***name* in as the current request.

request store *name*
Saves the current request in a file named **bftp-save.***name*. Currently, name can consist of numbers and letters only.

BFTP SET COMMANDS

set account

account-name Sets the account for logging in to the source and destination hosts. Many hosts do not require this.

set append true |false

Sets to true or false the request to append transferred file to destination files. If the destination file does not exist, the file is created. The default is false.

set copy

Source file will be copied to the destination filename. Copy is the default.

set delete

Source file will be deleted. Note that when delete is set, no connection is made to the destination host, so only source parameters are required.

set mailbox *mailbox-name*

Sets the mailbox where BFTP transfer results are returned. The mailbox should be in standard internet format, for example: **farah@doc**. The default is *username@host*.

set mode stream | block | compress

Sets the FTP transfer mode to stream, block, or compress. The default mode is stream.

set move

When **set copy** is the default, the source file will be deleted after it has been copied.

set multiple true | false

Sets to true or false the request to transfer multiple files. To use wildcards in sourcefile names (for example, datafile*), **multiple** must be set to true. The default is false.

set port source *n* **| destination** *n*

Sets the port for the source or destination system of FTP connection. The default is 21 for both source and destination.

set structure file | record | page

Sets the FTP structure to file, record, or page. The default is file.

set time *StartTime retry-interval maximum-retries*

Sets the start time, the starting retry interval, and the maximum number of tries for a transfer. The default time is **now**, the default retry interval is 15 minutes, and the default number of tries is 5. Each time that a transfer is retried following a failure, the retry interval is doubled, up to a maximum of 4 hours. You must press the New Line key after *StartTime* because *StartTime* may contain spaces. BFTP prompts you for the retry-interval and maximum number of tries.

set type image | ascii |ebcdic | local

Sets the FTP type and format and byte size parameters. Note that a normal text file is usually **ascii**, and binary file is often the same as an image file. The default is **ascii** and **nonprint**.

The representation type may be one of network ASCII, EBCDIC, image, or local byte size with a specified byte size (for PDP-10's and PDP-20's mostly). The network ASCII and EBCDIC types have a further subtype which specifies whether vertical format control (NEWLINE characters, form feeds, etc.) are to be passed through (nonprint), provided in TELNET format, or provided in ASA carriage control format.

set unique true | false
Sets to true or false the request to use the STOU command. If the STOU command is supported by the destination host, the file will be stored into a file having a unique filename. The default is false.

set verbose true | false
Sets to true or false the request to display full FTP conversations for the **verify** and **transfer** commands. The default is false. Transfers run by the **submit** command always run as if **verbose1 is true.**

SPECIAL EDITTING CHARACTERS

<return>
Accept current command/field.

<escape>
Complete current command/field, or display default.

<space>
Complete and delimit current command/field.

<delete>
Erase last character.

<control-L>
Refresh screen.

<control-R>
Refresh line.

<control-U>
Erase line.

<control-W>
Erase current token.

FILES

bftp creates a number of files that are used to keep track of requests that are in progress:

bftp*123456789***.atjob**
bftp*123456789***.cmd**
bftp*123456789***.list**
bftp*123456789***.msg**
bftp*123456789***.req**
bftp_saved_info

The files that are saved via the **request save** command are as follows:

bftp-save.*request-name*

bftp usually stores its files in the home directory of the user who is logged on. To have **bftp** store these files in another directory, use the system **setenv** command to set **$BFTPDIR**, for example

setenv BFTPDIR *~/var/spool/bftp/yourname*

/etc/bftp.conf, sets the maximum number of simultaneous transfers controlled by this host. This can be used to limit network congestion. No file, or a file containing the value 0 means no limit.

/usr/bin/fts (File Transfer Service) is the program that actually coordinates the transfer. It should only be invoked via BFTP.

SEE ALSO

at(1), **cron**(1M), **crontab**(1M), **ftp**(1C), **ftpd**(1M)

NOTES

Some hosts do not correctly support the FTP PASV command. This may cause a `Malformed PASV reply` or a `Connection refused` error.

Transfers from a DG/UX 4.20 source host may not always complete, depending on the **mode**, **structure**, and **type** selected.

NAME

ftp – use file transfer program

SYNOPSIS

ftp [**−v**] [**−d**] [**−i**] [**−n**] [**−g**] [*host* [*port*]]

DESCRIPTION

The **ftp** program is the user interface to the Internet standard File Transfer Protocol (FTP). The program lets a user transfer files to and from a remote network site.

You may specify the client *host* with which **ftp** is to communicate on the command line. The **ftp** program will then try to establish a connection to an FTP server on that host and enter the command interpreter. Otherwise, **ftp** will enter its command interpreter and await instructions.

OPTIONS

The **−v** (verbose on) option forces **ftp** to show all responses from the remote server, as well as report on data transfer statistics. If **ftp** is invoked from the terminal, the verbose is set to on by default.

The **−n** option keeps **ftp** from attempting auto-login upon initial connection. You must use a user command. If auto-login is enabled, **ftp** will check the **.netrc** file in the user's home directory for an entry listing a login, password, and account for the remote machine. This sample **.netrc** entry

machine remote1 login gerry password fastcar

with the username **gerry** and the password **fastcar** will allow you to auto-login to **remote1**.

If no entry exists in the **.netrc** file, **ftp** will provide as a default the user name associated with the real user ID on the local machine as the user identity on the remote machine. For example, if you had used **su**(1) to become root, **ftp** would provide root as the default name rather than your login name. Then, **ftp** will prompt for a password (if required) and, optionally, will prompt for an account with which to log in.

The **−i** option turns off interactive prompting during multiple file transfers.

The **−d** option enables debugging.

The **−g** option disables filename globbing.

When **ftp** is awaiting commands from the user, it shows a prompt: **ftp>**. If you omit one or more arguments to a command, **ftp** will generally either prompt for the arguments one at a time or print a "help" message that explains the correct way to use the command. The **ftp** program recognizes the following commands:

abort Abort the previous file transfer command. If **abort** is invoked when a data transfer has been interrupted, output from the transfer is aborted. The data connection closes and a reply is sent to the user indicating that the service request terminated abnormally.

account *account-number*
Send an account number for a system logon or access to a specific process. During the login procedure, **ftp** automatically prompts you for your account number if one is needed.

append *local-file* [*remote-file*]
: Append a local file to a file on the remote machine. If *remote-file* is left unspecified, the local filename is used in naming the remote file. If *remote-file* does not exist, it will be created. File transfer uses the current settings for *type*, *mode*, and *structure*.

bell Sound a bell after some of the file transfer commands are completed.

bye Terminate the file transfer session with the remote server and exit **ftp**.

cd *remote-directory*
: Change the working directory on the remote machine to *remote-directory*.

cdup Change the working directory on the remote machine to the parent directory.

close Terminate the file transfer session with the remote server and return to the local command interpreter.

delete *remote-file*
: Delete the file *remote-file* on the remote machine.

debug [*debug-value*]
: Toggle debugging mode. If you specify an optional *debug-value*, it is used to set the debugging level. Setting the debug level to zero turns debugging off; setting it to any other value turns debugging on. When debugging is on, **ftp** prints each command sent to the remote machine, preceded by the string -->.

