

OpenVOS STREAMS TCP/IP User's Guide

Stratus Technologies
R421-04

Notice

The information contained in this document is subject to change without notice.

UNLESS EXPRESSLY SET FORTH IN A WRITTEN AGREEMENT SIGNED BY AN AUTHORIZED REPRESENTATIVE OF STRATUS TECHNOLOGIES, STRATUS MAKES NO WARRANTY OR REPRESENTATION OF ANY KIND WITH RESPECT TO THE INFORMATION CONTAINED HEREIN, INCLUDING WARRANTY OF MERCHANTABILITY AND FITNESS FOR A PURPOSE. Stratus Technologies assumes no responsibility or obligation of any kind for any errors contained herein or in connection with the furnishing, performance, or use of this document.

Software described in Stratus documents (a) is the property of Stratus Technologies Bermuda, Ltd. or the third party, (b) is furnished only under license, and (c) may be copied or used only as expressly permitted under the terms of the license.

Stratus documentation describes all supported features of the user interfaces and the application programming interfaces (API) developed by Stratus. Any undocumented features of these interfaces are intended solely for use by Stratus personnel and are subject to change without warning.

This document is protected by copyright. All rights are reserved. No part of this document may be copied, reproduced, or translated, either mechanically or electronically, without the prior written consent of Stratus Technologies.

Stratus, the Stratus logo, ftServer, the ftServer logo, Continuum, StrataLINK, and StrataNET are registered trademarks of Stratus Technologies Bermuda, Ltd.

The Stratus Technologies logo, the Continuum logo, the Stratus 24 x 7 logo, ActiveService, ftScalable, and ftMessaging are trademarks of Stratus Technologies Bermuda, Ltd.

RSN is a trademark of Lucent Technologies, Inc.

All other trademarks are the property of their respective owners.

TCP Wrappers copyright information:

Copyright (c) 1987 Regents of the University of California.
All rights reserved.

Redistribution and use in source and binary forms are permitted provided that the above copyright notice and this paragraph are duplicated in all such forms and that any documentation, advertising materials, and other materials related to such distribution and use acknowledge that the software was developed by the University of California, Berkeley. The name of the University may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Copyright 1995 by Wietse Venema. All rights reserved. Some individual files may be covered by other copyrights.

This material was originally written and compiled by Wietse Venema at Eindhoven University of Technology, The Netherlands, in 1990, 1991, 1992, 1993, 1994 and 1995.

Redistribution and use in source and binary forms are permitted provided that this entire copyright notice is duplicated in all such copies.

This software is provided "as is" and without any expressed or implied warranties, including, without limitation, the implied warranties of merchantability and fitness for any particular purpose.

Manual Name: *OpenVOS STREAMS TCP/IP User's Guide*

Part Number: R421

Revision Number: 04

OpenVOS Release Number: 19.1.0

Publication Date: November 2017

Stratus Technologies, Inc.
5 Mill and Main Place, Suite 500
Maynard, Massachusetts 01754-2660

© 2017 Stratus Technologies Bermuda, Ltd. All rights reserved.

Contents

Preface	ix
----------------	----

1. Overview of STREAMS TCP/IP (STCP)	1-1
FTP, TELNET, and TFTP	1-1
Guidelines for Using This Manual	1-3

2. The FTP Utility	2-1
Using FTP	2-1
Beginning and Ending an FTP Session	2-3
Establishing Communications	2-4
Opening a Connection	2-5
Logging In to the Remote Host	2-7
Creating an Automatic Login File	2-9
Opening a Proxy Connection	2-10
Closing a Connection	2-12
Displaying Status Information	2-13
Displaying the Status of the FTP Session	2-13
Displaying the Status of the Current Connection	2-14
Displaying the System Type of the Remote Host	2-14
Setting Modes	2-14
File-Transfer Modes	2-17
Verbose Mode	2-21
Bell Mode	2-21
Interactive-Prompting Mode	2-21
Global File-Name-Expansion Mode	2-22
Store-Unique and Receive-Unique Modes	2-24
Case-Mapping Mode	2-25
CR-Stripping Mode	2-25
Character-Translation Mode	2-25
File-Name-Mapping Mode	2-27
Debugging Mode	2-29
Hash-Mark Mode	2-30
Sendport Mode	2-30

Passive Mode	2-30
Performing File Operations	2-31
Sending Files to the Remote Host	2-32
Appending a Local File	2-33
Appending a Remote File	2-35
Receiving Files from the Remote Host	2-36
Renaming Files on the Remote Host	2-38
Deleting Files on the Remote Host	2-39
Displaying the Last Modification Time	2-40
Displaying File Size	2-40
Performing Directory Operations	2-40
Displaying the Current Directory	2-41
Changing the Current Directory	2-42
Listing the Contents of a Directory	2-43
Creating and Deleting Directories	2-44
Defining and Using Macros	2-45
Defining and Executing a Macro	2-45
Defining and Executing a Macro That Accepts Arguments	2-48
Verifying Subcommand Status	2-50
Getting Help	2-50
Getting Information about FTP Subcommands	2-51
Getting Information about FTP Protocol Commands	2-52
Issuing FTP Protocol Commands	2-52
Issuing Operating-System Commands	2-53
Issuing Operating-System Commands from FTP	2-54
Issuing Operating-System Commands from a Subprocess	2-54
Connecting to a Stratus Remote Host	2-55
Logging In	2-55
Determining the Initial Current Directory	2-56
Appending a File	2-56
Sample FTP Sessions	2-57
FTP Session with a Stratus Remote Host	2-57
FTP Session with a Non-Stratus Remote Host	2-60

3. The TELNET Utility	3-1
Using TELNET	3-2
Beginning and Ending a TELNET Session	3-4
Establishing Communications	3-5
Opening a Connection	3-5
Closing a Connection	3-7
Working in Input Mode	3-8
Logging In to the Remote Host	3-8
Setting the Terminal Type	3-8
Issuing Subcommands	3-10

Issuing Input Requests	3-10
Defining Key Sequences for Input Requests	3-12
Setting Operating Modes	3-13
Displaying Information	3-15
Displaying Online Help for Subcommands	3-15
Displaying the Current Settings	3-16
Displaying the Status of the TELNET Session	3-16
Connecting to a Stratus Remote Host	3-17
Logging In to the VOS Operating System	3-17
Setting the Terminal Type	3-18
Determining the Initial Current Directory	3-18
The OpenVOS Command Interface	3-18
The STCP TELNET Server	3-19
Providing Connections for Applications	3-19
Issuing TELNET Input Requests	3-19
Sample TELNET Session	3-20

4. The TFTP Utility	4-1
Using TFTP	4-1
TFTP Example	4-2

Appendix A. The <code>ftp</code> Command, Subcommands, and Server Messages	A-1
<code>ftp</code>	A-2
The FTP Subcommands	A-4
The FTP Server Messages	A-20

Appendix B. The <code>telnet</code> Command and Subcommands	B-1
<code>telnet</code>	B-2
The TELNET Subcommands	B-4

Appendix C. The <code>tftp</code> Command	C-1
<code>tftp</code>	C-2

Index	Index-1
--------------	---------

Figures

Figure 1-1. Protocols, STCP, and the Network Interface	1-2
Figure 2-1. FTP Client/Server Connection	2-2
Figure 3-1. TELNET Client/Server Connection	3-2

Tables

Table 2-1.	FTP Subcommands for Establishing Communications	2-4
Table 2-2.	Keywords for the <code>.netrc</code> File	2-9
Table 2-3.	FTP Subcommands in Proxy Connections	2-12
Table 2-4.	FTP Subcommands for Displaying Status Information	2-13
Table 2-5.	FTP Subcommands for Setting Modes	2-15
Table 2-6.	FTP Subcommands for Setting the File-Transfer Type	2-17
Table 2-7.	File-Transfer Types for Transfers between Stratus Hosts	2-20
Table 2-8.	FTP Subcommands for Displaying Fixed-Setting File-Transfer Modes	2-21
Table 2-9.	Sample Output from File-Name-Mapping Mode	2-28
Table 2-10.	FTP Subcommands for File Operations	2-31
Table 2-11.	FTP Subcommands for Directory Operations	2-41
Table 2-12.	FTP Subcommands for Macros	2-45
Table 2-13.	FTP Subcommands for Getting Help	2-50
Table 2-14.	FTP Subcommands for Issuing Operating-System Commands	2-53
Table 3-1.	TELNET Subcommands	3-3
Table 3-2.	TELNET Input Requests	3-11
Table 3-3.	Input Requests and Corresponding <code>set</code> Arguments	3-12
Table 3-4.	TELNET Operating Modes	3-14
Table 3-5.	Responses to Client Requests Supported by STCP TELNET Server	3-20

Preface

The *OpenVOS STREAMS TCP/IP User's Guide* (R421) documents how to use the STREAMS TCP/IP (STCP) version of FTP, TELNET, and TFTP on a Stratus module running OpenVOS Release 19.1.0.

This manual is intended for users who want to use the STCP version of FTP, TELNET, and TFTP for Internet network communications.

If you were a user of OS TCP/IP, you may want to read the *OpenVOS STREAMS TCP/IP Migration Guide* (R418) for differences between STCP and OS TCP/IP.

If you have need for related STCP administrative or programming information, you may want to become familiar with the following manuals.

- *OpenVOS STREAMS TCP/IP Administrator's Guide* (R419)
- *OpenVOS STREAMS TCP/IP Programmer's Guide* (R420)

Manual Version

This manual is a revision. A new argument was added to the `ftp` command.

Change bars, which appear in the margin, note the specific changes to text since the previous publication of this manual. Note, however, that change bars are not used in new chapters or appendixes.

Manual Organization

This manual contains four chapters and three appendixes.

[Chapter 1](#) contains an overview of STCP.

[Chapter 2](#) describes how to use FTP.

[Chapter 3](#) describes how to use TELNET.

[Chapter 4](#) describes how to use TFTP.

[Appendix A](#) provides reference documentation for the FTP command and its arguments.

[Appendix B](#) provides reference documentation for the TELNET command and its arguments.

[Appendix C](#) provides reference documentation for the TFTP command and its arguments.

Related Manuals

Refer to the following Stratus manuals for additional information about STCP.

- *OpenVOS STREAMS TCP/IP Migration Guide* (R418)
- *OpenVOS STREAMS TCP/IP Administrator's Guide* (R419)
- *OpenVOS STREAMS TCP/IP Programmer's Guide* (R420)

Refer to the following Stratus manuals for information about OS TCP/IP.

- *VOS Communications Software: OS TCP/IP User's Guide* (R222)
- *VOS OS TCP/IP Administrator's Guide* (R223)
- *VOS Communications Software: OS TCP/IP Programmer's Manual* (R224)

Refer to the following Stratus manuals for additional VOS information.

- *OpenVOS Commands Reference Manual* (R098)
- *OpenVOS Installation Guide* (R386)

Notation Conventions

This manual uses the following notation conventions.

Warnings, Cautions, Notices, and Notes

Warnings, cautions, notices, and notes provide special information and have the following meanings:



WARNING _____

A warning indicates a hazardous situation that, if not avoided, could result in death or serious injury.



AVERTISSEMENT _____

Un avertissement indique une situation dangereuse qui, si pas évitée, pourrait entraîner la mort ou des blessures graves.



CAUTION

A caution indicates a hazardous situation that, if not avoided, could result in minor or moderate injury.



MISE EN GARDE

Une mise en garde indique une situation dangereuse qui, si pas évitée, pourrait entraîner des blessures mineures ou modérées.

NOTICE

A notice indicates information that, if not acted on, could result in damage to a system, hardware device, program, or data, but does not present a health or safety hazard.

NOTE

A note provides important information about the operation of an ftServer system or related equipment or software.

Typographical Conventions

The following typographical conventions are used in this manual:

- Italics introduces or defines new terms. For example:

The *master disk* is the name of the member disk from which the module was booted.

- Boldface emphasizes words in text. For example:

Every module **must** have a copy of the `module_start_up.cm` file.

- Monospace represents text that would appear on your terminal's screen (such as commands, subroutines, code fragments, and names of files and directories). For example:

```
change_current_dir (master_disk)>system>doc
```

- Monospace italic represents terms that are to be replaced by literal values. In the following example, the user must replace the monospace-italic term with a literal value.

```
list_users -module module_name
```

-
- Monospace bold represents user input in examples and figures that contain both user input and system output (which appears in monospace). For example:

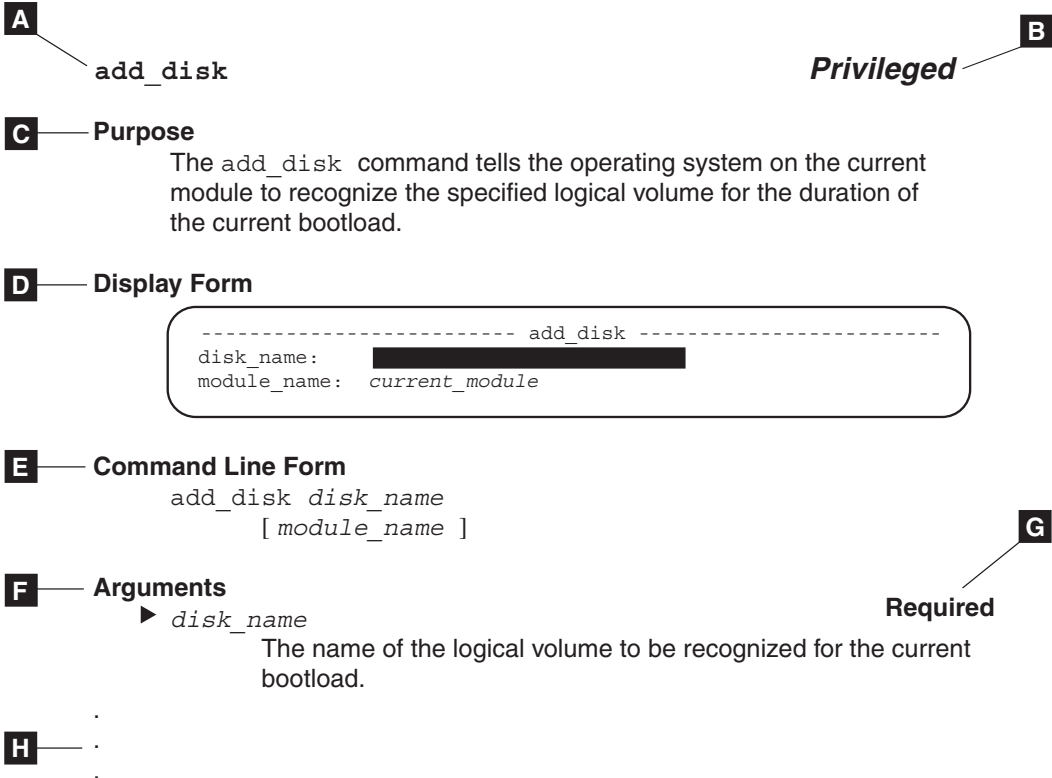
```
display_access_list system_default
```

```
%dev#m1>system>acl>system_default
```

```
w  *.*
```

Format for Commands and Requests

Stratus manuals use the following format conventions for documenting commands and requests. (A *request* is typically a command used within a subsystem, such as `analyze_system`.) Note that the command and request descriptions do not necessarily include each of the following sections.





- A** **name**
The name of the command or request is at the top of the first page of the description.
- B** **Privileged**
This notation appears after the name of a command or request that can be issued only from a privileged process.
- C** **Purpose**
Explains briefly what the command or request does.

D Display Form

Shows the form that is displayed when you type the command or request name followed by `-form` or when you press the key that performs the `DISPLAY FORM` function. Each field in the form represents a command or request argument. If an argument has a default value, that value is displayed in the form.

The following table explains the notation used in display forms.

The Notation Used in Display Forms

Notation	Meaning
	Required field with no default value.
	The cursor, which indicates the current position on the screen. For example, the cursor may be positioned on the first character of a value, as in <code>a11</code> .
<code>current_user</code> <code>current_module</code> <code>current_system</code> <code>current_disk</code>	The default value is the current user, module, system, or disk. The actual name is displayed in the display form of the command or request.

E Command-Line Form

Shows the syntax of the command or request with its arguments. You can display an online version of the command-line form of a command or request by typing the command or request name followed by `-usage`.

The following table explains the notation used in command-line forms. In the table, the term *multiple values* refers to explicitly stated separate values, such as two or more object names. Specifying multiple values is **not** the same as specifying a star name. When you specify multiple values, you must separate each value with a space.

The Notation Used in Command-Line Forms

Notation	Meaning
<i>argument_1</i>	Required argument.
<i>argument_1</i> ...	Required argument for which you can specify multiple values.
$\left\{ \begin{array}{l} \textit{argument_1} \\ \textit{argument_2} \end{array} \right\}$	Set of arguments that are mutually exclusive; you must specify one of these arguments.
$\left[\textit{argument_1} \right]$	Optional argument.
$\left[\textit{argument_1} \right]$...	Optional argument for which you can specify multiple values.
$\left[\begin{array}{l} \textit{argument_1} \\ \textit{argument_2} \end{array} \right]$	Set of optional arguments that are mutually exclusive; you can specify only one of these arguments.
Note: Dots, brackets, and braces are not literal characters; you should not type them. Any list or set of arguments can contain more than two elements. Brackets and braces are sometimes nested.	

F Arguments

Describes the command or request arguments. The following table explains the notation used in argument descriptions.

G The Notation Used in Argument Descriptions

Notation	Meaning
$\boxed{\text{CYCLE}}$	There are predefined values for this argument. In the display form, you display these values in sequence by pressing the key that performs the <code>CYCLE</code> function.
Required	<p>You cannot issue the command or request without specifying a value for this argument.</p> <p>If an argument is required but has a default value, it is not labeled Required since you do not need to specify it in the command-line form. However, in the display form, a required field must have a value—either the displayed default value or a value that you specify.</p>
(Privileged)	Only a privileged process can specify a value for this argument.

-
- H** The following additional headings may appear in the command or request description: Explanation, Error Messages, Examples, and Related Information.

Explanation

Explains how to use the command or request and provides supplementary information.

Error Messages

Lists common error messages with a short explanation.

Examples

Illustrates uses of the command or request.

Related Information

Refers you to related information (in this manual or other manuals), including descriptions of commands, subroutines, and requests that you can use with or in place of this command or request.

Getting Help

If you have a technical question about ftServer system hardware or software, try these online resources first:

- **Online documentation at the StrataDOC Web site.** Stratus provides complimentary access to StrataDOC, an online-documentation service that enables you to view, search, download, and print customer documentation. You can access StrataDOC at the following Web site:
<http://stratadoc.stratus.com>
- **Online support from Stratus Customer Service.** You can find the latest technical information about an ftServer system in the Stratus Customer Service Portal at the following Web site:

<http://www.stratus.com/go/support>

The Service Portal provides access to Knowledge Base articles for all Stratus product lines. You can locate articles by performing a simple or advanced keyword search, viewing recent articles or top FAQs, or browsing a product and category.

To log in to the Service Portal, enter your employee user name and password or, if you have not been provided with a login account, click **Register Account**. When registering a new account, ensure that you specify an email address from a company that has a service agreement with Stratus.

If you cannot resolve your questions with these online self-help resources, and the ftServer system is covered by a service agreement, contact the Stratus Customer Assistance Center (CAC) or your authorized Stratus service representative. To contact the CAC, use the Service Portal to log a support request. Click **Customer Support** and

Add Issue, and then complete the **Create Issue** form. A member of our Customer Service team will be glad to assist you.

Commenting on This Manual

You can comment on this manual using one of the following methods. When you submit a comment, be sure to provide the manual's name and part number, a description of the problem, and the location in the manual where the affected text appears.

- From StrataDOC, click the **site feedback** link at the bottom of any page. In the pop-up window, answer the questions and click **Submit**.
- From any email client, send email to `comments@stratus.com`.
- From the Stratus Customer Service Portal, log on to your account and create a new issue.

Stratus welcomes any corrections and suggestions for improving this manual.

Chapter 1

Overview of STREAMS TCP/IP (STCP)

STREAMS TCP/IP (STCP) is a STREAMS-based implementation of the standard Transmission Control Protocol/Internet Protocol (TCP/IP) family of networking protocols for Stratus modules running OpenVOS. The STCP product functions within a STREAMS environment. STCP is similar in functionality to a previously released Stratus product, OS TCP/IP.

STCP supports a number of standard TCP/IP protocols including three user-level protocols: File Transfer Protocol (FTP), TELNET, and Trivial File Transfer Protocol (TFTP). This manual provides the information you need to use FTP, TELNET, and TFTP on VOS STREAMS-based I/O subsystems. This chapter contains the following sections.

- [“FTP, TELNET, and TFTP”](#)
- [“Guidelines for Using This Manual”](#)

For additional information about how to administer STCP, see the *OpenVOS STREAMS TCP/IP Administrator's Guide* (R419). For additional information about programming within the STCP environment, see the *OpenVOS STREAMS TCP/IP Programmer's Guide* (R420).

FTP, TELNET, and TFTP

FTP, TELNET, and TFTP provide communications services among computers, or *hosts*, in a network that supports the TCP/IP protocol.

NOTE

Since FTP, TELNET, and TFTP entail some security risk, you should use SSH or SFTP instead.

FTP is a file transfer protocol that allows user authentication, copying files, and transfer of data between hosts. TELNET is a protocol that provides remote login to another host so a user on one host can access the resources of the other host. TFTP is a simple protocol for transferring files from one host to another. Unlike FTP, TFTP does not require user authentication and can transfer only a limited set of files.

Figure 1-1 shows the relationship between the TCP/IP protocols, STCP, and the network interface.

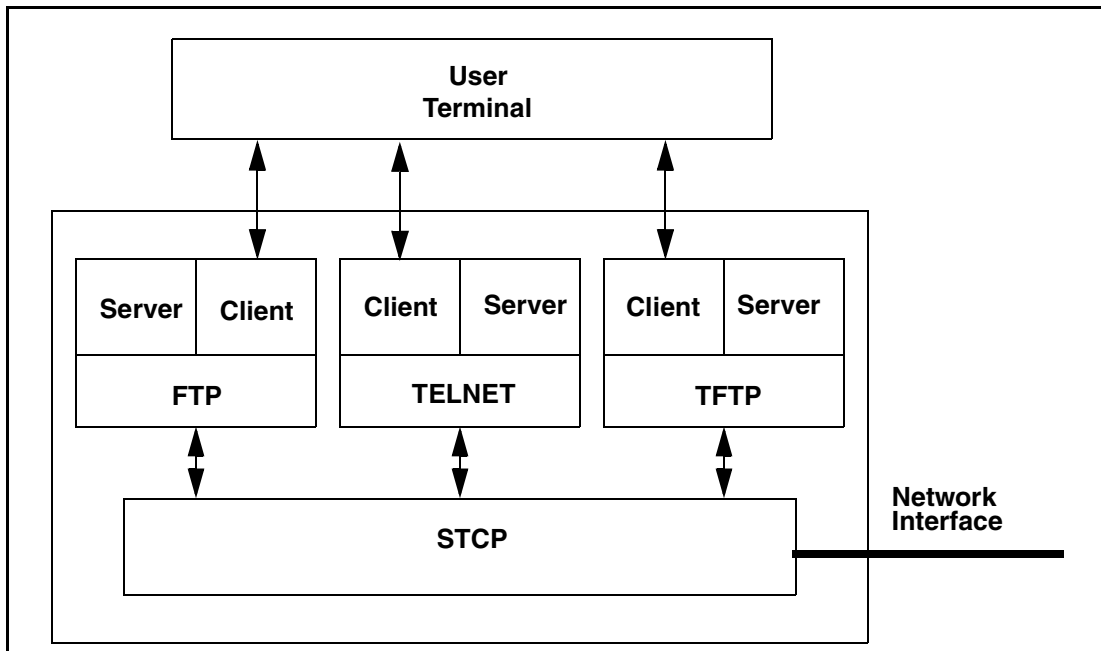


Figure 1-1. Protocols, STCP, and the Network Interface

FTP, TELNET, and TFTP consist of a client program and a server program, as illustrated in Figure 1-1. A Stratus *client program* opens a connection and communicates with the server program on another host. A Stratus *server program* receives input from and responds to the client program on another host. A *network interface* is the communications hardware that resides on the module to create the module's physical connection to the network.

In this manual, the host on which you are using the client program is called the *local host*. When you open a connection with the server program on another host, that host is called the *remote host*.

When you are using FTP, TELNET, or TFTP, you are in an FTP, TELNET, or TFTP *session*. To begin a session, you issue an OpenVOS command that starts the client program. During a session, you issue FTP, TELNET, or TFTP commands, which are called *subcommands* in this manual to distinguish them from OpenVOS commands. The actual codes that the client and server programs exchange are called *protocol commands*.

Guidelines for Using This Manual

This section describes how to use this manual.

The remainder of the manual documents FTP, TELNET, and TFTP.

- For information about using FTP, see [Chapter 2, “The FTP Utility.”](#)
- For information about using TELNET, see [Chapter 3, “The TELNET Utility.”](#)
- For information about using TFTP, see [Chapter 4, “The TFTP Utility.”](#)
- For reference information on the `ftp` command, see [Appendix A, “The `ftp` Command, Subcommands, and Server Messages.”](#)
- For reference information on the `telnet` command, see [Appendix B, “The `telnet` Command and Subcommands.”](#)
- For reference information on the `tftp` command, see [Appendix C, “The `tftp` Command.”](#)

Because STCP and OS TCP/IP are different implementations of the TCP/IP protocol, their user interfaces and functionality differ somewhat. In this manual, all references to TCP/IP, FTP, TELNET, and TFTP apply to the STCP product unless otherwise specified. For information about the compatibility of STCP with OS TCP/IP, see the *OpenVOS STREAMS TCP/IP Migration Guide* (R418). For documentation related to each of these products, see the Preface of this manual.

Many examples in this manual include one or more informational messages that appear on the screen after a subcommand is issued. In some cases, these messages are generated by the server on the remote host and therefore may be different from the message that appears on your screen.

Chapter 2

The FTP Utility

The FTP utility, referred to as FTP throughout this manual, is a user interface for the Stratus implementation of the STREAMS FTP protocol for remote file transfer. FTP allows you to communicate on a TCP/IP network with other hosts that support this protocol. FTP provides a set of subcommands for establishing communications and accessing another host's directories and files for file-transfer and related operations.

This chapter describes FTP. It contains the following sections.

- [“Using FTP”](#)
- [“Beginning and Ending an FTP Session”](#)
- [“Establishing Communications”](#)
- [“Displaying Status Information”](#)
- [“Setting Modes”](#)
- [“Performing File Operations”](#)
- [“Performing Directory Operations”](#)
- [“Defining and Using Macros”](#)
- [“Getting Help”](#)
- [“Issuing FTP Protocol Commands”](#)
- [“Issuing Operating-System Commands”](#)
- [“Connecting to a Stratus Remote Host”](#)
- [“Sample FTP Sessions”](#)

The FTP subcommands are summarized in [Appendix A](#).

Using FTP

This section provides an overview of an FTP session, and defines terms and illustrates a client/server connection. The Stratus FTP utility consists of two programs. The *FTP client* is a program that begins an FTP session and processes the FTP subcommands that you issue during the session. The *FTP server* is a program that provides access to the files and directories on a Stratus host when it is the remote host for an FTP session.

While you are using FTP, you are in an FTP session. The module on which you are using FTP is called the local host. To begin an FTP session, you log in to your Stratus module and issue the `ftp` command. This command executes the `ftp.pm` program, which is the Stratus FTP client program.

During an FTP session, you issue the FTP subcommands described in this chapter. In response to the FTP subcommands, the FTP client on the local host opens a connection with an FTP server on another host, which is the remote host. The Stratus FTP client processes subcommands that you issue and manages the exchange of input and output associated with the connection. The remote host's FTP server provides access to the files and directories on the remote host, within the access restrictions imposed by the remote host.

Figure 2-1 illustrates a connection between an FTP client running on a Stratus local host and an FTP server running on a remote host.

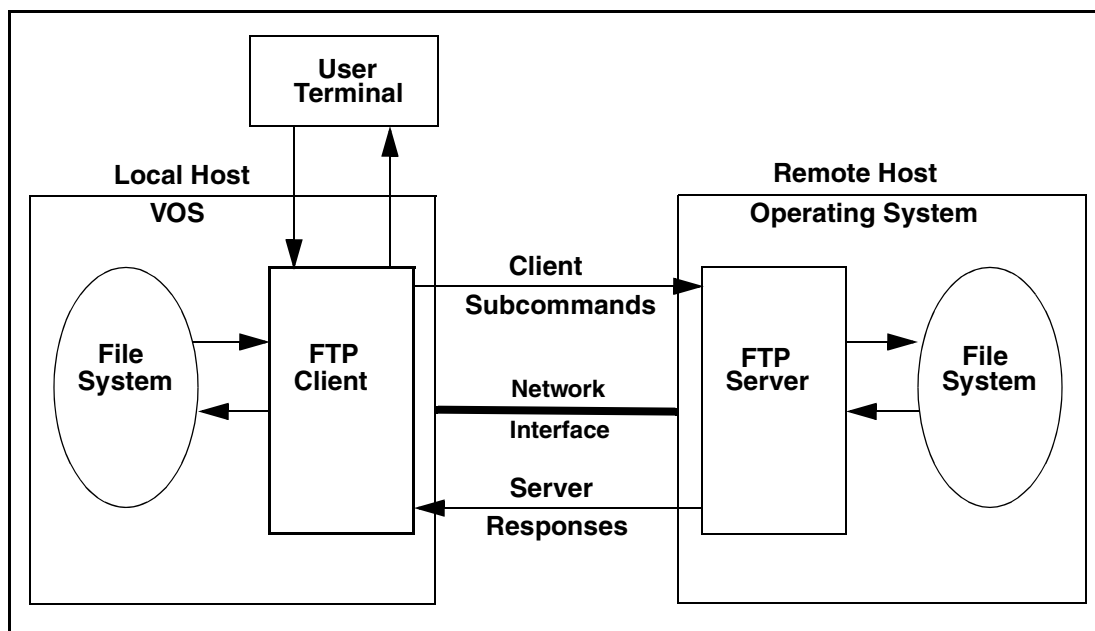


Figure 2-1. FTP Client/Server Connection

Beginning and Ending an FTP Session

This section explains how to begin and end an FTP session. To begin an FTP session, you issue the `ftp` command. This command has the following format.

```
ftp [host] [-debug] [-no_verbos] [-no_globbing]
    [-no_interactive] [-no_autologin] [-os_break]
```

If you issue the `ftp` command with no arguments, the FTP session has the following characteristics.

- You are not connected to a remote host.
- Debugging mode is off. When this mode is on, each time you issue a subcommand, FTP displays the actual FTP protocol commands that it sends; when this mode is off, these commands are not displayed.
- Verbose mode is on. When this mode is on, FTP displays the messages that the FTP server returns; when this mode is off, the messages are not displayed.
- Global file-name-expansion mode is on. When this mode is on, you can use star names and abbreviations when specifying files.
- Interactive-prompting mode is on. When this mode is on, FTP asks you for verification of each file if a subcommand will affect multiple files or if a file will overwrite an existing file on the local host.
- Default break mode is off. When this mode is on, FTP always uses the default break handler, which is the operating-system break handler; when this mode is off, FTP uses its own break handlers during data transfers. The FTP break handlers allow you to abort the current data transfer without terminating the `ftp` process. For example, when you issue your terminal's break command, such as `Ctrl Break`, during a file transfer, the FTP break handlers abort the file transfer and return you to the FTP prompt, `ftp>`.

Most of the `ftp` command arguments have corresponding FTP subcommands. For additional information about the `host` argument and the corresponding subcommand, see the next section, “[Establishing Communications](#).” For additional information about the mode arguments and the corresponding subcommands, see the section “[Setting Modes](#)” later in this chapter.