CAUTION: We cannot guarantee that all commands will function normally in debug mode. Expect some unusual results.

dir [*remote-directory* [*local-file*]]
: Print a listing of the directory contents in the directory *remote-directory* and, optionally, place the output in *local-file*. If no directory is specified, the current working directory on the remote machine is used. If no local file is specified, output comes to the terminal. If the remote directory does not exist, nothing is returned.

disconnect A synonym for **close**.

exit Abruptly terminate the FTP session and exit.

get *remote-file* [*local-file*]
: Retrieve the *remote-file* and store it on the local machine. If the local filename is not specified, it is given the same name it has on the remote machine. The current settings for *type*, *mode*, and *structure* are used while transferring the file.

glob Toggle filename globbing. With filename globbing enabled, each local file or pathname is processed for **csh(1)** metacharacters. These characters include *?[]~{}. Remote files specified in multiple item commands, e.g., **mget**, are globbed by the remote server. With globbing disabled all files and pathnames are treated literally.

hash Toggle hash-sign (#) printing for each data block transferred. The size of a data block is 2048 bytes.

help [*command*]
Print an informative message about the meaning of *command*. If no argument is given, **ftp** prints a list of the known commands.

lcd [*directory*] Change the working directory on the local machine. If no *directory* is specified, the user's home directory is used.

ls [*remote-directory* [*local-file*]]
Print an abbreviated listing of the contents of a directory that is on the remote machine. If *remote-directory* is left unspecified, the current working directory is used. If *local-file* is specified, the listing is put there; otherwise, the output is sent to the terminal.

mdelete *remote-files*
Delete the specified files on the remote machine. If globbing is enabled, the specification of remote files will first be expanded using **ls**.

mdir *remote-files local-file*
Obtain a directory listing of multiple files on the remote machine and place the result in *local-file*.

mget [*remote-files*]
Retrieve the specified files from the remote machine and place them in the current local directory. If globbing is enabled, the specification of remote files will first be expanded using **ls**. If no files are specified, **mget** prompts for them.

mkdir [*directory-name*]
Make a directory on the remote machine. If *directory-name* is not specified, **mkdir** prompts you. If the directory already exists, **mkdir** tells you. It will not overwrite an existing directory.

mls [*remote-files* [*local-file*]]
Obtain an abbreviated listing of multiple files on the remote machine and place the result in *local-file*. If no files are specified, **mls** prompts you. If *local-file* does not exist, it is created.

mode [*mode-name*]
Set the file transfer *mode* to *mode-name*. If you do not specify a *mode-name*, **mode** displays the current mode. Three modes are available: *block*, *compressed*, and *stream*. The following table defines the available modes:

Mode	**Meaning**

stream	Transmits data as a stream of bytes without any restrictions on the type used. This is the default mode.
block	File is transmitted as a series of data blocks, each preceded by three header bytes. You can use record structures and any representation type in this mode.
compress	Sends regular data, compressed data, and control information. Regular data is sent in a byte string, compressed data is sent in replications or fillers, and control information is sent in a two-byte sequence.

mput [*local-files*]
Transfer multiple local files from the current local directory to the current working directory on the remote machine. If you do not specify *local-files*, **mput** prompts you for them. If a file does not exist, **mput** will give you an error message and continue.

open *host* [*port*]
Establish a connection to the specified *host* FTP server. An optional port number may be supplied, in which case **ftp** will try to contact an FTP server at that port. If the **auto-login** option is on (default), **ftp** will also try to automatically log the user in to the FTP server (see above).

prompt Toggle interactive prompting. Interactive prompting occurs during multiple file transfers to let the user selectively retrieve or store files. If prompting is turned off, any **mget** or **mput** will transfer all files without interruption.

put *local-file* [*remote-file*]
Store a local file on the remote machine. If *remote-file* is left unspecified, the local filename is used in naming the remote file. File transfer uses the current settings for *type*, *mode*, and *structure*. If *remote-file* already exists, it is overwritten.

pwd Print the name of the current working directory on the remote machine.

quit A synonym for **bye**.

quote *arg1 arg2* ...
Specified arguments are sent, verbatim, to the remote FTP server. A single FTP reply code is expected in return. This command is usually used for debugging or for working around local restrictions.

recv *remote-file* [*local-file*]
A synonym for **get**.

reinit Terminate the user and reinitialize the command connection. Resets all transfer parameters to their default values. The command connection remains open.

remotehelp [*command-name*]
: Request help from the remote FTP server. If a *command-name* is specified, a more informative message about the *command-name* is given.

rename [*from*] [*to*]
: Rename the file *from* on the remote machine, to the file *to*. If no names are specified, **rename** prompts you for them. If *from* does not exist, an error is reported; if *from* is specified but *to* is not, **rename** shows you a syntax description. If the *to* file already exists, it is overwritten.

restart
: Restart the last transfer aborted by a system crash. The transfer restarts where it was aborted.

CAUTION:
: Only files transferred in **compress** or **block** transfer mode can be restarted. You must use **restart** before you begin any other data transfer.

rmdir *directory-name*
: Delete a directory on the remote machine.

runique
: Toggle the use of unique naming of files transferred from other systems. When on, if a file on the local machine has the same name as the file transferred, a number is appended to the filename of the transferred file. The numbers assigned per transfer run consecutively from 1-99. **runique** is off by default.

send *local-file* [*remote-file*]
: A synonym for **put**.

sendport
: Toggle the use of **port** commands. By default, **ftp** tries to use a **port** command when establishing a connection for each data transfer. If the **port** command fails, the default data port will be used. When **port** commands are disabled, no attempt will be made to use them for each data transfer.

site
: Display information about the remote system. The DG/UX system supplies the following format:

Operating system:	DG/UX
Storage structure:	file
Storage representation type:	ascii
Storage filler:	NULL
Acceptable local byte size:	Multiple of 8 bits
Default page size:	2048 transfer bytes
Default EOR delimiter:	<NL>

status
: Show the current status of the local and remote environments. Displays the current values for the transfer parameters (*mode*, *type*, *format*, and *structure*) and modes (**verbose**, **bell**, **prompt**, **hash**, **globbing**, **sendport**, **runique**, and **sunique**).

struct [*s-name*]

Set the file transfer *structure* to *s-name*. The default structure is *file*. The table below lists the file transfer structures and what they mean:

Structure	**Meaning**
file	There is no internal structure. The file is a continuous sequence of bytes.
page	The file is made up of independent indexed pages.
record	The file is made up of sequential records.

The DG/UX operating system does not support **record** structured files. If you specify **record** structure, all EOR delimiters will be converted to **<NL>** for storage.

page structure will be accepted with only the Local Byte type and is supported only in the stream mode (see **type** command below).

sunique Toggle the use of unique naming of files transferred to other systems. When on, if a file on the remote machine has the same name as the file transferred, a number is appended to the filename of the transferred file. The numbers assigned per transfer run consecutively from 1-99. **sunique** is off by default.

type [*t-name* [*vertical-format*]]

Set the file transfer *type* to *t-name*. If no type is specified, the current type is printed. The default type is network ASCII. If you include the *t-name*, the type is set to *t-name*. *T-name* is the character transfer type.

Some transfer types have formats. Choose the formats by substituting a string for *vertical-format*. If you do not choose a format, the format will be ASCII (the default). See the following table for the transfer types and available vertical-formats:

Type	**Vertical-formats**
ascii	[**no-print** | **telnet** | **carriage-control**]
ebcdic	[**no-print** | **telnet** | **carriage-control**]
binary	

image	
local_byte	[*byte_size*]

NOTE: *byte_size* must be a multiple of 8 bits.

The **binary** transfer type is the same as the **image** transfer type.

The *vertical-format* determines the vertical controls and how the information is represented on a printing device. The default *vertical-format* is *no-print*. The following list defines the *vertical-formats*:

Vertical-format	Description
no-print	The file need not contain vertical format information. A printer process can assume standard values for spacing and margins. Typically, this format is used with files that will be stored or processed.
telnet format controls	The file contains ASCII/EBCDIC vertical format controls, such as <**CR**>, <**LF**>, and <**FF**>, that the printer process can interpret. The sequence <**CRLF**> denotes the end-of-line.
carriage-control	The file contains American National Standards Institute (ANSI) FORTRAN vertical format control characters. If lines and records are formatted according to the ANSI standard, vertical format controls are read in before the data is printed.

user *user-name* [*password* [*account*]]
Identify yourself to the remote FTP server. If the password is not specified and the server requires it, **ftp** will prompt the user for it (after disabling local echo). If an account field is not specified, and the FTP server requires it, the user will be prompted for it. Unless **ftp** is invoked with auto-login disabled, this process is done automatically on initial connection to the FTP server.

verbose Toggle verbose mode. In verbose mode, all responses from the FTP server are displayed to the user. If verbose is on when a file transfer completes, statistics on the efficiency of the transfer are reported. Unless **ftp** is not invoked from the terminal, verbose is on by default.

? [*command*] A synonym for **help**.

! Invoke a shell on the local machine.

Command arguments that have embedded spaces may be enquoted with quote (") marks.

FILE NAMING CONVENTIONS

Files specified as arguments to **ftp** commands are processed according to the following rules:

1) If the filename "-" is specified, **stdin** (for reading) or **stdout** (for writing) is used.

2) If the first character of the filename is a vertical line (|), the rest of the argument is interpreted as a shell command. **Ftp** then forks a shell, using **popen(3)** with the argument supplied, and reads or writes from the **stdout** or **stdin**, respectively. If the shell command includes spaces, the argument must be enquoted; e.g., "| ls -lt".

3) Failing the above checks, if globbing is enabled, local filenames are expanded according to the rules used in **csh(1)**.

FILE TRANSFER PARAMETERS

Several parameters control the transmission and the representation of data as the data is transferred. These transfer parameters are *mode*, *structure*, and *type*. *Mode* defines how the data bits are transferred, while *structure*, and *type* define how the data is represented as it is being transferred. For more information about these parameters, see the commands **mode**, **type**, and **struct** above.

If you want a transferred file to be identical to the original file, make sure the transfer parameters are appropriately set before transferring the file.

INTERRUPTING A FILE TRANSFER

FTP allows you to interrupt a file transfer that is in progress. To interrupt a file transfer, enter the interrupt process character. The interrupt character can be different from system to system (for DG/UX, the interrupt character is usually ^C). The interrupt character suspends the data transfer and displays a menu on the screen. The menu lists the commands available. Unless your first command is the **help** command, you can execute only one of these commands. If you use the **help** command first, you can execute one other command.

The following table lists and explains the available commands.

Command	**Function**
interrupt character	Terminates the FTP-user process.
abort	Aborts data transfer, closes data connection, but leaves command connection open.
quit	Completes data transfer, closes data connection, terminates user, and closes command connection.
reinit	Completes data transfer, terminates user, but leaves the command connection open.
status	Displays status information and continues data transfer.
continue	Continues the transfer.
help [*command*]	Displays available commands or syntax for one of the available commands.

NOTE: If you enter the interrupt character when no data is in transfer, your **ftp** user process will terminate.

SEE ALSO

hosts(5), **init**(3N), **tftp**(1C), **rcp**(1C)

BUGS

Many FTP server implementations that you might connect with do not support experimental operations such as print working directory; they also may not work correctly if data transfer is interrupted.

Errors are not handled consistently, especially in the commands that are preceded by **m**. Before executing a command, check to see that the files you want to transfer exist. Also, after executing a command, check to see that the file transfer was successful.

NAME

hostid – set or print identifier of current host system

SYNOPSIS

hostid [*identifier*]

DESCRIPTION

The **hostid** command (without an argument) prints the identifier of the current host in hexadecimal. This numeric value is expected to be unique across all hosts and is normally set to the host's Internet address. The superuser can set **hostid** by giving a hexadecimal *identifier*; this is usually done in the parameter file **/etc/tcpip.params**.

SEE ALSO

gethostid(2)
sethostid(2)
hostname(1c)

NAME

hostname – set or print name of current host system

SYNOPSIS

hostname [*nameofhost*]

DESCRIPTION

The **hostname** command (without an argument) prints the name of the current host. The superuser can set the *nameofhost* by specifying an argument. The parameter used at boot time is defined in

/etc/tcpip.params

and is used in

/usr/sbin/init.d/rc.tcpipport

SEE ALSO

gethostname(2)
sethostname(2)

NAME

rcp– remote file copy

SYNOPSIS

rcp [**−p**] *filename1 filename2*
rcp [**−pr**] *filename. . .directory*

DESCRIPTION

The **rcp** command copies files between machines. Each *filename* or *directory* argument is either a remote file name of the form:

*hostname***:***path*

or a local file name (containing no **:** characters, or a **/** before any **:** characters).

If a *filename* is not a full path name, it is interpreted relative to your home directory on *hostname*. A *path* on a remote host may be quoted (using \, **"**, or **'**) so that the metacharacters are interpreted remotely.

rcp does not prompt for passwords; your current local user name must exist on *hostname* and allow remote command execution by **rsh**(1).

rcp handles third party copies, where neither source nor target files are on the current machine. Hostnames may also take the form

username@hostname:filename

to use *username* rather than your current local user name as the user name on the remote host. **rcp** also supports Internet domain addressing of the remote host, so that:

username@host.domain:filename

specifies the username to be used, the hostname, and the domain in which that host resides. Filenames that are not full path names will be interpreted relative to the home directory of the user named *username*, on the remote host.

The following options are available:

−p	Attempt to give each copy the same modification times, access times, and modes as the original file.
−r	Copy each subtree rooted at *filename*; in this case the destination must be a directory.

FILES

$HOME/.profile, $HOME/.rhosts, /etc/hosts.equiv.

EXAMPLES

```
 $ rcp sys8:/udd/test1 test2 ⤸
$
```

Copies the remote file **test1** from host sys8 into the file **test2** in your current directory.

```
$ rcp -r sys8:net net2 ⤸
$
```

Copies the contents of the remote directory **net** into the local directory **net2**.

The destination argument (**net2**) must either be a directory or not exist. If **net2** does not exist, a directory with that name will be created.

```
$ rcp wilsonh@sys8:test1 sys9:net/test1 ↵
$
```

Copies **test1**, which is located on the remote machine **sys8**, into the file **test1** on the remote machine **sys9**. The name **wilsonh** represents the user's username on **sys8**.

SEE ALSO

ftp(1), **rlogin**(1), **rsh**(1), **hosts.equiv**(4).

NOTES

rcp is meant to copy between different hosts; using **rcp** to copy a file onto itself, as with:

```
rcp tmp/file myhost:/tmp/file
```

results in a severely corrupted file.

rcp does not detect all cases where the target of a copy might be a file in cases where only a directory should be legal.

rcp can become confused by output generated by commands in a **$HOME/.profile** on the remote host.

rcp requires that the source host have permission to execute commands on the remote host when doing third-party copies.

If you forget to quote metacharacters intended for the remote host you get an incomprehensible error message.

NAME

remsh – remote shell

SYNOPSIS

remsh *host* [**–l** *username*] [**–n**] *command*
host [**–l** *username*] [**–n**] *command*

DESCRIPTION

Use the **remsh** command to connects to the specified *host* and executes the specified *command*. The **remsh** command copies its standard input to the remote command, the standard output of the remote command to its standard output, and the standard error of the remote command to its standard error. Interrupt, quit and terminate signals are passed to the remote command; **remsh** normally terminates when the remote command does.

NOTE: Your system administrator may choose to call this command **rsh** in addition to **remsh**.

The remote username used is the same as your local username, unless you specify a different remote name with the **–l** option. This remote name must be equivalent to the originating account. You will not need to give a password.

You can have a private equivalence list in a file **.rhosts** in your log-in directory. Each line in this file should contain a *remote-hostname* and a *username* separated by a space, indicating the users (and their respective systems) to whom you want to give access to your account.

The **/etc/hosts.equiv** file allows users who have accounts on two systems to use Remote Commands between systems without creating a **.rhosts** file. Each line in the **/etc/hosts.equiv** file should contain a *hostname*. This file gives users from the systems listed and who have accounts on the local system access to their accounts. These users must have the same username on both systems.

CAUTION:
If two different users from foreign systems listed in the **/etc/hosts.equiv** file have the same username, then the two users will have access to each other's accounts.

If you omit *command*, you will be logged in on the remote host using **rlogin**.

Unquoted shell metacharacters are interpreted on the local machine, whereas quoted metacharacters are interpreted on the remote machine. Thus, the command:

remsh *otherhost* **cat** *remotefile* **>>** *localfile*

appends the remote file *remotefile* to the local file *localfile,* whereas:

remsh *otherhost* **cat** *remotefile* **">>"** *otherremotefile*

appends *remotefile* to *otherremotefile*.