If you issue the `ftp` command without specifying a remote host, FTP displays the `ftp>` prompt. If you specify a remote host, a sequence of login prompts will appear before the `ftp>` prompt appears on the screen. The login prompts are described in the next section.

The following example illustrates the `ftp` command followed by the `ftp>` prompt.

```
ftp
ftp>
```

At the `ftp>` prompt, you can issue an FTP subcommand. In this chapter, most of the examples that illustrate subcommands include the `ftp>` prompt and the messages that the FTP server returns.

To end an FTP session, you issue the `bye` or `quit` subcommand. These subcommands perform the same function; you can use them interchangeably. If a connection with a remote host is open when you issue one of these subcommands, FTP closes the connection.

The following example illustrates the `bye` subcommand.

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

Establishing Communications

Before you can use FTP to access files and directories on a remote host, you must establish communications with the remote host. To establish communications, you open a connection with a remote host and log in to the remote host.

[Table 2-1](#) lists and describes the FTP subcommands for establishing communications. Some of the subcommands have a corresponding `ftp` argument for performing the same function when you issue the `ftp` command. The table lists the subcommands in the order in which they are presented in this section.

Table 2-1. FTP Subcommands for Establishing Communications

Subcommand	ftp Argument	Description
<code>open</code>	<i>host</i>	Opens a connection with a remote host
<code>user</code>	None	Sends login data to the remote host
<code>proxy</code>	None	Issues subcommands for a proxy connection
<code>close</code> <code>disconnect</code>	None	Closes a connection

This section discusses the following topics.

- [“Opening a Connection”](#)
- [“Logging In to the Remote Host”](#)
- [“Creating an Automatic Login File”](#)
- [“Opening a Proxy Connection”](#)
- [“Closing a Connection”](#)

Opening a Connection

Before you can access the files and directories on a remote host, you open an FTP connection with the remote host. When you open a connection with a remote host, the remote host creates an FTP server process to communicate with your FTP client.

There are two ways to open a connection.

- As you begin an FTP session, you can issue the `ftp` command with the *host* argument. If you do not specify a remote host, FTP begins the session without opening a connection. The `ftp` command issued with the *host* argument has the following format.

```
ftp host
```

- During an FTP session, you can open a connection by issuing the `open` subcommand. The `open` subcommand has the following format.

```
ftp> open host [port_number]
```

The *host* argument specifies the remote host. The *port_number* argument specifies the FTP command port on the remote host. (You can specify the command port only when you open a connection with the `open` subcommand; the `ftp` command does not accept the *port_number* argument.)

The value for the *host* argument can be either a host name or an Internet address. A *host name* is a descriptive name that uniquely identifies a host on an Internet network; an *Internet address* is a numeric identifier that uniquely identifies a remote host on an Internet network.

The following example illustrates a host name.

```
system_a
```

The following example illustrates an Internet address in the standard protocol format. You will see the address in this format in certain error and informational messages. The integer following the comma specifies a port number.

```
192.9.200.33,21
```

Host names are resolved to Internet addresses either locally through the `hosts` file (path name `>system>stcp>hosts`) or externally through the Domain Name Service. The system administrator determines whether your implementation of STCP uses the `hosts` file or the Domain Name Service to resolve host names.

If the host name for a remote host cannot be resolved to an Internet address, you cannot use the host name to open a connection with that remote host. However, you can still open a connection with the remote host by specifying its Internet address.

The following example illustrates the `ftp` command issued with a host name specified for the `host` argument. The example includes the messages that appear on the screen while the connection is being opened.

```
ftp system_a
Connected to system_a.admin.myfirm.com.
220 system_a FTP server (FTP 1.0 for Stratus STCP) ready.
(Compatible with OS TCP/IP)
```

The following example illustrates the `open` subcommand issued with an Internet address specified for the `host` argument and a port number specified for the `port_number` argument. You can specify a port number only if you specify a host.

```
ftp> open 192.9.200.1 78
```

The port number specifies the port on the remote host that receives incoming FTP protocol commands. The default FTP command port is 21, and most remote hosts use this port. To determine which port a remote host uses for the FTP command port, look in the `services` file on the remote host. On a Stratus remote host, the path name for this file is `>system>stcp>services`. (For information about ports, well-known ports, and the `services` file, see the *OpenVOS STREAMS TCP/IP Administrator's Guide* (R419).)

If the remote host does not respond to a connection request, FTP returns the following error message.

```
ftp: connect: Timeout period has expired.
```

If you have specified a valid address for the remote host and you continue to receive this error message, the remote host you are trying to reach may have been disconnected from the network.

If the FTP server is disabled on the remote host, FTP returns the following error message.

```
ftp: connect: Connection refused.
```

If you receive an error message indicating that there may be problems with the remote host's server, you can contact the system administrator of the remote host for assistance.

Logging In to the Remote Host

Before you can access files and directories on the remote host, you log in to the remote host. You must be a registered user on the remote host. When logging in, you specify your user name and, if required, your password, as registered on the remote host.

When you issue the `ftp` command or the `open` subcommand to open a connection, FTP attempts to perform the login procedure before displaying the `ftp>` prompt. There are three ways to log in to the remote host.

- If you have an automatic login file with an entry for the remote host, FTP performs an automatic login procedure before displaying the `ftp>` prompt.
- If you do not have an automatic login file with an entry for the remote host, FTP displays login prompts after you issue the `ftp` command or the `open` subcommand. If your login entries are correct, FTP logs you in before displaying the `ftp>` prompt.
- If FTP is unable to log you in when opening the connection, you can log in by issuing the `user` subcommand at the `ftp>` prompt. For example, this situation can occur if you specify your password incorrectly.

To use the automatic login procedure, you must create an automatic login file in your home directory. You use this file to store login data for the remote hosts that you access regularly. For information about the automatic login file, see the section “[Creating an Automatic Login File](#)” later in this chapter.

If FTP is unable to perform an automatic login procedure, login prompts and messages appear after you issue the `ftp` command or the `open` subcommand. When you specify your password on the `ftp>` subcommand line, FTP displays it on the screen. When you enter your password in response to a prompt, FTP does not display it on the screen.

NOTE _____

FTP does **not** encrypt passwords; it sends them to the server as ordinary data.

The following example illustrates the login prompts and messages.

```
ftp> open system_a
Connected to system_a.admin.myfirm.com.
220 system_a FTP server (FTP 1.0 for Stratus STCP) ready.
(Compatible with OS TCP/IP)
Name: Susan_Smith
331 Password required for Susan_Smith.
Password:
230 User Susan_Smith logged in.
```

If FTP is unable to log you in while opening the connection, you can log in by issuing the `user` subcommand. This subcommand has the following format.

```
ftp> user user_name [password]
```

The `user_name` argument is required and specifies your user name. The `password` argument specifies your password. If you do not specify a required password, you will be prompted to supply it.

The following example illustrates the `user` subcommand and the response from the remote host. Note that the user specifies an alias, `sue`, for her registered user name, `Susan_Smith`. You can specify an alias for your user name if the remote host recognizes aliases during login.

```
ftp> user sue
331 Password required for sue.
Password:
```

The remote host returns the following message.

```
230 User Susan_Smith.FTP_User logged in.
```

The remote host determines which directory is your initial login directory. For information about how the remote host determines this directory, see the documentation for the remote host. For a Stratus remote host, this information is presented later in this chapter in the section “[Connecting to a Stratus Remote Host.](#)”

If you attempt to log in but receive a message similar to the following message, the remote host has security restrictions implemented. In this case, see your system administrator if you require access to the remote host.

```
550 Susan_Smith.FTP_User not allowed to use FTP on this
system.
Login failed.
ftp>
```

Creating an Automatic Login File

When you open a connection with a remote host, FTP attempts to perform an automatic login procedure by looking for a file named `.netrc`. This file contains automatic login data. It must be located in your home directory on the local host. If a `.netrc` file exists and contains entries for the specified remote host, FTP performs an automatic login procedure using the login data in this file.

To set up automatic login for a remote host, you must create an entry for the remote host in a file named `.netrc` in your home directory on the local host. Each entry consists of a set of keywords and corresponding values for the remote host. [Table 2-2](#) lists the keywords and describes the keyword values.

Table 2-2. Keywords for the `.netrc` File

Keyword	Value	Description
machine	<i>host</i>	Specifies a remote host. The value that you specify for <i>host</i> in the <code>.netrc</code> file must exactly match the value that you specify for <i>host</i> when you issue the <code>ftp</code> command or the <code>open</code> subcommand.
login	<i>user_name</i>	Specifies the user's name as registered on the remote host.
password	<i>password</i>	Specifies the user's password as registered on the remote host. If you specify a password and the <code>.netrc</code> file can be read by anyone other than the user who issued the <code>ftp</code> command, FTP aborts the login process. If you do not specify a password and the remote host requires one, FTP prompts you for the password.
account	<i>account_password</i>	Not supported.
macdef	<i>macro_name</i>	Defines an FTP macro. The macro definition begins on the line following the <code>macdef</code> keyword and continues to the next blank line or the end-of-file marker. The rules for defining a macro in the <code>.netrc</code> file are the same as those for defining a macro with the <code>macdef</code> subcommand. If you define a macro with the name <code>init</code> , this macro is automatically executed as the last step in the automatic login process for the remote host. (See “Defining and Using Macros” later in this chapter for additional information about FTP macros.)

In the `.netrc` file, each keyword and its corresponding value must be separated from the next keyword by a space, a blank line, or a tab. If an entry includes a macro

definition, a blank line or an end-of-file marker indicates the end of the macro definition. (For information about macros, see the section “[Defining and Using Macros](#)” later in this chapter.)

The following example illustrates a typical `.netrc` file. It includes automatic login entries for two remote hosts, `atlanta` and `dallas`. The entry for `atlanta` includes a macro definition for one macro, named `init`. This macro is executed as the last step in the automatic login process for the remote host named `atlanta`. It is separated from the next automatic login entry by a blank line.

```
machine    atlanta
login      Susan_Smith
password   myword
macdef     init
           sunique
           runique

machine    dallas
login      Susan_Smith
```

When the automatic login process reads the `.netrc` file, it searches for the `machine` keyword with a `host` value that matches the remote host that you specified when you opened the connection. The automatic login process then uses the values for the remaining keywords until it reaches the next `machine` keyword. If the `.netrc` file does not include required login data, the automatic login process prompts you for this data. For example, if the remote host requires a password and the `.netrc` file does not specify it, FTP prompts you for the password.

Opening a Proxy Connection

A *proxy connection* is a concurrent connection between two remote hosts. The `proxy` subcommand allows you to open a proxy connection and issue subcommands relevant to the proxy connection. The `proxy` subcommand has the following format.

```
ftp> proxy ftp_subcommand
```


To open a proxy connection, you issue the `proxy` subcommand with the `open` subcommand. This subcommand generates prompts for logging in to the remote host on the proxy connection, as illustrated in the following example.

```
ftp> proxy open system_b
Connected to system_b
220 system_b FTP server (FTP 1.0 for Stratus STCP) ready.
(Compatible with OS TCP/IP)
Name (system_b:Susan_Smith.Marketing): sue
331 Password required for sue.
Password:
230 User Susan_Smith.Marketing logged in.
ftp>
```

When a proxy connection is open, only subcommands issued with the `proxy` subcommand affect the proxy connection. The connection that you open with the `host` argument of the `ftp` command or with the `open` subcommand is called the *primary connection*. Subcommands issued without the `proxy` subcommand affect the primary connection, not the proxy connection. Most FTP sessions involve only a primary connection; this type of connection is assumed in the examples in this manual unless otherwise specified.

The `ftp_subcommand` argument specifies the subcommand that you want to issue for the proxy connection. The subcommands that you can issue are a subset of the FTP subcommands. To see a list of the subcommands that you can issue for a proxy connection, issue the `proxy` subcommand followed by a question mark (?), as illustrated in the following example.

```
ftp> proxy ?
Commands may be abbreviated.  Commands are:
```

	delete	mdelete	proxy	send
?		mdir	put	
	dir	mget	pwd	size
append	disconnect	mkdir		sequential
ascii	form	mls	quote	status
	get	mode	recv	struct
binary		modtime	remotehelp	sunique
image		mput	rename	system
	help	nlist	reset	tenex
case	lappend	nmap	rhelph	type
cd	literal	ntrans	rmdir	user
cdup	ls	open	rstatus	
close		passive	runique	

Some of the FTP subcommands function differently when you issue them with the `proxy` subcommand. [Table 2-3](#) lists these subcommands and describes the differences in how they function.

Table 2-3. FTP Subcommands in Proxy Connections

Subcommand	Description of Differences
<code>open</code>	Does not define new macros if automatic login is used and the automatic login file contains macro definitions
<code>close</code> <code>disconnect</code>	Does not erase existing macro definitions
<code>send</code> <code>put</code> <code>append</code> <code>mput</code>	Transfers files from the remote host on the proxy connection to the remote host on the primary connection
<code>recv</code> <code>get</code> <code>mget</code>	Transfers files from the remote host on the primary connection to the remote host on the proxy connection

NOTE _____

If you use the `proxy` subcommand, the remote host for the proxy connection must support the FTP protocol command `PASV`. (To determine whether a remote host supports this command, you can issue the `remotehelp` subcommand, which lists the FTP protocol commands that the remote host supports.)

Closing a Connection

The `close` and `disconnect` subcommands close a connection with a remote host. These subcommands perform the same function; you can use them interchangeably. When you close a connection, you do not end the FTP session. When the `ftp>` prompt appears, you can issue another FTP subcommand.

The following examples illustrate the `close` and `disconnect` subcommands.

```
ftp> close
system_a:221 Goodbye.

ftp> disconnect
system_a:221 Goodbye.
```

If a proxy connection is also open, you must close the proxy connection by issuing the `proxy` subcommand with the `close` or `disconnect` argument, as illustrated in the following example.

```
ftp> proxy close
system_b:221 Goodbye.
```

Displaying Status Information

The status subcommands provide information about the current status of the FTP session and the connection with the remote host. [Table 2-4](#) lists and describes the status subcommands. They are listed in the order in which they are presented in this section.

Table 2-4. FTP Subcommands for Displaying Status Information

Subcommand	Description
<code>status</code>	Displays the status of the current FTP session
<code>rstatus</code>	Displays the status of the current connection with the remote host
<code>system</code>	Indicates the system type of the remote host

This section discusses the following topics.

- [“Displaying the Status of the FTP Session”](#)
- [“Displaying the Status of the Current Connection”](#)
- [“Displaying the System Type of the Remote Host”](#)

Displaying the Status of the FTP Session

The `status` subcommand displays information about the current FTP session. The status message indicates the name of the remote host attached to the primary connection and, if applicable, the name of the remote host attached to the proxy connection. Status information also includes the name of the TCP device and the current settings for all of the FTP modes. Mode settings are described in the section [“Setting Modes”](#) later in this chapter.

The following example illustrates the `status` subcommand and sample output.

```
ftp> status
Connected to system_a.admin.myfirm.com.
No proxy connection.
Mode: stream; Type: ascii; Form: non-print; Structure: file
Verbose: on; Bell: off; Prompting: on; Globbing: on
Store unique: off; Receive unique: off
Case: off; CR stripping: on
Ntrans: off
Nmap: off
Debug: off
Hash mark printing: off; Use of PORT cmds: on
```

Displaying the Status of the Current Connection

The `rstatus` subcommand displays the status of the current connection with the remote host. The status message indicates the name of the remote host, the login name of the current user, the settings for certain modes, and the data connection if one exists. A data connection is opened when you issue a file-transfer subcommand.

The following example illustrates the `rstatus` subcommand and sample output.

```
ftp> rstatus
211- system_a FTP server (FTP 1.0 for Stratus STCP) status:
    Connected to system_x
    Logged in as Susan_Smith.FTP_User
    TYPE: ASCII; FORM: Nonprint; STRUcture: File; transfer
MODE: Stream
    No data connection
211 End of status
```

Displaying the System Type of the Remote Host

The `system` subcommand indicates the system type of the remote host. The following example illustrates the `system` subcommand. The remote host is running a VOS system.

```
ftp> system
215 VOS
```

Setting Modes

An FTP mode specifies a characteristic of the FTP session, such as the display or nondisplay of messages returned by the server on the remote host. Each FTP mode has a default setting and a subcommand for changing the setting during an FTP

session. You can also change the default setting for some of the FTP modes when you issue the `ftp` command.

Three of the file-transfer modes (transmission, format, and structure) have fixed settings. A fixed setting is a default setting that is the only setting for the mode; it cannot be changed in the Stratus implementation of FTP. If you issue the subcommands for any of these modes (`mode`, `form`, and `struct`, respectively), FTP displays the fixed setting for the mode. In addition, the CR-stripping mode is permanently on; it cannot be changed.

[Table 2-5](#) lists and describes the subcommands for changing FTP modes or displaying their fixed settings. The subcommands are listed in the order in which they are presented in this section. This order corresponds to the status message, which is described in the section “[Displaying Status Information](#)” earlier in this chapter.

Table 2-5. FTP Subcommands for Setting Modes (Page 1 of 2)

Subcommand	ftp Command Line/Form Argument	Default Setting	Description
ascii image binary sequential tenex type	None	ascii	Changes the file-transfer type
mode	None	stream	Displays the fixed setting for file-transfer transmission mode
form	None	non-print	Displays the fixed setting for file-transfer format mode
struct	None	file	Displays the fixed setting for file-transfer structure mode
verbose	-no_verbose	On	Changes verbose mode
bell	None	Off	Changes bell mode
prompt	-no_interactive	On	Changes interactive-prompting mode
glob	-no_globbing	On	Changes global file-name-expansion mode
sunique	None	Off	Changes store-unique mode

Table 2-5. FTP Subcommands for Setting Modes (Page 2 of 2)

Subcommand	ftp Command Line/Form Argument	Default Setting	Description
runique	None	Off	Changes receive-unique mode
case	None	Off	Changes case-mapping mode
cr	None	On	Permanently set on
ntrans	None	Off	Changes character-translation mode
nmap	None	Off	Changes file-name-mapping mode
debug	-debug	Off	Changes debugging mode
hash	None	Off	Changes hash-mark mode
sendport	None	On	Changes sendport mode
passive	None	Off	Changes to passive mode

This section discusses the following topics.

- [“File-Transfer Modes”](#)
- [“Verbose Mode”](#)
- [“Bell Mode”](#)
- [“Interactive-Prompting Mode”](#)
- [“Global File-Name-Expansion Mode”](#)
- [“Store-Unique and Receive-Unique Modes”](#)
- [“Case-Mapping Mode”](#)
- [“CR-Stripping Mode”](#)
- [“Character-Translation Mode”](#)
- [“File-Name-Mapping Mode”](#)
- [“Debugging Mode”](#)
- [“Hash-Mark Mode”](#)
- [“Sendport Mode”](#)
- [“Passive Mode”](#)

File-Transfer Modes

The file-transfer mode specifies the current file-transfer type. When you issue a subcommand to transfer a file to or from the local host, FTP creates a file of the type specified by the current file-transfer type. (The file-transfer type is also called the *representation type*.) You can set the file-transfer type to one of the following standard types.

- *ASCII* for transferring text files
- *Image* or *binary* for transferring nontext files
- *Tenex* for transferring files to or from a Tenex machine

In addition to these standard file-transfer types, Stratus provides a nonstandard file-transfer type, *sequential*, which supports transfers of sequential files. This file-transfer type is for file transfers where the local host and the remote host are both Stratus computers.

The default file-transfer type is ASCII. To change the file-transfer type, you can issue one of the subcommands listed in [Table 2-6](#). If you issue the `type` subcommand without specifying a file-transfer type, FTP displays the current file-transfer type.

Table 2-6. FTP Subcommands for Setting the File-Transfer Type (Page 1 of 2)

Source File	File-Transfer Type	Subcommand
Text file	ASCII	Issue one of the following commands: ftp> <code>ascii</code> ftp> <code>type ascii</code>
Nontext file	Image or binary	Issue one of the following commands: ftp> <code>image</code> ftp> <code>type image</code> ftp> <code>type image 0</code> ftp> <code>binary</code> ftp> <code>type binary</code> ftp> <code>type binary 0</code>
Nontext file, fixed-length records	Image or binary	Issue one of the following commands: ftp> <code>image record_size</code> ftp> <code>type image record_size</code> ftp> <code>binary record_size</code> ftp> <code>type binary record_size</code>

Table 2-6. FTP Subcommands for Setting the File-Transfer Type (Page 2 of 2)

Source File	File-Transfer Type	Subcommand
File transferred to or from a Tenex machine	Tenex	Issue one of the following commands: ftp> tenex ftp> type tenex
Sequential file (Stratus-to-Stratus only)	Sequential	Issue one of the following commands: ftp> type seq ftp> type sequential ftp> seq ftp> sequential

NOTE

You can transfer a text file while the file-transfer type is set to image or binary without corrupting the contents of the transferred file. However, if you transfer a nontext file while the file-transfer type is set to ASCII, the contents of the transferred file are unpredictable.

The *record_size* argument for the `type image` or `type binary` subcommand specifies the record size for the destination file. You can specify a record size of 0 through 32767. Once you specify a record size, it is applied to all file transfers until you change it.

For file transfers between Stratus hosts, FTP supports the following Stratus file types: sequential, extended sequential, stream, and fixed (fixed-length records). FTP does not directly support file transfers for the Stratus relative and indexed file types. There is, however, an indirect procedure for transferring files of these types, using the sequential file-transfer type.

The sequential file-transfer type is a nonstandard file-transfer type that supports transfers of binary sequential files, which cannot be transferred as ordinary sequential files (that is, with the ASCII file-transfer type). The following procedure describes the steps for using the sequential file-transfer type to preserve the file characteristics in file transfers involving any Stratus file type. This procedure is only applicable when both hosts are Stratus hosts.

1. At operating-system command level in the directory in which the source file is located, create a save file by issuing the OpenVOS `save` command for the file to be transferred and specifying a port name that is attached to a non-existent file (`save` creates it as a sequential file).

2. Within FTP, set the file-transfer type to sequential by issuing the `type seq` subcommand. This ensures that the file is transferred in binary mode and is created as a VOS sequential file.
3. Transfer the sequential file by issuing the appropriate file-transfer subcommand.
4. At operating-system command level in the directory in which the destination file is located, restore the file by issuing the OpenVOS `restore` command.

NOTICE

If you attempt to use the sequential file-transfer type when the transfer is not between Stratus hosts, the results are unpredictable and may be undesirable.

To transfer an extended sequential file from a VOS system to a remote system, the following rules apply:

- If an extended sequential file is **larger** than 2 GB, you can transfer it to the following:
 - A VOS module that supports extended sequential files. To transfer the file, create an empty extended sequential file on the target module. Set the file's maximum record size and extent size, and then name the new file as the target of the `append` subcommand. If you use the `put` subcommand, the `ftp` command creates a sequential file, but the transfer fails if you attempt to write more than 2 GB of data to the file from any source.
 - A non-VOS system that supports stream files larger than 2 GB. To transfer the file, use the `put` subcommand. The transfer results in a stream file.
- If an extended sequential file is **smaller** than 2 GB, you can transfer it to the following:
 - A VOS module that supports extended sequential files. To transfer the file, use the `put` subcommand. The transfer results in a sequential file.
 - A VOS module that does not support extended sequential files. To transfer the file, use the `put` subcommand. The transfer results in a sequential file.
 - A non-VOS system. To transfer the file, use the `put` subcommand. The transfer results in a stream file.

To transfer an extended sequential file from a remote system to a VOS system that supports extended sequential files, create an empty extended sequential file on the target VOS module. Set the file's maximum record size and extent size, and then name the new file as the target of the `ftp lappend` subcommand. If you use the `get` subcommand, the `ftp` command creates a sequential file, but the transfer fails if you attempt to write more than 2 GB of data to the file from any source.

For more information about extended sequential files, see the description of the `create_file` command in the *OpenVOS Commands Reference Manual* (R098)

To preserve the integrity of Stratus file types during transfers between Stratus hosts, use the corresponding file-transfer types presented in [Table 2-7](#).

Table 2-7. File-Transfer Types for Transfers between Stratus Hosts

Source-File Type	Destination-File Type	File-Transfer Type	Subcommand
Sequential or extended sequential	Sequential or extended sequential	ASCII	Issue one of the following commands: <code>ftp> ascii</code> <code>ftp> type ascii</code>
Sequential save or extended sequential	Sequential save or extended sequential	Sequential	Issue one of the following commands: <code>ftp> seq</code> <code>ftp> type seq</code>
Stream	Stream	Image or binary	Issue one of the following commands: <code>ftp> image</code> <code>ftp> type image</code> <code>ftp> type image 0</code> <code>ftp> binary</code> <code>ftp> type binary</code> <code>ftp> type binary 0</code>
Fixed-length records	Fixed-length records	Image or binary	Issue one of the following commands: <code>ftp> type image record_size</code> <code>ftp> type binary record_size</code>

NOTICE _____

[Table 2-7](#) indicates the FTP file-transfer type for each Stratus file type. If you use a file-transfer type other than what is indicated for the source-file type in the table, the results are unpredictable.

There are three subcommands for displaying the fixed-setting file-transfer modes. These modes cannot be changed in the Stratus implementation of FTP. [Table 2-8](#) lists and describes the subcommands for displaying the fixed-setting file-transfer modes.

Table 2-8. FTP Subcommands for Displaying Fixed-Setting File-Transfer Modes

Subcommand	Fixed Setting	Description
<code>mode</code>	<code>stream</code>	Transmits a file as a stream of bytes.
<code>form</code>	<code>non-print</code>	Retains, in the transferred file, the ASCII characters for vertical format control, such as formfeed and newline. This mode applies only when the current file-transfer type is ASCII.
<code>struct</code>	<code>file</code>	Transmits a file as a stream of bytes with no recognition of internal structure, such as records or pages.

Verbose Mode

When verbose mode is on, FTP displays the responses that the remote host returns when you issue a subcommand. Responses include informational and error messages. The default setting is on. When you begin an FTP session, you can turn off verbose mode by specifying the `-no_verbose` argument when you issue the `ftp` command, as illustrated in the following example.

```
ftp -no_verbose
```

During an FTP session, you can change the setting for verbose mode by issuing the `verbose` subcommand, as illustrated in the following example.

```
ftp> verbose
```

The examples throughout this chapter assume that verbose mode is on.

Bell Mode

When bell mode is on, your terminal beeps whenever a file transfer is completed. The default setting is off. You can change the setting for bell mode by issuing the `bell` subcommand, as illustrated in the following example.

```
ftp> bell
```

Interactive-Prompting Mode

When interactive-prompting mode is on, prompting occurs during file transfer or deletion if multiple files are affected, or if a file overwrites an existing file on the local host. Before each transfer or deletion, you are prompted to verify that you want to apply the current subcommand to a specified file. When interactive-prompting mode is off, file transfer or deletion occurs without prompting.

The default setting is on. When you begin an FTP session, you can turn off interactive-prompting mode by specifying the `-no_interactive` argument when you issue the `ftp` command, as illustrated in the following example.

```
ftp -no_interactive
```

During an FTP session, you can change the setting for interactive-prompting mode by issuing the `prompt` subcommand, as illustrated in the following example.

```
ftp> prompt
```

If interactive-prompting mode is on, prompting occurs in the following situations.

- When you attempt to transfer multiple files (using the subcommand `mput` or `mget`), you are prompted to verify that you want to transfer each file.
- When you attempt to delete multiple files (using the subcommand `mdelete`), you are prompted to verify that you want to delete each file.
- When you attempt to transfer a file from the remote host to the local host (using the subcommand `recv`, `get`, or `mget`), and the file has the same name as an existing file on the local host, you are prompted to verify that you want to overwrite the existing file.

When you are prompted for verification, you respond by typing `y` (yes) or `n` (no).

The following example uses the `mdelete` subcommand to illustrate interactive-prompting mode. A prompt appears on the screen before each deletion, and a confirmation message appears after each deletion.

```
ftp> mdelete *
mdelete acctng.cobol? y
250 DELE command successful.
mdelete make_reports.cobol? y
250 DELE command successful.
mdelete help_file? n
```

Global File-Name-Expansion Mode

When global file-name-expansion mode is on, you can use abbreviations, defined in the `abbreviations` file in your home directory, and star names. FTP expands each name and applies the current subcommand to all files with names that match the star name or abbreviation. (Global file-name expansion is also called *globbing*.) When global file-name-expansion mode is off, FTP interprets every name literally; it does not expand abbreviations or star names. The default setting is on.

When you begin an FTP session, you can turn off global file-name-expansion mode by specifying the `-no_globbing` argument when you issue the `ftp` command, as illustrated in the following example.

```
ftp -no_globbing
```

During an FTP session, you can change the setting for global file-name-expansion mode by issuing the `glob` subcommand, as illustrated in the following example.

```
ftp> glob
```

If you use a star name to specify file names on the local host, and global file-name-expansion mode is on, FTP expands the star name and selects all files with names that match the star name. For example, if you issue the following subcommand, the FTP client expands the star name to specify all files in the current directory with the suffix `.cobol` and sends the files to the remote host.

```
ftp> mput *.cobol
```

Using the same example, if global file-name-expansion mode is off, FTP responds with a message indicating that there is no file named `*.cobol`.

You can use star names to specify names on the local and remote hosts. However, if you use a star name with the subcommands `mdelete` and `mget`, the star name is expanded on the remote host according to the **remote host's rules** for star-name expansion.

You can use abbreviations for specifying names on the local host only. If you use an abbreviation, the operating system performs a substitution based on the definitions in your `abbreviations` file.

For example, it would be appropriate to use an abbreviation to specify a source file for the `put` subcommand but not to specify the destination file. The `put` subcommand transfers a file from the local host to the remote host. The first argument specifies the name of the source file, which is interpreted on the local host, and the second argument specifies the name of the destination file, which is interpreted on the remote host.

The following example illustrates an appropriate use of an abbreviation, `dly_up`, which is defined in the user's `abbreviations` file as the file name `daily_accounting_update`.

```
ftp> put dly_up daily_accounting_update
```

The following example illustrates an inappropriate use of the abbreviation from the previous example. If you do not specify a second argument, FTP uses the source file name as the destination file name.

```
ftp> put dly_up
```

Using an abbreviation in this example does not yield the expected results. On the local host, the operating system substitutes the full path name for the abbreviation and sends the appropriate file to the remote host. However, the remote host interprets the abbreviation literally and creates the file with the file name `dly_up`.

(For additional information about the `abbreviations` file, see the *OpenVOS Commands User's Guide* (R089).)

Store-Unique and Receive-Unique Modes

Store-unique mode and receive-unique mode perform similar functions. When these modes are on, they ensure that transferred files have unique file names. The default setting for both modes is off.

During an FTP session, you can change the setting for store-unique mode by issuing the `sunique` subcommand and for receive-unique mode by issuing the `runique` subcommand, as illustrated in the following examples.

```
ftp> sunique
Receive unique on.
ftp> runique
Store unique on.
```

When store-unique mode is on, FTP ensures that each file that is transferred to the remote host is assigned a unique file name. If the name of a file transferred to the remote host matches the name of an existing file on the remote host, the FTP server on the remote host changes the file name to make it unique and returns a message indicating the unique file name. The FTP server on the remote host determines the way in which uniqueness is implemented.

NOTE

If you turn on store-unique mode, the remote host must support the FTP protocol command `STOU`. If the remote host does not support this command, the `send`, `put`, and `mput` subcommands will not work when store-unique mode is on. (To determine whether a remote host supports this command, you can issue the `remotehelp`

subcommand, which lists supported FTP protocol commands.)

When receive-unique mode is on, FTP ensures that each file that is transferred to the local host is assigned a unique file name. If the name of a file transferred to the local host matches the name of an existing file on the local host, FTP changes the file name to make it unique and returns a message indicating the unique file name. To make the file name unique, FTP appends the characters `.n` to the name, where *n* represents an integer from 1 to 99. FTP uses these numbers in sequence until the maximum of 99 has been reached. When there are no more numbers available, FTP generates an error message and the file transfer does not occur.