Hostnames are specified in the file **/etc/hosts**. Each host has one standard name (the first name given in the file) and one or more optional nicknames.

remsh can be set up to use a favorite remote system by typing only the name of the host. To set up this feature, create a symbolic link in a directory on your search path, named the desired hostname and directed at **/usr/bin remsh**. If you are running **csh**(1), you should then run **rehash** to pick up this new link. For example, assume that you have **/usr/writers** as a directory on your path and **poets** is the name of a remote system you want to log in to. You would make the link as follows: **ln -s /usr/bin/remsh /usr/writers/poets**. This would allow you to log into the remote system **poets** by typing **poets** from the shell.

FILES

/etc/hosts

SEE ALSO

rlogin(1c)

BUGS

If you are using **csh**(1) and put an **remsh** in the background without redirecting its input away from the terminal, the command will block even if no reads are posted by the remote command. If you do not want input, redirect the input of **remsh** to **/dev/null** using the **−n** option.

You cannot run an interactive command (such as **vi**(1)) with **remsh**; use **rlogin**(1C).

Stop signals stop the local **remsh** process only.

NAME

rlogin – remote login

SYNOPSIS

rlogin *rhost* [**−e***c*] [**−l** *username*]

DESCRIPTION

Use the **rlogin** command to log in to another system over the network. The remote system will prompt you for a login and password, as in **login**(1C), unless auto-login is set up.

All echoing takes place on the remote host, so **rlogin** is transparent. Flow control via **^S** and **^Q** occurs on the local machine. To have these flow control characters processed on the remote machine, invoke **rlogin** with the **-8** switch. The flushing of input and output on interrupts are handled properly. A line of the form "~." disconnects from the remote host, where "~" is the escape character. A different escape character may be specified by the **−e** option. Do not type a space between the **-e** option and the new escape character.

You can use **remsh** to streamline the process of logging into remote systems. Although **rlogin** is used to log in to the remote system, you will need to type only the hostname of the remote system, omitting **rlogin** from the command line. To set up this feature, create a symbolic link in a directory on your search path, named the desired hostname, and directed at **/usr/bin remsh**. If you are running **csh**(1), you should then run **rehash** to pick up this new link. For example, assume that you have **/usr/writers** as a directory on your path and **poets** is the name of a remote system you want to log in to. You would make the link as follows: **ln -s /usr/bin/remsh /usr/writers/poets**. This would allow you to log into the remote system **poets** by typing **poets** from the shell.

If you are using **csh**(1), you can suspend a remote login session and return to the shell by using the escape sequence (~) followed by the suspend command. The suspend command is **^z** by default.

Specify a different *username* with the **-l** option. (There must be a space between the **-l** and the *username*.) Use this option when your username on the foreign system is different from your username on the current system.

You can enable auto-login by having a private equivalence list in a file **.rhosts** in your log-in directory. Each line in this file should contain a **remote hostname** and a *username* separated by a space, indicating the users (and their respective systems) to whom you want to give access to your account.

The **/etc/hosts.equiv** file allows users with accounts on two systems to use Remote Commands between systems without creating a **.rhosts** file. Each line in **/etc/hosts.equiv** should contain a *hostname*. This file gives users from the systems listed who have accounts on the local system access to their accounts. These users must have the same username on both systems.

Use the optional **−8** argument to allow an eight-bit data path.

WARNING:

> If two different users from foreign systems listed in the **/etc/hosts.equiv** file have the same username, then the two users will have access to each other's accounts.

The **rlogin** command and **rlogind** server allow for the dynamic exchange of window size information. This is particularly useful in an environment in which you use windowing software such as X windows. Suppose that within a window, you use **rlogin** to log in to a host. If you change that window's dimensions through the mouse, the new dimensions are propagated to the corresponding remote server, **rlogind**. The remote kernel data structures are then changed to reflect these size changes. This information exchange is transparent to a user. For this enhancement to be fully realized, both the local and remote machines must be running the appropriate versions of **rlogin** and **rlogind**.

EXAMPLES

```
    $ rlogin syst3  ↵

    login:jones ↵
      Password:
```

Connects to the remote system **syst3**. The remote system prompts for a username and a password.

```
    $ rlogin syst4 −ep ↵

    login:smith ↵
      Password:
```

Connects to the remote system **syst4**. Changes the escape character to **p**. The remote system prompts for a username and password.

SEE ALSO

remsh(1C)

BUGS

More terminal characteristics should be added.

NAME

ruptime – show host status of local machines

SYNOPSIS

ruptime [**−a**] [**−r**] [**−t** | **−u** | **−l**]

DESCRIPTION

Use the **ruptime**(1C) command to display a status line for each machine that is on the local network and running **rwhod**(1C). These lines are formed from packets broadcast once every three minutes by each host running **rwhod** on the network.

Machines for which no status report has been received for eleven minutes are shown as being down.

Users who are idle an hour or more are not counted unless the **−a** flag is given.

Normally, the listing is sorted alphabetically by hostname. The **−l**, **−r**, **−t**, and **−u** flags specify sorting by load average, reverse sort, uptime, and number of users, respectively.

EXAMPLES

In the following example, the last three columns represent load averages for the intervals 1, 5, and 15 minutes. The load average is the average number of jobs in the run queue. It is a relative indication of how busy the systems are.

```
$ ruptime ↵
 sys14   up   10:46,  4 users, load  0.04,  0.03,  0.04
 sys16 down    1:14
 sys10   up 1+02:11,  1 user,  load  2.40,  2.52,  2.43
 $
```

Shows the host status of the machines on the local area network.

FILES

/var/spool/rwho/whod.*

SEE ALSO

rwho(1C)

NAME

rwho – who's logged in on local machines

SYNOPSIS

rwho [**−a**]

DESCRIPTION

The **rwho** command produces output similar to **who**(1), but for all machines that are on the local network and running **rwhod**(1M). If no report has been received from a machine for eleven minutes, **rwho** assumes the machine is down and provides no information on its users.

If users haven't typed to the system for a minute or more, then **rwho** reports this idle time. However, if users haven't typed to the system for an hour or more, **rwho** doesn't display their status unless you use the **−a** flag.

Command line flags other than **−a** are ignored.

EXAMPLES

```
$ rwho −a ⤸

jones   sys10:tty00  Dec 17 08:07
wilson  sys04:tty03  Dec 17 08:02  2:15
smith   sys08:tty25  Dec 17 07:01
brown   sys02:tty15  Dec 17 08:03   :14
```

Displays users who are logged in on machines that are on the local area network and running **rwhod**, including those who have not typed to the system in an hour or more.

FILES

/var/spool/rwho/whod.*

SEE ALSO

ruptime(1C), **rwhod**(1M)

BUGS

The **rwho** command becomes unwieldy when the number of machines on the local net is large.

NAME

telnet – log in to another host over network

SYNOPSIS

telnet [–i] [–d] [–o] [–bi] [–bo] [–e] [–s] [–l] [host [port]]

DESCRIPTION

Use the **telnet** command to log in to another host using the TELNET protocol. If you invoke **telnet** without arguments, it enters command mode, indicated by its prompt (**telnet>**). In this mode, it accepts and executes the commands listed below. When you invoke **telnet** with arguments, it first checks to see if the arguments match the switches. TELNET then checks to see if the **host** and **port** number are legal. If the arguments are legal, TELNET performs an **open** command (see below) with those arguments.

After a connection has been opened, **telnet** enters input mode. The text you type is sent directly to the remote host. Some special characters, however, are scanned for NVT translation (see *Using TCP/IP on the DG/UX™ System* for details). You cannot invoke TELNET commands in input mode; however, you can use the escape character (initially ^]) to enter local mode. While in local mode, you can invoke a TELNET command. After TELNET executes the command, it returns you to input mode.

The following commands are available. You only need to type the first three letters of any command to uniquely identify it.

bye	Exit from TELNET program.
close	Close a TELNET session and return to TELNET command mode.
crmod	Toggle carriage return mode. When enabled, this mode changes the current setting for **CR** in **stty**(1). If carriage return characters received from the remote host are mapped to **CR**; for example, **crmod** changes the setting to **NL**, where **NL** is a line feed and a carriage return. If the current setting in **stty** is **NL**, **crmod** changes the setting to **CR**.
debug	Toggle debug mode. Also, toggle the ability to use certain commands. With debug on, the following commands are enabled: **listen**, **map**, **send**, and **mode**. These commands let you wait for an incoming connection, change character mappings, and change modes without negotiating options.
CAUTION:	Beware when using the **debug** command. TELNET commands can be unpredictable when used in debug mode.
escape [*string*]	Set the TELNET escape character. Use the same syntax as for strings in the C programming language. If the string is not specified, **telnet** will prompt for it.
listen [*port*]	Listen on a port for an incoming connection. This command works only in debug mode. Use the escape character to abort the connection, or the Interrupt key (often Ctrl-C) if the connection has

not yet been made.

log [*logfile*]	Instruct TELNET to put the data that the remote host sends to your terminal to a *logfile*. Logging starts when you enter **log** with the name of a *logfile*. Logging stops when you enter **log** without an argument. If the *logfile* does not exist, it will be created. If it does, TELNET appends data to the end of the file. Due to buffering, data may not be fully written to the logfile until logging is stopped.
help [*command*]	Get help. With no arguments, **telnet** prints a help summary. If a command is specified, **telnet** will print the help information available about the command only. The ? command is identical to the **help** command.
map *string NVT_char*	Substitute a string of your choice for an NVT character. You can use this command only in debug mode. NVT characters are: IP, AO, AYT, EC, EL, BRK, and EOR (see "Definitions" below for descriptions of these characters).
mode *option type*	Change mode regardless of negotiated option. You can use this command only in debug mode. *Option* can be one of the following: EC, BI, BO, LI, SGA, ST, TM, EX (see below for descriptions of these options). *Type* can be one of the following: on, off, always, never (see "Definitions" below for descriptions of these types).

CAUTION: When using the option types *never* and *always*, you could ask for a particular option that the remote server does not want. In such a case, you can expect unusual results during the connection.

negotiate *option type*	Request negotiation on an option. You can only request a negotiation; you cannot send an announcement of the current mode. TELNET does not notify you that the change has been made or not. Use the command **status** to see the results. *Option* can be one of the following: SGA, EC, BI, BO, ST, TM, EX, LI (see "Definitions" below for descriptions of these options). *Type* can be one of the following: on, off, always, never (see "Definitions" below for descriptions of these types).
open *host* [*port*]	Open a connection to the named host. If no port number is specified, **telnet** will attempt to contact a TELNET server at the default port. The host specification may be either a hostname (see **hosts(4)**) or an Internet address specified in the dot notation (see **inet(3n)**.
options	Toggle viewing of TELNET options processing. When options viewing is enabled, all TELNET

option negotiations will be displayed. Options sent by **telnet** are displayed as SENT, while options received from the TELNET server are displayed as RCVD.

prompt *string*	Substitute a string of your choice in place of the normal command prompt.
quit	Close any open TELNET session and exit **telnet**. (Go back to the shell, or to the program that called **telnet.**)
resume	Exit local mode and continue any suspended remote mode. Returning to remote mode does not automatically produce a shell prompt, refesh the screen, or enter any characters. You may take these actions yourself.
send *NVT_char*	Send NVT special characters across your network connection. You can substitute any of the following for *NVT_char*: Synch, IP, AO, AYT, EC, EL, BRK, or EOR (see "Definitions" below for descriptions of these characters).
shell [*command*]	Create a shell process without terminating TELNET. If you have a network connection, it will remain suspended until you terminate the shell process. Terminate the shell process by entering the **exit** command. The **!** command is identical to the **shell** command.
status	Show the current status of **telnet** parameters. This includes the host you are connected to, as well as the state of debugging.
terminator *string*	Add *string* to the list of terminators. Terminators determine when to ship characters in line mode. The list of terminators includes the default characters (see below) and any you specify. They cannot be the escape character or NVT special characters. The default terminators are: New Line Interrupt character End-of-file character Switch key (if defined) Each character in the string you specify will be a terminator.
un-term *string*	Cancel terminator status for specified characters.
z	Suspend **telnet**. This command works **only** when the user is using DG/UX's **csh(1)**; it interacts with the C shell's job control facilities. When you issue a suspend command, a job number will be returned; you are then placed in the C shell. To return to the

	telnet session, you type **fg job#**. **Job#** is the job number that was returned when you suspended the TELNET session. See **csh(1)** for more information on how the C shell handles job control.
? [*command*]	Get help. With no arguments, **telnet** prints a help summary. If a command is specified, **telnet** will print the help information available about the command only.
! [*command*]	Create a shell process without terminating TELNET. If you have a network connection, it will remain suspended until you terminate the shell process. Terminate the shell process by entering the **exit** command. This command is identical to the **shell** command.

Some of the commands shown above are available as *switches*. Switches allow you to issue commands when you execute **telnet**, without having to enter command mode first. The following table shows the available switches, their corresponding commands, and definitions:

Switch	**Command**	**Definition**
−i*s_type*	Negotiate	Negotiate binary input option.
−o*s_type*	Negotiate	Negotiate binary output option.
−d	Debug mode	Turns debug mode on.
−e*s_type*	Negotiate	Negotiate remote echo option.
−s*s_type*	Negotiate	Negotiate remote side suppress-go-ahead option.
−l[*port*]	Listen	Listen for connections on the given port number.
−bi, −bo		These mode switches send and receive the data as is, with no translation from either side. All control characters are received and not ignored.

The argument *s_type* indicates whether or not you want the option. You must substitute either the letter **a**, for always, or the letter **n**, for never.

For the argument [*port*], you must substitute the port number of the connection you are monitoring. If you do not specify a port number, **telnet** will assign one to you.

DEFINITIONS

This section describes the NVT characters, options, and types that are used with the commands **map**, **mode**, **negotiate**, and **send**. The NVT characters are as follows:

NVT Character	**Meaning**
Synch	A TCP urgent notification with the command data mark (DM).
Interrupt process (IP)	Suspends, interrupts, aborts, or terminates a user process.
Break character (BRK)	Sends the appropriate break character to the remote process.
Abort output (AO)	Allows a process to run to completion, but does not send the output to the user's terminal.
Are you there (AYT)	Provides the user with visible evidence that the system is still up and running.
Erase character (EC)	Deletes the last preceding character or *printed position* the user types. Printed position means several characters that are a result of overstrikes.
Erase line (EL)	Deletes all the data on the current line of input.
End of record (EOR)	Allows the user to flush the input buffer before a terminator character is encountered.

The strings used for *option* in the commands **mode** and **negotiate** are as follows:

Option	**Meaning**
SGA	Suppress go-aheads
EC	Foreign echoing (for **negotiate**)
EC	Local echoing (for **mode**)
BI	Binary input
BO	Binary output
ST	Status
TM	Timing mark
EX	Extended option
LI	Line mode

When using the command **mode**, substitute one of the following strings for *type*:

Type	**Function**
on	Turns on the mode regardless of the option that normally controls it. However, if the option is negotiated after your change, the mode changes to correspond with the change in the option.
off	Turns off the mode regardless of the option that normally controls it. However, if the option is negotiated after your change, the mode changes to correspond with the change in the option.
always	Turns the mode on and leaves it on regardless of the option. Mode is changed when you invoke the change or when you close the connection. When you close the connection, the mode returns to the default setting.
never	Turns the mode off and leaves it off regardless of the option. Mode changes when you invoke the change or when you close the connection. When you close the connection, the mode returns to the default setting.

When using the command **negotiate**, substitute one of the following strings for *type*:

String	**Function**
on	Try to negotiate an option **on**. All future requests to negotiate the option **off** will be honored.
off	Try to negotiate an option **off**. All future requests to negotiate the option **on** will be honored.
always	Try to negotiate an option **on**. All future requests from the server program to negotiate the option **off** will be honored. However, the user program immediately sends one request to the server to negotiate the option back **on**.
never	Demands an option to be negotiated **off** and left **off**. All future requests to negotiate it **on** will be refused. You must know whether or not the server will abort when that option is refused.

SEE ALSO

hosts(4), **inet**(3N), **rlogin**(1), **telnetd**(1M)

NAME

tftp – DARPA trivial file transfer protocol for DG/UX

SYNOPSIS

tftp [**host** [**port**]]

DESCRIPTION

The **tftp** program is the user interface to a very simple network file transfer protocol. The program lets a user transfer files to and from a remote network site.