Case-Mapping Mode

When case-mapping mode is on, FTP assigns file names composed of all lowercase characters when transferring files to the local host with the `mget` subcommand. If a source file name contains uppercase characters, FTP changes them to the corresponding lowercase characters in the destination file name. The default setting is off. During an FTP session, you can change the setting for this mode by issuing the `case` subcommand, as illustrated in the following example.

```
ftp> case
```

CR-Stripping Mode

The CR-stripping mode allows your local FTP to strip out the ASCII CR characters from an ASCII file when it is transferred to your local host. The CR-stripping mode only affects ASCII files received on your local host, not files that you transfer to another system. It is permanently on by default. During an FTP session, issuing the `cr` subcommand has no effect on this default setting.

Character-Translation Mode

Character-translation mode is applied if you do not specify a file name for the destination file when you transfer a file. When character-translation mode is on, FTP assigns a file name to the destination file by replacing specified characters in the source file name with the corresponding translation characters in the destination file name. The default setting is off.

During an FTP session, you can change the setting for character-translation mode by issuing the `ntrans` subcommand, which has the following format.

```
ftp> ntrans [ inchars [ outchars ] ]
```

The *inchars* argument specifies the characters in the source file name that you want to translate. The *outchars* argument specifies the corresponding translation

characters in the destination file name. For example, to change the asterisks (*) in a source file name to underlines (_), you would issue the `ntrans` subcommand with the following arguments.

```
ftp> ntrans * _
```

When you issue a file-transfer subcommand (`send`, `put`, `mput`, `recv`, `get`, or `mget`) without specifying a file name for the destination file, FTP uses the source file name to assign a file name to the destination file. If character-translation mode is on, FTP assigns the file name to the destination file by changing the *inchars* characters in the source file name to the corresponding *outchars* characters.

The translation specified by an `ntrans` subcommand remains in effect until you specify a different translation by issuing another `ntrans` subcommand. To turn off character-translation mode, you issue the `ntrans` subcommand without arguments.

The following example illustrates the use of character-translation mode to translate dollar signs (\$) in the source file name to underlines (_) in the destination file name.

```
ftp> ntrans $ _
ftp> put s$myfile
250 PORT command successful.
150 Opening ASCII mode data connection for s_myfile.
226 Transfer complete.
3074 bytes sent in 1.61 seconds (1.86 Kbytes/s).
```

Message 150 indicates the file name of the destination file. As shown in the example, FTP translates the source file name, `s$myfile`, to `s_myfile`, which is the destination file name.

You can use character-translation mode to delete characters in the destination file name. If there are more characters in the *inchars* argument than in the *outchars* argument, FTP deletes the characters in the *inchars* argument that do not have a corresponding character in the *outchars* argument.

The following example illustrates the use of character-translation mode to delete dollar signs in destination file names.

```
ftp> ntrans $
ftp> put s$myfile
250 PORT command successful.
150 Opening ASCII mode data connection for smyfile.
226 Transfer complete.
3074 bytes sent in 1.61 seconds (1.86 Kbytes/s).
```

In this example, FTP creates the destination file name, `smyfile`, by using all of the characters in the source file name except for the \$ character.

File-Name-Mapping Mode

File-name-mapping mode is applied if you do not specify a file name for the destination file when you transfer a file. When file-name-mapping mode is on, FTP maps specified components in an input file name to corresponding components in the output file name. The default setting is off.

During an FTP session, you can change the setting for file-name-mapping mode by issuing the `nmap` subcommand, which has the following format.

```
ftp> nmap [inpattern outpattern]
```

The *inpattern* argument specifies the components in the source file name, and the *outpattern* argument specifies the mapping of the corresponding components in the destination file name. If you issue the `nmap` subcommand without arguments, file-name-mapping mode is turned off.

When you issue a file-transfer subcommand without specifying a file name for the destination file, FTP maps the source file-name components specified in the *inpattern* argument to match the pattern specified in the *outpattern* argument. FTP then uses the mapped file name as the name of the destination file.

NOTE

The characters `$`, `[`, `]`, and `,` are reserved characters when used in `nmap` arguments. If you want to use a reserved character as a literal, you must precede the character with a reverse slant (`\`).

When specifying a pattern, you use the character sequences `$1` through `$9` to indicate file-name components in the *inpattern* and *outpattern* arguments. The following example illustrates the use of file-name-mapping mode to append the suffix `.txt` to file names during file transfers.

```
ftp> nmap $1 $1.txt
ftp> put myfile
250 PORT command successful.
150 Opening ASCII mode data connection for myfile.txt.
226 Transfer complete.
3074 bytes sent in 1.61 seconds (1.86 Kbytes/s).
```

Message 150 indicates the name of the destination file. As shown in the example, FTP maps the source file name, `myfile`, to `myfile.txt`, which is the destination file name.

You can use the sequence \$0 in the *outpattern* argument to specify the original file name, as illustrated in the following example.

```
ftp> nmap $1.txt.$2 $0.backup
ftp> put myfile.txt.new
250 PORT command successful.
150 Opening ASCII mode data connection for
myfile.txt.new.backup.
226 Transfer complete.
3074 bytes sent in 1.61 seconds (1.86 Kbytes/s).
```

During the transfer, FTP maps the source file name, `myfile.txt.new`, to `myfile.txt.new.backup`, which is the destination file name.

You can include blank spaces in the *outpattern* argument, as illustrated in the following example.

```
ftp> nmap $1 $1 text
ftp> put myfile
250 PORT command successful.
150 Opening ASCII mode data connection for myfile text.
226 Transfer complete.
3074 bytes sent in 1.61 seconds (1.86 Kbytes/s).
```

During the transfer, FTP maps the source file name, `myfile`, to `myfile text`, which is the destination file name.

You can specify two alternative patterns for the *outpattern* argument by using the `[map1,map2]` format. FTP applies the *map1* pattern if the *map1* component in the source file is not a null string; otherwise, FTP applies the *map2* pattern. The following example illustrates the use of the `nmap` subcommand to specify this type of mapping.

```
ftp> nmap $1.$2.$3 [$1,$2].[$2,file]
```

[Table 2-9](#) lists the destination file names that result when FTP applies the file-name mapping specified in the previous example.

Table 2-9. Sample Output from File-Name-Mapping Mode

Source File Name	Destination File Name
myfile.data	myfile.data
myfile.data.old	myfile.data
myfile	myfile.file
.myfile	myfile.myfile

Debugging Mode

When debugging mode is on, each time you issue a subcommand, FTP displays the actual FTP protocol commands that it sends to the remote host. The lines generated by debugging mode begin with the characters `--->`. The default setting is off. When you begin an FTP session, you can turn on debugging mode by specifying the `-debug` argument when you issue the `ftp` command, as illustrated in the following example.

```
ftp -debug
```

During an FTP session, you can change the setting for debugging mode by issuing the `debug` subcommand, as illustrated in the following example.

```
ftp> debug
```

The following example illustrates the use of the `debug` subcommand to turn on debugging mode. In this example, a connection has been opened, and the `put` subcommand sends a file to the remote host. The debugging messages display the actual FTP protocol commands sent to the remote host: `PORT` (establishes the data connection) and `STOR` (sends the file).

```
ftp> debug
Debugging on.
ftp> put sort_list.cm
---> PORT 192,9,200,33,10,139
250 PORT command successful.
---> STOR sort_list.cm
150 Opening ASCII mode data connection for sort_list.cm.
226 Transfer complete.
3074 bytes sent in 1.69 seconds (1.78 Kbytes/s)
```

NOTE

If you enter your password while debugging mode is on, the password will appear on the screen.

Hash-Mark Mode

When hash-mark mode is on, FTP displays a number sign (#, also called a hash mark) each time a data block (4096 bytes) is transferred. The default setting is off. During an FTP session, you can change the setting for hash-mark mode by issuing the `hash` subcommand, as illustrated in the following example.

```
ftp> hash
Hash mark printing on (4096 bytes/hash mark).
```

Sendport Mode

When sendport mode is on, FTP sends the FTP protocol command `PORT` to the remote host when establishing a connection for a file transfer. The `PORT` command tells the server on the remote host which port to use for transferring data. The default setting is on. During an FTP session, you can change the setting for sendport mode by issuing the `sendport` subcommand, as illustrated in the following example.

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

If sendport mode is off, FTP does not send the `PORT` command and the default data port is used. The default data port is one less than the command port. The default command port is 21, so typically, the default data port is 20. (To specify a command port, you issue the `open` subcommand with the `port` argument.)

NOTE

On some remote hosts, FTP does not accept the `PORT` command. If you open a connection with a remote host that does not accept this command, you must turn off sendport mode before issuing a file-transfer subcommand.

Passive Mode

When passive mode is on, the server waits for the client to establish a connection with it instead of attempting to connect to a client-specified port. The server responds with the IP address and the port number it is listening on.

During an FTP session, you can change the setting for passive mode by issuing the `passive` subcommand, as illustrated in the following example.

```
ftp> passive
Passive mode on.
```

Active mode is the default; if you are in passive mode, enter the `passive` subcommand again to switch back into active mode, as illustrated in the following example.

```
ftp> passive
Passive mode on.
ftp> passive
Passive mode off.
```

Performing File Operations

File operations provide access to the files on the remote host. The file-operation subcommands transfer files to and from the local host. They also delete and rename files on the remote host. The arguments for some of the subcommands specify file names. When you specify a file name, you can use full or relative path names as defined on the system on which the file resides. Before using the file-operation subcommands, you must open a connection with a remote host and set the modes that you want to use, as described earlier in this chapter.

NOTE

When FTP transfers a file, it sends or receives a copy of the file. The source file remains unchanged.

[Table 2-10](#) lists and describes the file-operation subcommands. The subcommands are listed in the order in which they are presented in this section.

Table 2-10. FTP Subcommands for File Operations (Page 1 of 2)

Subcommand	Description
send put	Transfers a file from the local host to the remote host
append	Transfers a file from the local host to the remote host and appends the file to the end of an existing file
lappend	Transfers a file from the remote host to the local host and appends the file to the end of an existing local file
mput	Transfers one or more files from the local host to the remote host
recv get	Transfers a file from the remote host to the local host
mget	Transfers one or more files from the remote host to the local host
rename	Renames a file on the remote host

Table 2-10. FTP Subcommands for File Operations (Page 2 of 2)

Subcommand	Description
<code>delete</code>	Deletes a file on the remote host
<code>mdelete</code>	Deletes one or more files on the remote host
<code>modtime</code>	Indicates the time at which a file on the remote host was last modified
<code>size</code>	Indicates the size, in bytes, of a file on the remote host

This section discusses the following topics.

- [“Sending Files to the Remote Host”](#)
- [“Appending a Local File”](#)
- [“Appending a Remote File”](#)
- [“Receiving Files from the Remote Host”](#)
- [“Renaming Files on the Remote Host”](#)
- [“Deleting Files on the Remote Host”](#)
- [“Displaying the Last Modification Time”](#)
- [“Displaying File Size”](#)

Sending Files to the Remote Host

The `send`, `put`, `append`, and `mput` subcommands transfer files from the local host to the remote host. These subcommands have the following format.

```
ftp> send local_file [remote_file]

ftp> put local_file [remote_file]

ftp> append local_file [remote_file]

ftp> mput local_file...
```

The `local_file` argument specifies the name of the source file that you want to transfer. The `remote_file` argument specifies the name of the destination file. If you do not specify a value for the `remote_file` argument, FTP transfers the file to the current directory on the remote host, using the file name specified in the `local_file` argument.

You can ensure that the destination file name is unique by turning on store-unique mode. If store-unique mode is on, FTP uses a unique file name when it creates the destination file. If the destination file name matches the name of an existing file, FTP

changes the destination file name to make it unique. If store-unique mode is off, FTP overwrites an existing file that has the same name as the destination file. (See the section “[Store-Unique and Receive-Unique Modes](#)” earlier in this chapter for additional information about these modes and required support on the remote host for the FTP protocol command STOU.)

The `send` and `put` subcommands perform the same function; you can use them interchangeably. These subcommands transfer a single file from the local host to the remote host.

The following example illustrates the `send` subcommand. In this example, the remote host is a UNIX[®] system. Upon execution of the `send` subcommand, the user issues the `dir` subcommand to ensure that the file appears in the directory on the remote host.

```
ftp> send test.a
250 PORT command successful.
150 Opening ASCII mode data connection for test.a.
226 Transfer complete.
138 bytes sent in 0.09 seconds (01.41 Kbytes/s)

ftp> dir test*
226 Transfer complete.
250 PORT command successful.
150 Opening ASCII mode data connection for test*.
total 7443
-rw-rw-rw-  1 sue      wheel      133 Nov 15 11:02 test.a
-rw-rw-rw-  1 sue      wheel    72710 Oct 26 12:32 test.b
-rw-rw-rw-  1 sue      wheel    59052 Sep 21 16:39 test.c
-rwxrwxrwx  1 sue      comm     23917 Sep 15 12:57 test.d
-rw-rw-rw-  1 sue      wheel   7450624 Sep 20 10:12 test.e
226 Transfer complete.
329 bytes received in 0.29 seconds (01.09 Kbytes/s)
```

Appending a Local File

The `append` subcommand transfers a file from the local host to the remote host and appends the file to the end of an existing file. FTP applies the following rules to determine the destination file.

- If you specify a value for the `remote_file` argument and a destination file with that name already exists, FTP appends the transferred file to the specified file. If the destination file does not exist, FTP transfers the source file to the current directory on the remote host, which creates a destination file with the name specified in the `remote_file` argument.
- If you do not specify a value for the `remote_file` argument, FTP looks in the current directory on the remote host for a destination file with a name that matches

the `local_file` argument and then appends the transferred file to it. If the destination file does not exist, FTP transfers the source file to the current directory on the remote host, creating a destination file with the name specified in the `local_file` argument.

The following example illustrates the `append` subcommand. Upon execution of the subcommand, the user issues the `dir` subcommand to ensure that the file was appended to the destination file successfully. If the size of the destination file increases by the number of bytes in the source file, the `append` operation can be considered successful.

```
ftp> append test.a
250 PORT command successful.
150 Opening ASCII mode data connection for test.a.
226 Transfer complete.
138 bytes sent in 0.06 seconds (02.13 Kbytes/s)

ftp> dir test*
250 PORT command successful.
150 Opening ASCII mode data connection for test*.
total 7443
-rw-rw-rw-  1 sue      wheel      266 Nov 15 11:03 test.a
-rw-rw-rw-  1 sue      wheel    72710 Oct 26 12:32 test.b
-rw-rw-rw-  1 sue      wheel    59052 Sep 21 16:39 test.c
-rwxrwxrwx  1 sue      comm     23917 Sep 15 12:57 test.d
-rw-rw-rw-  1 sue      wheel  7450624 Sep 20 10:12 test.e
226 Transfer complete.
329 bytes received in 0.28 seconds (01.11 Kbytes/s)
```

If you use the `append` subcommand to transfer a file to a Stratus remote host, the file to which you are appending the transferred file determines the organization of the appended file, regardless of the file-transfer type for the transfer. For example, if the file-transfer type for the transfer is `image 4096`, and the file to which you are appending a transferred file is a stream file, the appended file is stored as a stream file.

The `mput` subcommand transfers one or more files from the local host to the current directory on the remote host. This subcommand transfers only the files contained in a specified directory; it does not transfer any subdirectories.

To specify multiple files, separate each file name with a space. If global file-name-expansion mode is on, you can also use a star name to specify multiple files. (See the section “[Global File-Name-Expansion Mode](#)” earlier in this chapter for additional information about this mode.)

The following example illustrates the use of the `mput` subcommand to transfer two files.

```
ftp> mput acctng.dat make_reports.pm
```

If interactive-prompting mode is on, FTP prompts you to verify that you want to transfer each file affected by the `mput` subcommand. If interactive-prompting mode is off, `mput` transfers each affected file without prompting. (See “[Interactive-Prompting Mode](#)” earlier in this chapter for additional information about this mode.)