This is a much simpler program than **ftp(1c)**. It does not let you display a remote directory, invoke a shell, or do other kinds of file and directory manipulation. Generally, you would use it only when communicating with a remote host that does not support **ftp**.

A **host** is identified by **hostname** or dot-format Internet addresses. A **port** is identified by small integers. If **port** is not specified (the usual case), a default port is assumed. When invoked, **tftp** displays a prompt, **tftp>**. You may then issue any of the following commands:

?
: Help. Displays the **tftp** command list.

connect [host [port]]
: Identifies a remote host with which to communicate. **Host** is a **hostname** or a dot-format Internet address. **Port** is an integer.

quit
: Terminates **tftp**.

mode [*name*]
: Sets the file transfer mode. File transfers (**get**, **put**) made after the mode is set will be performed in that mode. *Name* is one of the following:

  ascii transfers standard ASCII text files.

  binary transfers binary files, such as compiled programs.

  mail Sends files as mail to a user, rather than to a file. (Not implemented.)

get
: Transfers a file from the remote host to a local file or directory. If **connect** has not been issued for the remote host desired, then **host:** must be specified explicitly. It is used much like **cp(1)**.

  get, executed without an argument, will prompt for both filenames.

  get [*host*:]*/rdirpath/file* copies remote file *file* from the remote host into a file named *file* in the current local directory. It will overwrite an existing file of the same name.

  get [*host*:]*/rdirpath/file1* [*/ldirpath/*]*file2* copies remote file *file1* into local file *file2*.

get [*host*:]**/***rdirpath***/***file* **...** [**/***ldirpath***/**]*dir*" copies the remote file *file* into the specified local directory. More than one remote file can be specified; **tftp** assumes that the last name given is the appropriately specified local directory that you want the remote files moved to.

All *rdirpath*s must be absolute pathnames, i.e., specified all the way from the root directory **/** on the remote host. *Ldirpath*s may be absolute or relative pathnames (if not specified, the current local directory is assumed). All **hosts** are specified by **hostname** or dot-format Internet addresses.

put

Transfers a file from the local host to the remote host. If **connect** has not been issued for the remote host desired, then **host:** must be specified explicitly. Otherwise, it is used much like **cp(1)**.

put, executed without an argument, returns a help message showing you how to use it.

put [*host*:]**/***rdirpath file* copies local file *file* to the remote host into a file of the same name in the specified remote directory. It will overwrite an existing file of the same name.

put [**/***ldirpath***/**]*file1* [*host*:]**/***rdirpath***/***file2* copies local file *file1* into remote file *file2*.

put [**/***ldirpath***/**]*file* [**host:**]**/***rdirpath* copies local file *file* into the specified remote directory. More than one local file can be specified; **tftp** assumes that the last name given is an appropriately specified remote directory.

All *rdirpath*s must be absolute pathnames, i.e., specified all the way from the root directory **/**on the remote host. *Ldirpath*s may be absolute or relative pathnames (if not specified, the local current working directory is assumed). **Host**s are specified by **hostname** or by dot-format Internet addresses.

rexmt

Sets the amount of time in seconds to wait before a retry is sent. Default is 5 seconds. You might want to increase the amount of time if the network is very slow.

status

Displays the settings for **host, mode, trace, verbose, rexmt,** and **timeout**.

timeout

Sets the amount of time in seconds to wait before giving up on a file transfer. Default is 25 seconds; you might want to increase this number if the network is very slow.

trace

Turns on trace mode. When on, packet transfers are displayed on the screen. During **put** and **get,** each packet transfer prints the packet header on the screen. The packet header contains information such as type of packet and the packet number. Retries show up as multiple lines with the same packet number.

verbose

Turns on verbose mode. Verbose mode has no meaning in DG/UX; nonetheless, it is visible to **status**.

This protocol trades flexibility for absolute simplicity. It uses a reliable, lock-step packet mechanism. Security depends on file permissions and how much outsiders know about your directories and files. Remote users are governed by the "others," or **o** section of a file permission.

ERRORS

Most errors terminate the command, including:

* file not found,
* user not found,
* access violation (you don't have access to a directory to which you tried to send a file),
* internal errors (a server receives a badly formed packet).

SEE ALSO

chmod(1)
chown(1)
cp(1)
ftp(1C)
hosts(5)
inet(3N)
ftpd(1M)
tftpd(1M)

End of Chapter

Index

Note: Boldfaced page numbers (e.g., **1-5**) indicate definitions of terms or other key information.

Documentation Set

This section lists documents relevant to the AViiON product line. The titles of Data General manuals are followed by nine-digit numbers used for ordering; you can order any of these manuals via mail or telephone (see the TIPS Order Form in the back of this manual).

Following the list of Data General manuals are relevant documents published by other organizations (no ordering information is provided).

Documents specifically referred to in the text of this manual are also listed in the "Related Documents" section of the Preface.

Data General Software Manuals

88open Binary Compatibility Standard (069-701043)

Specifies a Binary Compatibility Standard (BCS).

88open Object Compatibility Standard (069-701044)

Specifies an Object Compatibility Standard (OCS) for operating systems based on Motorola MC88100 as well as future related microprocessors. Provides for portability of application-level software at the linkable level by specifying interfaces between the object file and the operating system libraries.

C: A Reference Manual (069-100226)

Describes lexical structure, the preprocessor, declarations, types, expressions, statements, functions, programs, and the run-time libraries.

Documenter's Tool Kit Technical Summary for the DG/UX™ System (069-701041)

Provides technical details about the tools supplied with the Documenter's Tool Kit; specifically, the **mm** macroinstruction package, the **tbl** text processor, and the nroff/troff formatter.

Green Hills Software User's Manual C-88000 (069-100230)

Describes the differences in the C programming language when run on an 88000 system.

Green Hills Software User's Manual Fortran-88000 (069-100232)

Describes differences in the FORTRAN programming language when run on an 88000 system.

Green Hills Software User's Manual Pascal-88000 (069-100231)

Describes differences in the Pascal programming language when run on an 88000 system.

IEEE Standard Portable Operating System Interface for Computer Environments (POSIX.1) (069-701045)

Specifies a POSIX standard.

Installing and Managing the DG/UX™ System (093-701052)

Shows how to install and manage the DG/UX operating system on AViiON hosts that will run as stand-alone, server, or client systems. Aimed at system administrators who are familiar with the UNIX operating system.

Learning the UNIX® Operating System (069-701042)

Helps beginners learn UNIX fundamentals through a step-by-step tutorial. (UNIX is a U.S. registered trademark of American Telephone and Telegraph Company.)

Managing NFS® and Its Facilities on the DG/UX™ System (093-701049)

Shows how to install, manage, and use the DG/UX ONC™/NFS product. Contains information on the Network File System (NFS), the Yellow Pages (YP), Remote Procedure Calls (RPC), and External Data Representation (XDR). (NFS is a U.S. registered trademark of Sun Microsystems, Inc. ONC is a trademark of Sun Microsystems, Inc.)

OSF/Motif™ Application Environment Specification (069-100326)

Specifies the interfaces that support the development of portable programs for OSF/Motif platforms.

OSF/Motif™ Programmer's Guide (069-100324)

A guide to programming using the various components of the OSF/Motif environment: the toolkit, window manager, and user interface language.

OSF/Motif™ Style Guide (069-100323)

Provides a framework for behavior specifications to guide application developers, widget developers, and window manager developers in the design of new products consistent with Presentation Manager and the OSF/Motif user interface.

Porting Applications to the DG/UX™ System (069-701059)

Describes how to port UNIX application programs to the DG/UX system.

POSIX.1 Conformance Document (069-701078)

Gives definitions and general requirements for conforming to the IEEE POSIX.1 standard.

Programmer's Reference for the DG/UX™ System (093-701055 and 093-701056)

Alphabetical listing of manual pages for programming commands on the DG/UX system. This two-volume set includes information on system calls, file formats, subroutines, and libraries.

Programmer's Reference for the X.25 Provider Interface on the DG/UX™ System (093-701082)

Describes how to use the data structures and messages of the X.25 Provider Interface in application programs.

Programming in the DG/UX™ System Application Environment (093-701076)

Discusses libraries, interprocess communications, programming interface, common object file format, and other programming-related topics.

Programming with TCP/IP on the DG/UX™ System (093-701024)

Describes how to program with the TCP and IP protocols and UDP interfaces.

Setting Up and Managing PAD on the DG/UX™ System (093-701073)

Tells you how to set up and manage the Packet Assembler/Disassembler (PAD) for AViiON Systems package. Also contains manual pages for the PAD package.

Setting Up and Managing TCP/IP on the DG/UX™ System (093-701051)

Explains how to prepare for the installation of Data General's TCP/IP (DG/UX) package on AViiON computer systems. Contains information on tailoring the software for your site, managing the system, and troubleshooting system problems.

Setting Up and Managing X.25 on the DG/UX™ System (093-701071)

This manual is for X.25 wide area network system administrators. It describes how to set up and manage the X.25 for AViiON Systems software. It also contains manual pages for the X.25 package.

STREAMS Primer for the DG/UX™ System (069-701033)

Defines STREAMS, a set of tools for developing DG/UX system communications services; explains how to build a stream; and discusses user-level and kernel-level functions.

STREAMS Programmer's Guide for the DG/UX™ System (069-701034)

Describes the development methods and design philosophy of STREAMS.

System Manager's Reference for the DG/UX™ System (093-701050)

Contains an alphabetical listing of manual pages for commands relating to system administration or operation.

User's Reference for the DG/UX™ System (093-701054)

Contains an alphabetical listing of manual pages for commands relating to general system operation.

Using API LU0,1,2,3 for AViiON™ Systems (093-000679)

Explains how to use the application program interface (API) for Logical Unit (LU) types 0, 1, 2, and 3 of IBM's System Network Architecture (SNA).

Using API LU6.2 for AViiON™ Systems (093-000680)

Explains how to use the application program interface (API) for Logical Unit (LU) type 6.2 of IBM's System Network Architecture (SNA).

Using PAD on the DG/UX™ System (069-701079)

Describes the user interface to the X.25 Packet Assembler/Disassembler (PAD) for AViiON Systems package.

Using SNA 3270 for AViiON™ Systems (093-000677)

Explains how to use the 3278 display and 3287 printer emulation capabilities within a multi-user environment.

Using SNA for AViiON™ Systems (093-000676)

Explains how to activate the SNA link to the host, establish node processes, and create configurations.

Using SNA/RJE for AViiON™ Systems (093-000678)

Explains how to use the 3776 emulation capabilities to submit jobs to and receive output from the host.

Using TCP/IP on the DG/UX™ System (093-701023)

Introduces Data General's implementation of the TCP/IP family of protocols and describes how to use the package.

Using the DG/UX™ Editors (069-701036)

Describes the text editors **vi** and **ed**, the batch editor **sed**, and the command line editor **editread**.

Using the DG/UX™ Kernel Debugger (093-701075)

Explains how to use the DG/UX kernel debugger to analyze the state of the kernel's internal data structures and the state of the underlying hardware's registers and memory.

Using the DG/UX™ Software Development Tools (093-701078)

Discusses programming support tools (**awk**, **nawk**, **lex**, **yacc**, **ld**, **lint**, and **as**), archiving, the C language, and SCCS.

Using the DG/UX™ System (069-701035)

Describes the DG/UX system and its major features, including **mailx**, the C shell, the Bourne shell, and the filing system.

Using the Documenter's Tool Kit on the DG/UX™ System (069-701039)

Provides a series of tutorials about the tools included in the Documenter's Tool Kit package. Describes the **mm** and **mv** macroinstruction packages; the **tbl**, **eqn**, **pic**, and **grap** preprocessors; the tools **checkmm**, **diffmk**, **hyphen**, **ndx**, and **subj**, and the nroff/troff formatter.

Writing a Device Driver for the DG/UX™ System (093-701053)

Describes how to write a device driver for a DG/UX system running on an AViiON computer. Describes the drivers written to address specific devices or adapters that manage secondary bus access to specific devices.

xlib Programming Manual (069-100227)

Explains programming concepts and techniques for the X library, which is the lowest level programming interface to the X Window System.

xlib Reference Manual (069-100228)

Provides a programmer's reference to the X library, including information about functions, event types, macroinstructions, and structures.

X Window System User's Guide (069-100229)

Explains the X Window System and common client applications, and describes how to customize the X environment.

Other Organizations' Documents

To obtain any of the following documents, contact the indicated organization directly.

AIC-6250 High-Performance Protocol Chip data sheet (Adaptec)

Brooktree® Product Databook (Brooktree Corporation)

Local Area Controller Am7990 (LANCE) Technical Manual (Advance Micro Devices)

Memory Products Databook (SGS-Thompson Microelectronics)

Microprocessor Data Manual (Signetics)

The VMEbus Specification (Motorola)

uPD72120 Advanced Graphics Display Controller User's Manual (NEC, Inc.)

Z8536 Z-CIP/Z8536 CIO Counter/Timer and Parallel I/O Unit (Zilog, Inc.)

TIPS ORDERING PROCEDURES

TO ORDER

1. An order can be placed with the TIPS group in two ways:
   a) MAIL ORDER – Use the order form on the opposite page and fill in all requested information. Be sure to include shipping charges and local sales tax. If applicable, write in your tax exempt number in the space provided on the order form.

      Send your order form with payment to: Data General Corporation
      ATTN: Educational Services/TIPS G155
      4400 Computer Drive
      Westboro, MA 01581–9973

   b) TELEPHONE – Call TIPS at (508) 870–1600 for all orders that will be charged by credit card or paid for by purchase orders over $50.00. Operators are available from 8:30 AM to 5:00 PM EST.

METHOD OF PAYMENT

2. As a customer, you have several payment options:
   a) Purchase Order – Minimum of $50. If ordering by mail, a hard copy of the purchase order must accompany order.
   b) Check or Money Order – Make payable to Data General Corporation.
   c) Credit Card – A minimum order of $20 is required for Mastercard or Visa orders.

SHIPPING

3. To determine the charge for UPS shipping and handling, check the total quantity of units in your order and refer to the following chart:

Total Quantity	Shipping & Handling Charge
1–4 Units	$5.00
5–10 Units	$8.00
11–40 Units	$10.00
41–200 Units	$30.00
Over 200 Units	$100.00

If overnight or second day shipment is desired, this information should be indicated on the order form. A separate charge will be determined at time of shipment and added to your bill.

VOLUME DISCOUNTS

4. The TIPS discount schedule is based upon the total value of the order.

Order Amount	Discount
$1–$149.99	0%
$150–$499.99	10%
Over $500	20%

TERMS AND CONDITIONS

5. Read the TIPS terms and conditions on the reverse side of the order form carefully. These must be adhered to at all times.

DELIVERY

6. Allow at least two weeks for delivery.

RETURNS

7. Items ordered through the TIPS catalog may not be returned for credit.
8. Order discrepancies must be reported within 15 days of shipment date. Contact your TIPS Administrator at (508) 870–1600 to notify the TIPS department of any problems.

INTERNATIONAL ORDERS

9. Customers outside of the United States must obtain documentation from their local Data General Subsidiary or Representative. Any TIPS orders received by Data General U.S. Headquarters will be forwarded to the appropriate DG Subsidiary or Representative for processing.

TIPS ORDER FORM

Mail To: Data General Corporation
Attn: Educational Services/TIPS G155
4400 Computer Drive
Westboro, MA 01581 - 9973

BILL TO:	SHIP TO: (No P.O. Boxes - Complete Only If Different Address)
COMPANY NAME	COMPANY NAME
ATTN:	ATTN:
ADDRESS	ADDRESS (NO PO BOXES)
CITY	CITY
STATE ZIP	STATE ZIP

Priority Code ____________ (See label on back of catalog)

Authorized Signature of Buyer (Agrees to terms & conditions on reverse side) | Title | Date | Phone (Area Code) | Ext.