The following example illustrates the use of the `mput` subcommand while interactive-prompting mode is on. In this example, the subcommand transfers all files in the current directory on the local host that end in the suffix `.cobol`.

```
ftp> mput *.cobol
mput %s1#m110>Tools>Susan_Smith>acctng.cobol? y
250 PORT command successful.
150 Opening ASCII mode data connection for acctng.cobol.
226 Transfer complete.
158 bytes sent in 0.14 seconds (1.12 Kbytes/s)
mput %s1#m110>Tools>Susan_Smith>make_reports.cobol? y
250 PORT command successful.
150 Opening ASCII mode data connection for
make_reports.cobol.
226 Transfer complete.
79 bytes sent in 0.05 seconds (1.46 Kbytes/s)
```

Appending a Remote File

The `lappend` subcommand transfers a file from the remote host to the local host and appends the file to the end of an existing local file. This command is similar to the `append` subcommand, with the following differences:

- If the local file does not exist, `lappend` behaves in a similar manner to the `get` request.
- If the local file does exist, `lappend` appends data from the remote file to the local file.

The `lappend` request is useful in transferring data to unique VOS file types, such as relative files, extended sequential files, or files with embedded key indexes. VOS stream files are limited to a total size of 2 GB. To transfer a larger file (such as an extended sequential file or a stream file on a non-VOS system), create an empty file with the desired characteristics (for example, file type, extent type, indexes), and then use `lappend` to transfer records from the remote file to the new file. As part of the transfer, `ftp` builds embedded key indexes.

Relative or extended sequential files grow up to the limits imposed either by their extent size or, for extended sequential files, by the allocation factor, which is specified by the maximum record size when the file is created.

For more information about extended sequential files, see the description of the `create_file` command in the *OpenVOS Commands Reference Manual* (R098).

Receiving Files from the Remote Host

The `recv`, `get`, and `mget` subcommands transfer files from the remote host to the local host. These subcommands have the following format.

```
ftp> recv remote_file [local_file]

ftp> get remote_file [local_file]

ftp> mget remote_file...
```

The `remote_file` argument specifies the name of the source file that you want to transfer. The `local_file` argument specifies the name of the destination file. If you do not specify a value for the `local_file` argument, FTP transfers the file to the current directory on the local host, using the file name specified in the `remote_file` argument.

You can ensure that the destination file name is unique by turning on receive-unique mode. If receive-unique mode is on, FTP uses a unique file name when it creates the destination file. If the destination file name matches the name of an existing file, FTP changes the destination file name to make it unique. If receive-unique mode is off, FTP prompts you that the file already exists and then allows you to overwrite an existing file that has the same name as the destination file. (See the section “[Store-Unique and Receive-Unique Modes](#)” earlier in this chapter for additional information about these modes.)

The `recv` and `get` subcommands perform the same function; you can use them interchangeably. These subcommands transfer a single file from the remote host to the local host.

If you want to display the transferred file on the screen instead of writing it to a file, you specify a hyphen (-) as the value for the `local_file` argument. The following example illustrates the use of the `get` subcommand with the hyphen argument to display the file `test.a`.

```
ftp> get test.a -
```

The following example illustrates the `get` subcommand. Upon execution of the `get` subcommand, the user issues the operating-system command `list` to ensure that the file appears in the current directory on the remote host. (The `..` subcommand allows

you to issue an operating-system command during an FTP session. For information about this subcommand, see the section “[Issuing Operating-System Commands](#)” later in this chapter.)

```
ftp> get test.a
250 PORT command successful.
150 Opening ASCII mode data connection for test.a (4096
bytes).
226 Transfer complete.
3550 bytes received in 3.97 seconds (0.87 Kbytes/s)

ftp> .. list -files test*

Files: 1 Blocks: 46

w      46 test.a
```

The `mget` subcommand transfers one or more files from the remote host to the current directory on the local host. This subcommand transfers the files contained in a specified directory; it does not transfer any subdirectories.

To specify multiple files, separate each file name with a space. If global file-name-expansion mode is on, you can also use a star name to specify multiple files. (See the section “[Global File-Name-Expansion Mode](#)” earlier in this chapter for additional information about this mode.) Star names are expanded on the remote host according to the remote host’s rules for star-name expansion.

The following example illustrates the use of the `mget` subcommand to transfer two files.

```
ftp> mget acctng.dat make_reports.pm
250 PORT command successful.
150 Opening BINARY mode data connection for acctng.dat (4096
bytes).
226 Transfer complete.
3550 bytes received in 7.29 seconds (0.48 Kbytes/s)
250 PORT command successful.
150 Opening BINARY mode data connection for make_reports.pm
(4096 bytes).
226 Transfer complete.
3550 bytes received in 6.42 seconds (0.54 Kbytes/s)
```

The following example illustrates the use of the `mget` subcommand to transfer all files in the current directory on the remote host that have the `.cobol` suffix. Interactive prompting mode and global file-name-expansion mode are on. The star name

*.cobol produces the desired results because the remote host interprets the asterisk as a wildcard character.

```
ftp> mget *.cobol
mget acctng.cobol? y
250 PORT command successful.
150 Opening ASCII mode data connection for acctng.cobol (4096
bytes).
226 Transfer complete.
3074 bytes received in 2.51 seconds (1.20 Kbytes/s)
mget account_update.cobol? y

File already exists:
%s1#m110>Tools>Susan_Smith>reports>account_update.cobol
Delete it?? y
250 PORT command successful.
150 Opening ASCII mode data connection for
account_update.cobol (4096 bytes).
226 Transfer complete.
3074 bytes received in 2.51 seconds (1.20 Kbytes/s)
```

Renaming Files on the Remote Host

The `rename` subcommand renames files on the remote host. This subcommand has the following format.

```
ftp> rename from to
```

The *from* argument specifies the current file name, and the *to* argument specifies the new file name. The `rename` subcommand changes the name specified in the *from* argument to the name specified in the *to* argument. You cannot specify a star name with this subcommand, even if you are connected to a Stratus remote host.

The following example illustrates the use of the `rename` subcommand to rename the `accruals` file. The new name is `accruals.memo`. The responses from the remote host indicate that the subcommand was executed successfully.

```
ftp> rename accruals accruals.memo
350 File exists, ready for destination name
250 RNTD command successful.
```

Deleting Files on the Remote Host

The `delete` and `mdelete` subcommands delete files on the remote host. These subcommands have the following format.

```
ftp> delete remote_file

ftp> mdelete remote_file...
```

The `remote_file` argument specifies the file that you want to delete.

The `delete` subcommand deletes a single file. The following example illustrates the use of the `delete` subcommand to delete the `accruals.backup` file.

```
ftp> delete accruals.backup
250 DELE command successful.
```

The `mdelete` subcommand deletes one or more files. To specify multiple files, separate each file name with a space. If global file-name-expansion mode is on, you can also use a star name to specify multiple files. (See the section “[Global File-Name-Expansion Mode](#)” earlier in this chapter for additional information about this mode.) Star names are expanded on the remote host according to the remote host’s rules for star-name expansion.

The following example illustrates the use of the `mdelete` subcommand to delete the file `acctng.dat` and all files that end in the suffix `.cobol`.

```
ftp> mdelete acctng.dat *.cobol
```

If interactive-prompting mode is on, FTP prompts you to verify that you want to delete each file affected by the `mdelete` subcommand. If interactive-prompting mode is off, `mdelete` deletes each affected file without prompting.

The following example illustrates the use of the `mdelete` subcommand while interactive-prompting mode is on. In this example, the subcommand deletes all files in the current directory on the remote host that end in the suffix `.cobol`.

```
ftp> mdelete *.cobol
mdelete acctng.cobol? y
250 DELE command successful.
mdelete make_reports.cobol? y
250 DELE command successful.
```

Displaying the Last Modification Time

The `modtime` subcommand indicates the time at which a file on the remote host was last modified. This subcommand has the following format.

```
ftp> modtime file_name
```

The `file_name` argument specifies the file that you want to check, as shown in the following example.

```
ftp> modtime myfile
myfile 05/21/1990 19:51:09 GMT
```

The time is displayed as Greenwich Mean Time.

Displaying File Size

The `size` subcommand indicates the size, in bytes, of a file on the remote host. This subcommand has the following format.

```
ftp> size file_name
```

The `file_name` argument specifies the name of the file that you want to check, as shown in the following example.

```
ftp> size myfile
213 3548
```

The first integer, 213, is the message number; the second integer, 3548, is the size of the file, in bytes.

Performing Directory Operations

Directory operations provide access to the directories on the remote host. The directory-operation subcommands change the current directory, display the name of the current directory, list the contents of a directory, and create and delete directories. Before using the directory-operation subcommands, you must open a connection with a remote host, as described earlier in this chapter.

The *current directory* is the directory associated with your process. This directory is also called the *working directory*.

[Table 2-11](#) lists and describes the directory-operation subcommands. The subcommands are listed in the order in which they are presented in this section.

Table 2-11. FTP Subcommands for Directory Operations

Subcommand	Description
<code>pwd</code>	Displays the name of the current directory on the remote host
<code>cd</code>	Changes the current directory on the remote host
<code>cdup</code>	Changes the current directory on the remote host to the parent of the previous current directory
<code>lcd</code>	Changes the current directory on the local host
<code>dir</code> or <code>ls</code>	Lists the contents of a directory on the remote host, with details
<code>nlist</code>	Lists the contents of a directory on the remote host, without details
<code>mdir</code>	Lists the contents of one or more directories on the remote host, with details
<code>mls</code>	Lists the contents of one or more directories on the remote host, without details
<code>mkdir</code>	Creates a directory on the remote host
<code>rmdir</code>	Deletes a directory on the remote host, if the directory is empty

This section discusses the following topics.

- [“Displaying the Current Directory”](#)
- [“Changing the Current Directory”](#)
- [“Listing the Contents of a Directory”](#)
- [“Creating and Deleting Directories”](#)

Displaying the Current Directory

The `pwd` subcommand displays the name of the current directory on the remote host. The name of this subcommand stands for *print working directory*. The following example illustrates the `pwd` subcommand.

```
ftp> pwd
257 "%sysa#m1>Admin>Susan_Smith" is current directory.
```

Changing the Current Directory

The `cd`, `cdup`, and `lcd` subcommands change the current directory. These subcommands have the following format.

```
ftp> cd remote_directory

ftp> cdup

ftp> lcd [local_directory]
```

The `cd` subcommand changes the current directory on the remote host. The *remote_directory* argument specifies the directory that you want to use. The *remote_directory* argument is required; FTP does not use the initial current directory as a default value for this argument.

The following example illustrates the use of the `cd` subcommand to change the current directory on the remote host to the directory specified by the relative path name <Accounting>Payroll.

```
ftp> cd <Accounting>Payroll
250 CWD command successful.
```

The `cdup` subcommand changes the current directory on the remote host to the parent of the previous current directory. For example, if the previous current directory was <Accounting>Payroll, the `cdup` subcommand changes the current directory to <Accounting>, as illustrated in the following example.

```
ftp> cdup
250 CWD command successful.
```

The `lcd` subcommand changes the current directory on the local host. The *local_directory* argument specifies the directory that you want to use. If you specify a value for the *local_directory* argument, FTP changes the current directory on the local host to the specified directory. If you do not specify a value for the *local_directory* argument, FTP changes the current directory to your home directory.

The following example illustrates the use of the `lcd` subcommand to change from the user's home directory to a subdirectory, `memos`.

```
ftp> lcd memos
Local directory now %s1#m110>Tools>Susan_Smith>memos
```


Listing the Contents of a Directory

The `dir`, `nlist`, `ls`, `mdir` and `mls` subcommands list the contents of directories on the remote host. These subcommands have the following format.

```
ftp> dir [remote_directory [local_file]]
```

```
ftp> nlist [remote_directory [local_file]]
```

```
ftp> ls [remote_directory [local_file]]
```

```
ftp> mdir remote_directory... local_file
```

```
ftp> mls remote_directory... local_file
```

The *remote_directory* argument specifies the name of the directory on the remote host whose contents you want to list. The *local_file* argument specifies the name of the file on the local host to which you want to write the listing. If you specify a hyphen (-) for the *local_file* argument, or if you do not specify a value for the *local_file* argument, the listing is displayed on the screen.

For the `dir`, `ls`, and `nlist` subcommands, the *remote_directory* argument is optional. If you do not specify a value for the *remote_directory* argument, FTP lists the contents of the current directory on the remote host. If you specify a value for the *local_file* argument, you must also specify a value for the *remote_directory* argument.

The `dir` and `ls` subcommands provide a detailed listing, as illustrated in the following example. These subcommands perform the same function; you can use them interchangeably.

```
ftp> dir
250 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 7442
-rw-rw-rw-  1 sue      wheel      72710 Oct 26 12:32 test.b
-rw-rw-rw-  1 sue      wheel      59052 Sep 21 16:39 test.c
-rwxrwxrwx  1 sue      comm       23917 Sep 15 12:57 test.d
-rw-rw-rw-  1 sue      wheel    7450624 Sep 20 10:12 test.e
226 Transfer complete.
272 bytes received in 0.25 seconds (01.04 Kbytes/s)
```

The `nlist` subcommand provides a listing without details, as illustrated in the following example.

```
ftp> nlist
250 PORT command successful.
150 Opening ASCII mode data connection for file list.
test.b
test.c
test.d
test.e
226 Transfer complete.
```

The `mdir` and `mls` subcommands list the contents of multiple directories on the remote host. The `mdir` subcommand provides a more detailed listing than the `mls` subcommand. To specify multiple directories, you separate each directory name with a space. The `local_file` argument is required, and it must follow the `directory_name` argument, which is also required.

If interactive-prompting mode is turned on, a prompt is displayed before the subcommand is executed. If you issued the subcommand with only one argument, you are prompted to specify the name of the local host. If you issued the subcommand with multiple arguments, you are prompted to verify that the last argument specifies the name of the local file to which you want to write the listing. If you specify `no` in response to the prompt, you must reissue the subcommand with the `local_file` argument.

Creating and Deleting Directories

The `mkdir` subcommand creates a directory on the remote host, and the `rmdir` subcommand deletes a directory on the remote host. These subcommands have the following format.

```
ftp> mkdir remote_directory

ftp> rmdir remote_directory
```

The `remote_directory` argument specifies the directory that you want to create or delete.

The following examples illustrate the use of the `mkdir` and `rmdir` subcommands to create and delete a directory, `new_dir`.

```
ftp> mkdir new_dir
257 MKD command successful.
```

```
ftp> rmdir new_dir
250 RMD command successful.
```

If the remote directory is not empty, `rmdir` will not delete the remote directory. The following example illustrates the use of the `rmdir` subcommand when the directory is not empty.

```
ftp> rmdir old_dir
550 system_a>old_dir: The directory is not empty.
```

Defining and Using Macros

An FTP macro is a sequence of subcommands that you can define and store under a specified name. You can then issue the subcommands by executing the macro.

[Table 2-12](#) lists and describes the macro subcommands, which are listed in the order in which they are presented in this section.

Table 2-12. FTP Subcommands for Macros

Subcommand	Description
<code>macdef</code>	Defines a macro
<code>\$</code>	Executes a macro

This section discusses the following topics.

- [“Defining and Executing a Macro”](#)
- [“Defining and Executing a Macro That Accepts Arguments”](#)
- [“Verifying Subcommand Status”](#)

Defining and Executing a Macro

The `macdef` subcommand defines a macro. This subcommand has the following format.

```
ftp> macdef macro_name
```

The *macro_name* argument specifies the name that you want to assign to the macro. The macro name must be from one to eight characters in length.

Once you have issued the `macdef` subcommand, a message is displayed and the cursor appears on a blank line below the message. To define the macro, enter one or more subcommands that you want to store for later execution, as illustrated in the following example. The sample macro updates the `test_results.new` file on the remote host by first deleting the oldest version of the file, `test_results.old`, renaming the existing `test_results.new` file to `test_results.old`, and then sending a new version of the `test_results` file, named `test_results.new` on the remote host.

```
ftp> macdef mac_1
Enter macro line by line, terminating it with a blank line
delete test_results.old
rename test_results.new test_results.old
send test_results test_results.new
```

To specify the end of the macro, leave a blank line after the last entry in the definition. The `ftp>` prompt then appears, and you can execute the macro by issuing the `$` subcommand.

NOTE

The `$` character is a reserved character when used in a macro definition. If you want to use the `$` character as a literal, you must precede it with a reverse slant (`\`).

The `$` subcommand executes a macro. This subcommand has the following format.

```
ftp> $ macro_name [argument...]
```

The *macro_name* argument specifies the name of the macro that you want to execute. The *argument* argument specifies one or more arguments that FTP passes to macros that have been defined to accept arguments. Arguments are passed to the macro without global file-name expansion, which means that you cannot use star names or abbreviations. (For additional information about macros with arguments, see the section “[Defining and Executing a Macro That Accepts Arguments](#)” later in this chapter.)

The following example illustrates the use of the `$` subcommand to execute the `mac_1` macro, which was defined in an example earlier in this section.

```
ftp> $ mac_1
delete test_results.old
250 DELE command successful.
rename test_results.new test_results.old
350 File exists, ready for destination name.
250 RNT0 command successful.
send test_results test_results.new
250 PORT command successful.
150 Opening ASCII mode data connection for test_results.new.
226 Transfer complete.
4173 bytes sent in 2.58 seconds (1.58 Kbytes/s)
```

When you issue the `$` subcommand, you can omit the space between the `$` character and the macro name, as illustrated in the following example.

```
ftp> $mac_1
```

A macro remains defined until you close the current connection by issuing the `close` or `disconnect` subcommand. Each macro that you define while a connection is open, including macros defined in the automatic login file, must have a unique macro name.

There are maximum limits for the number of macros you can define concurrently and for the total number of bytes for all concurrent macro definitions. You can define a maximum of 16 macros, and the total number of bytes for all macro definitions must not exceed 4096. These maximum limits include macros defined in the automatic login file and any additional macros that you define with the `macdef` subcommand. If you attempt to define a new macro once these maximum limits have been reached, an error message will be displayed when you issue the `macdef` subcommand.

To see a list of the macros that have been defined for the current connection, you can issue the `status` subcommand. The status message lists the currently defined macros at the bottom of the message, as illustrated in the following example.

```
ftp> status
Connected to system_a.admin.myfirm.com.
No proxy connection.
Mode: stream; Type: ascii; Form: non-print; Structure: file
Verbose: on; Bell: off; Prompting: on; Globbing: on
Store unique: off; Receive unique: off
Case: off; CR stripping: on
Ntrans: off
Nmap: off
Debugging: off; Hash mark printing: off;
Use of PORT cmds: on
Macros:
    mac_1
```

Defining and Executing a Macro That Accepts Arguments

You can define a macro that accepts arguments when you execute it. In the macro definition, the `$` character indicates an insertion point for an argument. When you execute the macro, you can specify one or more arguments, which FTP passes to the macro at the corresponding insertion points.

There are two types of insertion points, `$n` and `$i`. The `$n` insertion point, where *n* is an integer that represents an argument number, allows you to insert one or more arguments in the macro definition. When you issue the `$` subcommand to execute the macro, you specify an argument for each of the insertion points. As FTP executes the macro, it inserts each argument at the corresponding insertion point. For example, FTP inserts the first argument at the `$1` insertion point and the second argument at the `$2` insertion point.

The following example illustrates the definition and execution of a macro that contains two `$n` insertion points in its definition.

```
ftp> macdef mac_2
Enter macro line by line, terminating it with a null line
send $1 $2

ftp> $mac_2 test test.new
send test test.new
250 PORT command successful.
150 Opening ASCII mode data connection for test.new.
226 Transfer complete.
3074 bytes sent in 3.24 seconds (0.93 Kbytes/s)
```

The `$i` insertion point allows you to insert a single argument in the macro definition. When you issue the `$` subcommand to execute the macro, you can specify one or more arguments for this insertion point. FTP executes the macro once for each argument. Each time FTP executes the macro, it inserts the next argument specified on the `$` subcommand line.

The following example illustrates the definition and execution of a macro that contains a `$i` insertion point in its definition. In the example, the user executes the macro with three arguments, and the macro is executed once for each argument.

```
ftp> macdef mac_3
Enter macro line by line, terminating it with a null line
send test $i

ftp> $mac_3 test.x test.xx test.xxx
send test test.x
250 PORT command successful.
150 Opening ASCII mode data connection for test.x.
226 Transfer complete.
51 bytes sent in 0.08 seconds (0.66 Kbytes/s)
send test test.xx
250 PORT command successful.
150 Opening ASCII mode data connection for test.xx.
226 Transfer complete.
51 bytes sent in 0.08 seconds (0.66 Kbytes/s)
send test test.xxx
250 PORT command successful.
150 Opening ASCII mode data connection for test.xxx.
226 Transfer complete.
51 bytes sent in 0.08 seconds (0.66 Kbytes/s)
```

Verifying Subcommand Status

STCP FTP uses the VOS `command_status` process variable to indicate whether an FTP subcommand completed successfully or unsuccessfully. When FTP is running, FTP returns a three-digit reply code from the FTP remote server after the completion of an FTP subcommand. The value of the VOS `command_status` process variable is explicitly set to that three-digit code. If the FTP subcommand fails, the `command_status` process variable is set to 0. So, for example, when a `put` subcommand completes successfully, the `command_status` process variable is set to the value 260, indicating completion of the transfer. If you have VOS command macro scripts that are running FTP subcommands, you can use this feature of STCP FTP to verify the status of the FTP subcommands you are using. For more information about how to use the `command_status` process variable, see the *OpenVOS C Subroutines Manual* (R068) and the *OpenVOS Commands Reference Manual* (R098).

Getting Help

Within FTP, the help subcommands provide access to information about the FTP subcommands that the local host supports and the FTP protocol commands that the remote host supports. [Table 2-13](#) lists and describes the help subcommands. The subcommands are listed in the order in which they are presented in this section.

Table 2-13. FTP Subcommands for Getting Help

Subcommand	Description
<code>help</code> <code>?</code>	Displays information about FTP subcommands that the local host supports
<code>remotehelp</code> <code>rhel</code>	Displays information about FTP protocol commands that the remote host supports

This section discusses the following topics.

- [“Getting Information about FTP Subcommands”](#)
- [“Getting Information about FTP Protocol Commands”](#)

Getting Information about FTP Subcommands

Two subcommands, `help` and `?`, display information about FTP subcommands that the local host supports. The `help` and `?` subcommands perform the same function; you can use them interchangeably. These subcommands have the following format.

```
ftp> help [subcommand]
```

```
ftp> ? [subcommand]
```

The *subcommand* argument specifies the FTP subcommand for which you want information. If you specify a subcommand, FTP displays the syntax of the subcommand and a brief description of its function, as illustrated in the following examples.

```
ftp> help rename
rename [file_name]    rename file
```

```
ftp> ? help
help [command]    print local help information
```

If you do not specify a subcommand, FTP displays a list of all of the FTP subcommands. In the following example, the user issues the `?` subcommand to display the FTP subcommands.

```
ftp> ?
Commands may be abbreviated.  Commands are:
```

!	cr	macdef	prompt	send
..	delete	mdelete	proxy	sendport
?	debug	mdir	put	size
\$	dir	mget	pwd	sequential
append	disconnect	mkdir	quit	status
ascii	form	mls	quote	struct
bell	get	mode	recv	sunique
binary	glob	modtime	remotehelp	system
image	hash	mput	rename	tenex
bye	help	nlist	reset	type
case	lappend	nmap	rhel	user
cd	lcd	ntrans	rmdir	verbose
cdup	literal	open	rstatus	
close	ls	passive	runique	

Getting Information about FTP Protocol Commands

The `remotehelp` and `rhel` subcommands display information about the FTP protocol commands that the remote host supports. The FTP protocol commands are the actual commands that FTP sends to the remote host when you issue an FTP subcommand. The FTP protocol commands are displayed on the screen when you are working in debugging mode.

The `remotehelp` and `rhel` subcommands perform the same function; you can use them interchangeably. These subcommands have the following format.

```
ftp> remotehelp [command]
```

```
ftp> rhel [command]
```

The *command* argument specifies the FTP protocol command for which you want information. If you specify an FTP protocol command, FTP displays the syntax of the command and a brief description of its function, as illustrated in the following example.

```
ftp> rhel quit
214 Syntax: QUIT (terminate service)
```

If you do not specify a command, FTP displays a list of all of the FTP protocol commands that the remote host supports, as illustrated in the following example. The list in this example was generated by a Stratus remote host.

```
ftp> remotehelp
214- The following commands are recognized (* =>'s
unimplemented).
```

USER	PASV	APPE	MRSQ*	ABOR*	SITE*	XMKD	XCUP
PASS	TYPE	MLFL*	MRCP*	DELE	SYST	RMD	STOU
ACCT*	STRU	MAIL*	ALLO	CWD	STAT	XRMD	SIZE
REIN*	MODE	MSND*	REST*	XCWD	HELP	PWD	RSIZ
QUIT	RETR	MSOM*	RNFR	LIST	NOOP	XPWD	MDTM
PORT	STOR	MSAM*	RNTO	NLST	MKD	CDUP	

```
214
```

Issuing FTP Protocol Commands

The `literal` and `quote` subcommands send one or more verbatim strings directly to the FTP server on the remote host. These subcommands perform the same function; you can use them interchangeably. They are used primarily to send FTP protocol commands to the remote host when the local host does not have a subcommand that sends the FTP protocol command. They are also useful for sending nonstandard FTP protocol commands that the remote host supports.

The `quote` (or `literal`) subcommand has the following format.

```
ftp> quote arg_1 [arg_2...]
```

The `arg_n` arguments, where *n* represents the argument number, specify one or more verbatim strings that the subcommand sends to the remote host.

The following example illustrates the `quote` (or `literal`) subcommand. The verbatim string, `NOOP`, issues a nonstandard FTP command to a remote host that supports this command. The command returns a one-line response.

```
ftp> quote NOOP
250 NOOP command successful.
```

The `quote` (or `literal`) subcommand can return only a one-line response; therefore, it is not useful to send an FTP subcommand that returns a multiple-line response. For example, if you issue the `NLST` command, which corresponds to the `dir` subcommand, you would see only one line of a multiple-line response.

Issuing Operating-System Commands

During an FTP session, you can issue operating-system commands without ending the session. For *internal commands*, which are built into the operating system, you can issue a single command directly from the `ftp>` prompt without creating a subprocess. For *external commands*, which are defined for the operating system but are not built in, and for issuing multiple commands interactively, you suspend the current FTP session and create a subprocess.

[Table 2-14](#) lists and describes the subcommands for issuing operating-system commands during an FTP session. The subcommands are listed in the order in which they are presented in this section.

Table 2-14. FTP Subcommands for Issuing Operating-System Commands

Subcommand	Description
<code>.</code>	Issues an operating-system command directly from FTP, without creating a subprocess. Commands must be internal.
<code>!</code>	Creates a subprocess from which you can issue one or more operating-system commands. Commands can be internal or external.

This section discusses the following topics.

- [“Issuing Operating-System Commands from FTP”](#)
- [“Issuing Operating-System Commands from a Subprocess”](#)

Issuing Operating-System Commands from FTP

The `..` subcommand allows you to issue operating-system commands directly from FTP, without creating a subprocess. This subcommand is faster than the `!` subcommand, which creates a subprocess. However, only internal commands can be issued with the `..` subcommand.

The `..` subcommand has the following format.

```
ftp> .. command_line
```

The *command_line* argument specifies the command line that you want to issue. The command line can consist of multiple command strings, each separated by a semicolon (;).

NOTE _____

The space between the `..` subcommand and the *command_line* argument is **not required**.

The following example illustrates the `..` subcommand. The operating-system command `help -type internal` displays a list of the internal commands.

```
ftp> .. help -type internal
```

Issuing Operating-System Commands from a Subprocess

The `!` subcommand suspends the current FTP session and creates a subprocess. From a subprocess, you can issue one or more internal or external operating-system commands.

The `!` subcommand has the following format.

```
ftp> ! [command_line]
```

The *command_line* argument specifies the command line that you want to issue. The command line can consist of multiple command strings, each separated by a semicolon (;).

NOTE _____

The space between the `!` subcommand and the *command_line* argument is **not required**.

If you issue the `!` subcommand and specify a command line, the operating system creates a subprocess, executes the commands on the command line, and then returns to the FTP session.

If you issue the `!` subcommand without specifying a command line, the operating system suspends the FTP session and creates an interactive subprocess in which you can execute one or more operating-system commands. To leave the subprocess and return to the FTP session, you issue the `logout` command.

The following example illustrates an interactive subprocess in which the user lists the files in the current directory and submits a batch request.

```
ftp> !
TYPE 'logout' to return to FTP
ready 15:44:34
ls

Files: 4 Blocks: 3

w          1 header.txt
w          1 ioctl
w          1 ftest.c
w          1 syscall.lst

ready 15:44:37
batch 'oc ftest -table'
ready 15:44:52
logout
```

Connecting to a Stratus Remote Host

This section provides information for connecting to a Stratus remote host for an FTP session. It discusses the following topics.

- [“Logging In”](#)
- [“Determining the Initial Current Directory”](#)
- [“Appending a Local File”](#)

The information in this section applies to FTP sessions on a Stratus or non-Stratus local host.

Logging In

Before you can log in to a Stratus remote host, you must be a registered user on the remote host. On a Stratus remote host, required login data consists of a user name and password. For login purposes, your user name consists of your person name and group name. For example, a user registered with the person name `Susan_Smith` and the group name `Admin` has the user name `Susan_Smith.Admin`. You can have a registered alias for your user name. For example, a user registered with the user name

`Susan_Smith.Admin` can have the alias `sue`. When logging in, you can use your alias to specify your user name.

Stratus remote hosts do not use account passwords, and the Stratus FTP server does not support the `account` subcommand.

The following example illustrates a login procedure for a Stratus remote host. The user logs in using an alias. Because a password is required, the password prompt appears below the subcommand line.

```
ftp> user sue
Password:
230 User Susan_Smith.Admin logged in.
```

If you enter the password on the subcommand line, the characters are displayed. If you enter the password at the password prompt, the characters are not displayed.

Determining the Initial Current Directory

The initial current directory is the directory in which your user process is running when you log in. If a home directory has been defined for your user name in the `Home Dir` field of the registration database, the initial current directory is your home directory.

If a home directory has not been defined for your user name, the Stratus FTP server uses the directory `>group_name>person_name`. If the directory `>group_name>person_name` does not exist, the Stratus FTP server creates a temporary directory that serves as the initial current directory. A temporary directory is not intended for file storage. Any files written to a temporary directory are deleted when you close the connection with the remote host. If your current directory on the Stratus remote host is a temporary directory, change your current directory to a permanent directory to avoid losing files that you transfer to the remote host.

To determine whether your initial current directory is a temporary directory, issue the `pwd` subcommand to check the directory name. The name of a temporary directory has the following format.

```
pd.hexadecimal_process_id.person_name
```

For example, a temporary directory might have the name `pd.010E144A.Overseer`.

Appending a File

If you use the `append` subcommand to transfer a file to a Stratus remote host, the file to which you are appending the transferred file determines the organization of the appended file, regardless of the file-transfer type for the transfer. For example, if the file-transfer type for the transfer is `image 4096`, and the file to which you are appending a transferred file is a stream file, the appended file is stored as a stream file.

Sample FTP Sessions

The sample FTP sessions presented in this section illustrate the use of many of the subcommands. In the first session, a Stratus local host is connected to a Stratus remote host. In the second session, a Stratus local host is connected to another vendor's remote host.

In the sample sessions, the messages beginning with three-digit numbers are status and error messages that the server on the remote host generates. For information about these messages, see the section “[The FTP Server Messages](#)” in [Appendix A](#).