ORDER #	QTY	DESCRIPTION	UNIT PRICE	TOTAL PRICE

A SHIPPING & HANDLING

☐ UPS	ADD
1-4 Items	$ 5.00
5-10 Items	$ 8.00
11-40 Items	$ 10.00
41-200 Items	$ 30.00
200+ Items	$100.00

Check for faster delivery

Additional charge to be determined at time of shipment and added to your bill.

☐ UPS Blue Label (2 day shipping)
☐ Red Label (overnight shipping)

B VOLUME DISCOUNTS

Order Amount	Save
$0 - $149.99	0%
$150 - $499.99	10%
Over $500.00	20%

Tax Exempt # or Sales Tax (if applicable) ____________

ORDER TOTAL	
Less Discount See B	-
SUB TOTAL	
Your local* sales tax	+
Shipping and handling - See A	+
TOTAL - See C	

C PAYMENT METHOD

☐ Purchase Order Attached ($50 minimum)
P.O. number is________. (Include hardcopy P.O.)
☐ Check or Money Order Enclosed
☐ Visa ☐ MasterCard ($20 minimum on credit cards)

Account Number ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐ Expiration Date ☐☐☐☐

Authorized Signature
(Credit card orders without signature and expiration date cannot be processed.)

THANK YOU FOR YOUR ORDER

PRICES SUBJECT TO CHANGE WITHOUT PRIOR NOTICE.
PLEASE ALLOW 2 WEEKS FOR DELIVERY.
NO REFUNDS NO RETURNS.

* Data General is required by law to collect applicable sales or use tax on all purchases shipped to states where DG maintains a place of business, which covers all 50 states. Please include your local taxes when determining the total value of your order. If you are uncertain about the correct tax amount, please call 508-870-1600.

Form 702
Rev. 8/87

DATA GENERAL CORPORATION TECHNICAL INFORMATION AND PUBLICATIONS SERVICE TERMS AND CONDITIONS

Data General Corporation ("DGC") provides its Technical Information and Publications Service (TIPS) solely in accordance with the following terms and conditions and more specifically to the Customer signing the Educational Services TIPS Order Form. These terms and conditions apply to all orders, telephone, telex, or mail. By accepting these products the Customer accepts and agrees to be bound by these terms and conditions.

1. CUSTOMER CERTIFICATION

Customer hereby certifies that it is the owner or lessee of the DGC equipment and/or licensee/sub-licensee of the software which is the subject matter of the publication(s) ordered hereunder.

2. TAXES

Customer shall be responsible for all taxes, including taxes paid or payable by DGC for products or services supplied under this Agreement, exclusive of taxes based on DGC's net income, unless Customer provides written proof of exemption.

3. DATA AND PROPRIETARY RIGHTS

Portions of the publications and materials supplied under this Agreement are proprietary and will be so marked. Customer shall abide by such markings. DGC retains for itself exclusively all proprietary rights (including manufacturing rights) in and to all designs, engineering details and other data pertaining to the products described in such publication. Licensed software materials are provided pursuant to the terms and conditions of the Program License Agreement (PLA) between the Customer and DGC and such PLA is made a part of and incorporated into this Agreement by reference. A copyright notice on any data by itself does not constitute or evidence a publication or public disclosure.

4. LIMITED MEDIA WARRANTY

DGC warrants the CLI Macros media, provided by DGC to the Customer under this Agreement, against physical defects for a period of ninety (90) days from the date of shipment by DGC. DGC will replace defective media at no charge to you, provided it is returned postage prepaid to DGC within the ninety (90) day warranty period. This shall be your exclusive remedy and DGC's sole obligation and liability for defective media. This limited media warranty does not apply if the media has been damaged by accident, abuse or misuse.

5. DISCLAIMER OF WARRANTY

EXCEPT FOR THE LIMITED MEDIA WARRANTY NOTED ABOVE, DGC MAKES NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY AND FITNESS FOR PARTICULAR PURPOSE ON ANY OF THE PUBLICATIONS, CLI MACROS OR MATERIALS SUPPLIED HEREUNDER.

6. LIMITATION OF LIABILITY

A. CUSTOMER AGREES THAT DGC'S LIABILITY, IF ANY, FOR DAMAGES, INCLUDING BUT NOT LIMITED TO LIABILITY ARISING OUT OF CONTRACT, NEGLIGENCE, STRICT LIABILITY IN TORT OR WARRANTY SHALL NOT EXCEED THE CHARGES PAID BY CUSTOMER FOR THE PARTICULAR PUBLICATION OR CLI MACRO INVOLVED. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO CLAIMS FOR PERSONAL INJURY CAUSED SOLELY BY DGC'S NEGLIGENCE. OTHER THAN THE CHARGES REFERENCED HEREIN, IN NO EVENT SHALL DGC BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES WHATSOEVER, INCLUDING BUT NOT LIMITED TO LOST PROFITS AND DAMAGES RESULTING FROM LOSS OF USE, OR LOST DATA, OR DELIVERY DELAYS, EVEN IF DGC HAS BEEN ADVISED, KNEW OR SHOULD HAVE KNOWN OF THE POSSIBILITY THEREOF; OR FOR ANY CLAIM BY ANY THIRD PARTY.

B. ANY ACTION AGAINST DGC MUST BE COMMENCED WITHIN ONE (1) YEAR AFTER THE CAUSE OF ACTION ACCRUES.

7. GENERAL

A valid contract binding upon DGC will come into being only at the time of DGC's acceptance of the referenced Educational Services Order Form. Such contract is governed by the laws of the Commonwealth of Massachusetts, excluding its conflict of law rules. Such contract is not assignable. These terms and conditions constitute the entire agreement between the parties with respect to the subject matter hereof and supersedes all prior oral or written communications, agreements and understandings. These terms and conditions shall prevail notwithstanding any different, conflicting or additional terms and conditions which may appear on any order submitted by Customer. DGC hereby rejects all such different, conflicting, or additional terms.

8. IMPORTANT NOTICE REGARDING AOS/VS INTERNALS SERIES (ORDER #1865 & #1875)

Customer understands that information and material presented in the AOS/VS Internals Series documents may be specific to a particular revision of the product. Consequently user programs or systems based on this information and material may be revision-locked and may not function properly with prior or future revisions of the product. Therefore, Data General makes no representations as to the utility of this information and material beyond the current revision level which is the subject of the manual. Any use thereof by you or your company is at your own risk. Data General disclaims any liability arising from any such use and I and my company (Customer) hold Data General completely harmless therefrom.

134-755-02

moisten & seal

CUSTOMER DOCUMENTATION COMMENT FORM

Your Name ______________________ Your Title ______________

Company ______________________ Phone ______________

Street ______________________________________

City ______________________ State ________ Zip ______

We wrote this book for you, and we made certain assumptions about who you are and how you would use it. Your comments will help us correct our assumptions and improve the manual. Please take a few minutes to respond. Thank you.

Manual Title ______________________ Manual No. ______________

Who are you?	☐ EDP/MIS Manager	☐ Analyst/Programmer	☐ Other ______
	☐ Senior Systems Analyst	☐ Operator	______
	☐ Engineer	☐ End User	

How do you use this manual? *(List in order: 1 = Primary Use)*

___ Introduction to the product	___ Tutorial Text	___ Other
___ Reference	___ Operating Guide	______

		Yes	No
About the manual:	Is it easy to read?	☐	☐
	Is it easy to understand?	☐	☐
	Are the topics logically organized?	☐	☐
	Is the technical information accurate?	☐	☐
	Can you easily find what you want?	☐	☐
	Does it tell you everything you need to know?	☐	☐
	Do the illustrations help you?	☐	☐

If you wish to order manuals, use the enclosed TIPS Order Form (USA only) or contact your sales representative or dealer.

Comments:

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 26 WESTBORO, MA 01581

POSTAGE WILL BE PAID BY ADDRESSEE

DataGeneral

Customer Documentation
MS E-111
4400 Computer Drive
P.O. Box 4400
Westboro, MA 01581-9890

134-755-02

Using
TCP/IP on
the DG/UX™
System

093-701023-02

Cut here and insert in binder spine pocket

Data General

Data General Corporation, Westboro, Massachusetts 01580

093-701023-02