This section discusses the following topics.

- “[FTP Session with a Stratus Remote Host](#)”
- “[FTP Session with a Non-Stratus Remote Host](#)”

FTP Session with a Stratus Remote Host

In the sample session that follows, the user performs the following tasks.

1. begins the session without opening a connection
2. opens a connection with the remote host
3. logs in to the remote host
4. displays the name of the current directory on the local host
5. changes the current directory on the local host
6. turns on debugging mode
7. displays the name of the current directory on the remote host
8. changes the current directory on the remote host
9. sets the file-transfer type to image and the record size to 4096
10. transfers the file `sort_list.pm` to the remote host
11. sets the file-transfer type to ASCII
12. lists the contents of the current directory on the remote host to ensure that the transfer was successful
13. turns off debugging mode
14. changes the current directory on the local host
15. transfers the files that match the star name `*.cobol` to the local host
16. lists the files on the local host that match the star name `*.cobol`
17. deletes the files that match the star name `*.cobol` on the remote host
18. ends the session

NOTE

For readability, a blank line precedes each FTP subcommand in the sample sessions. In an actual session, these blank lines do not appear on the screen.

ftp

```
ftp> open system_a
Connected to system_a.admin.myfirm.com.
220 system_a FTP server (FTP 1.0 for Stratus STCP) ready.
(Compatible with OS TCP/IP)
Name (system_a:Susan_Smith.Admin): sue
331 Password required for sue.
Password:
230 User Susan_Smith.Admin logged in.

ftp> .. display_current_dir
%sl#m110>Tools>Susan_Smith

ftp> lcd programs
Local directory is now %sl#m110>Tools>Susan_Smith>programs

ftp> debug
Debugging on.

ftp> pwd
---> PWD
257 "%sysa#m1>Admin>Susan_Smith" is current directory.

ftp> cd report_tools
---> CWD report_tools
250 CWD command successful.

ftp> type image 4096
---> TYPE I 4096
200 Type set to I.
```

(Continued on next page)


```
ftp> put sort_list.pm
---> PORT 192,9,200,33,25,20
250 PORT command successful.
---> STOR sort_list.pm
150 Opening BINARY mode data connection for sort_list.pm.
226 Transfer complete.
589824 bytes sent in 40.70 seconds (14.14 Kbytes/s)
```

```
ftp> type ascii
---> TYPE A
200 Type set to A.
```

```
ftp> dir
---> PORT 192,9,200,33,28,21
226 Transfer complete.
250 PORT command successful.
---> LIST
150 Opening ASCII mode data connection for *.
```

```
Files: 4 Blocks: 609
```

w	158	stm	90-05-19 11:53:44	acctng.cobol
w	145	fix-4096	90-05-19 14:44:46	acctng.pm
w	145	stm	90-05-19 11:42:51	make_reports.cobol
w	161	fix-4096	90-05-19 11:42:43	sort_list.pm

```
Dirs: 0
```

```
Links: 0
```

```
226 Transfer complete.
101 bytes received in 0.98 seconds (00.10 Kbytes/s)
```

```
ftp> debug
Debugging off.
```

```
ftp> lcd <source
Local directory now %s1#m110>Tools>Susan_Smith>source
```

(Continued on next page)

```
ftp> mget *.cobol
mget acctng.cobol? y

File already exists:
%s1#m110>Tools>Susan_Smith>source>acctng.cobol
Delete it?? y
250 PORT command successful.
150 Opening ASCII mode data connection for acctng.cobol (4096
bytes).
226 Transfer complete.
3074 bytes received in 2.32 seconds (1.29 Kbytes/s)
mget make_reports.cobol? y

File already exists:
%s1#m110>Tools>Susan_Smith>source>make_reports.cobol
Delete it?? y
250 PORT command successful.
150 Opening ASCII mode data connection for make_reports.cobol
(3473 bytes).
226 Transfer complete.
3548 bytes received in 3.51 seconds (0.99 Kbytes/s)

ftp> .. list *.cobol

Files: 2 Blocks: 303

w      145 stm      90-05-19 11:42:51  make_reports.cobol
w      158 stm      90-05-19 11:53:44  acctng.cobol

ftp> mdelete *.cobol
mdelete acctng.cobol? y
250 DELE command successful.
mdelete make_reports.cobol? y
250 DELE command successful.

ftp> quit
221 Goodbye.
```

FTP Session with a Non-Stratus Remote Host

In the sample session that follows, the user performs the following tasks.

1. begins the session and opens a connection
2. logs in to the remote host
3. displays the name of the current directory on the local host

4. turns on debugging mode
5. displays the name of the current directory on the remote host
6. sets the file-transfer type to image
7. lists the contents of the current directory on the remote host
8. transfers the file `test.a` to the local host
9. lists the contents of the current directory on the local host
10. deletes the file `test.a` on the remote host
11. creates the directory `reports` on the remote host
12. changes the current directory on the remote host to `reports`
13. transfers the file `report.jul` to the `reports` directory
14. lists the contents of the `reports` directory
15. ends the session

NOTE

For readability, a blank line precedes each FTP subcommand in the sample session. In an actual session, these blank lines do not appear on the screen.

```
ftp sys_a
Connected to sys_a.admin.myfirm.com.
220 sys_a FTP server (SunOS 4.1)
User (sys_a.admin.myfirm.com:Susan_Smith.Admin):Susan_Smith
331 Password required for Susan_Smith.
Password:
230 User Susan_Smith logged in.
ftp> .. display_current_dir
%sl#m110>Tools>Susan_Smith

ftp> debug
Debugging on.

ftp> pwd
---> PWD
257 "/net/usr/Susan_Smith" is current directory.

ftp> type image
---> TYPE I
200 Type set to I.
```

(Continued on next page)

```
ftp> dir
---> TYPE A
---> PORT 192,9,200,33,47,12
250 PORT command successful.
---> LIST
150 Opening ASCII mode data connection for /bin/ls.
total 22135
-rw-rw-rw-  1 sue      wheel      183523 Aug 22 10:26 test.a
-rw-rw-rw-  1 sue      wheel      46528 Aug 22 10:32 test.b
-rw-rw-rw-  1 sue      wheel      50624 Aug 22 10:38 test.c
-rwxrwxrwx  1 sue      comm       58816 Aug 22 10:42 test.d
226 Transfer complete.
456 bytes received in 0.28 seconds (01.58 Kbytes/s)
---> TYPE I

ftp> get test.a
---> PORT 192,9,200,33,49,13
250 PORT command successful.
---> RETR test.a
150 Opening BINARY mode data connection for test.a (183523 bytes).
226 Transfer complete.
183523 bytes received in 4.58 seconds (39.09 Kbytes/s)

ftp> .. list test*

Files: 1 Blocks: 46

w      46 test.a

ftp> delete test.a
---> DELE test.a
250 DELE command successful.

ftp> mkdir reports
---> MKD reports
257 MKD command successful.

ftp> cd reports
---> CWD reports
250 CWD command successful.
```

(Continued on next page)

```
ftp> put report.jul
---> PORT 192,9,200,33,55,16
250 PORT command successful.
---> STOR report.jul
150 Opening BINARY mode data connection for report.jul.
226 Transfer complete.
12422 bytes sent in .77 seconds (7.53 Kbytes/s)

ftp> dir
---> TYPE A
---> PORT 192,9,200,33,57,17
250 PORT command successful.
---> LIST
150 Opening ASCII mode data connection for /bin/ls.
total 2599
-rw-rw-rw-  1 sue      wheel      12422 Aug 22 15:27 report.jul
226 Transfer complete.
49 bytes received in 0.40 seconds (00.33 Kbytes/s)
---> TYPE I

ftp> quit
---> QUIT
221 Goodbye.
```

Chapter 3

The TELNET Utility

The TELNET utility, referred to as TELNET throughout this manual, is a user interface for the Stratus implementation of the TELNET protocol. TELNET allows you to log in remotely to another computer running TELNET on a TCP/IP network. TELNET provides a set of commands, called subcommands, for establishing communications and performing TELNET functions.

TELNET consists of two programs. The *TELNET client* is a program that begins a TELNET session and processes the TELNET subcommands that you issue during the session. The *TELNET server* is a program that provides access to the login process on a computer when it is the remote host for a TELNET session.

This chapter describes TELNET and contains the following sections.

- [“Using TELNET”](#)
- [“Beginning and Ending a TELNET Session”](#)
- [“Establishing Communications”](#)
- [“Working in Input Mode”](#)
- [“Setting Operating Modes”](#)
- [“Displaying Information”](#)
- [“Connecting to a Stratus Remote Host”](#)
- [“Sample TELNET Session”](#)

The TELNET subcommands are summarized in [Appendix B](#).

NOTE

Since FTP, TELNET, and TFTP entail some security risk, you should use SSH or SFTP instead.

Using TELNET

This section provides an overview of a TELNET session, and defines terms and illustrates a client/server connection. While you are using TELNET, you are in a TELNET session. The computer on which you are using TELNET is called the local host. To begin a TELNET session, you log in to your Stratus host and issue the `telnet` command. This command executes the TELNET client program, `telnet.pm`.

During a TELNET session, you issue the TELNET subcommands described in this chapter. In response to the TELNET subcommands, the TELNET client on the local host opens a network virtual terminal connection with a TELNET server on a remote host and processes the exchange of input and output associated with the connection. The remote host's TELNET server allows you to log in to the remote host, within the access restrictions imposed by the remote host.

Figure 3-1 shows a connection between a TELNET client running on a Stratus host and a TELNET server running on a remote host.

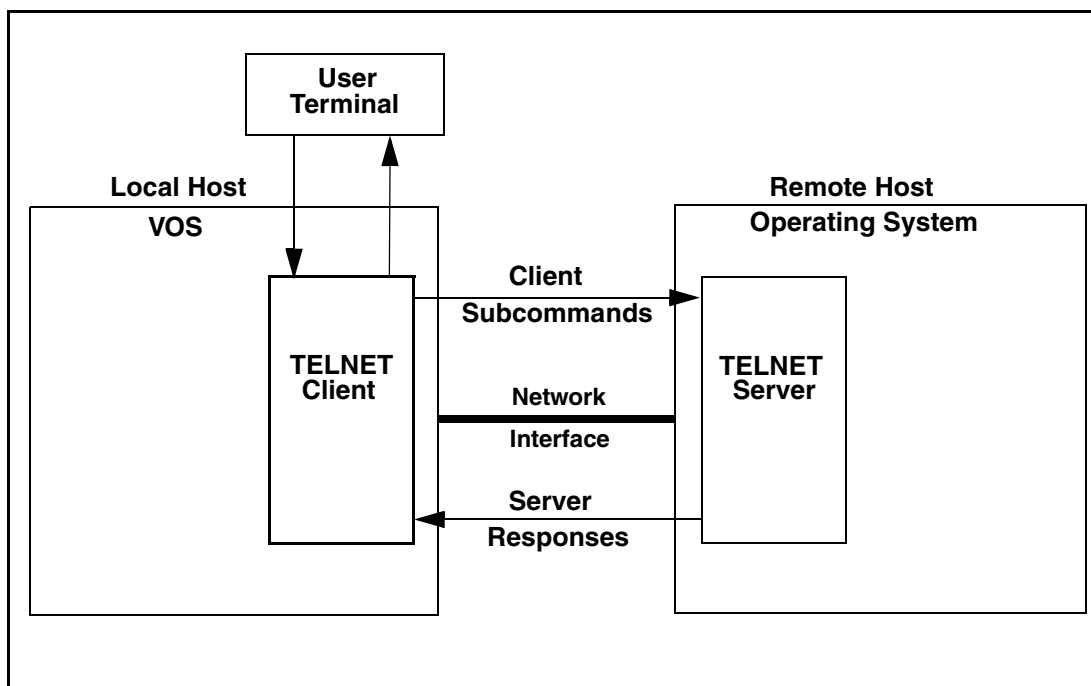


Figure 3-1. TELNET Client/Server Connection

The TELNET subcommands perform functions such as opening and closing a connection with the remote host, ending the TELNET session, performing TELNET

maintenance, and displaying online information. To issue the TELNET subcommands, you must be in *subcommand mode*. In subcommand mode, the TELNET prompt, `telnet>`, appears on the screen. While you are in subcommand mode, you can set one or more TELNET *operating modes*, which determine how the TELNET client responds to TELNET events.

During a TELNET session, you are in subcommand mode if you are not connected to a remote host. Once you open a connection to a remote host, you are in *input mode*. The text that you enter is sent to the remote host, and you function as a user of the remote host's operating system. From input mode, you can escape to subcommand mode without closing the current connection. You can also communicate with the server on the remote host by issuing a set of commands called *input requests*.

When you have finished working on the remote host, you log out, close the connection, and end the session.

[Table 3-1](#) lists and describes the TELNET subcommands. See [Appendix B](#) for a description of the `telnet` command and for an alphabetic listing and description of the TELNET subcommands.

Table 3-1. TELNET Subcommands

Subcommand	Description
<code>close</code>	Closes a connection with a remote host. If you opened the connection at <code>telnet</code> command level (<code>telnet host</code>), <code>close</code> also ends the TELNET session.
<code>display</code>	Displays the current settings for input requests and operating modes
<code>help</code> or <code>?</code>	Displays help information for subcommands
<code>open</code> [†]	Opens a connection with a remote host
<code>send</code>	Issues one or more input requests
<code>set</code>	Defines key sequences for issuing input requests directly from input mode
<code>status</code>	Displays status information for the current session
<code>toggle</code>	Turns an operating mode on or off
<code>quit</code>	Ends a TELNET session

[†] The `open` subcommand corresponds to the `telnet` command's *host* [*port_number*] arguments.

Beginning and Ending a TELNET Session

This section explains how to begin and end a TELNET session. To begin a TELNET session, you issue the `telnet` command. This command has the following format.

```
telnet host [port_number] [-address type] [-no_lookup]
```

If you issue the command with no arguments, the TELNET session is not connected to a remote host.

For additional information about the `telnet` arguments and the corresponding subcommands, see the next section, “[Establishing Communications](#),” and the description of the `telnet` command in [Appendix B](#).

If you issue the `telnet` command without specifying a remote host, you begin the session in subcommand mode. TELNET displays the TELNET prompt, `telnet>`. The following example illustrates the `telnet` command followed by the `telnet>` prompt.

```
telnet
telnet>
```

In this chapter, most of the examples that illustrate subcommands include the `telnet>` prompt.

If you do not have an open connection to a remote host, you can end a TELNET session by issuing the `quit` subcommand, as illustrated in the following example.

```
telnet> quit
```

If you have an open connection to a remote host, you can use one of the following methods to end a TELNET session.

- Escape to subcommand mode and issue the `quit` subcommand. TELNET performs the logout procedure on the remote host, closes the connection, and returns you to operating-system command level.
- Escape to subcommand mode and issue the `close` subcommand. TELNET performs the logout procedure on the remote host and closes the connection. If you opened the connection by issuing the `telnet` command with the `host` argument, TELNET ends the session and returns you to operating-system command level. Otherwise, TELNET returns you to subcommand mode, where you can issue the `quit` subcommand to end the session.
- Log out from the remote host. If you opened the connection by issuing the `telnet` command with the `host` argument, TELNET ends the session and returns you to operating-system command level. Otherwise, TELNET returns you to subcommand mode, where you can issue the `quit` subcommand to end the session.

If you issue the `quit` or `close` subcommand without logging out from the remote host, as described in the first two items of the preceding list, TELNET will perform the logout procedure. However, the recommended procedure is to log out first, as described in the last item.

Establishing Communications

This section explains how to open a connection with a remote host, and how to close a connection. This section discusses the following topics.

- [“Opening a Connection”](#)
- [“Closing a Connection”](#)

Opening a Connection

Before you can log in to a remote host, you must open a TELNET connection with the remote host. When you open a connection with a remote host, the remote host creates a TELNET server process to communicate with your TELNET client.

There are two ways to open a connection.

- As you begin a TELNET session, you can specify a remote host by issuing the `telnet` command with the *host* argument. You can also specify a port number with the *port_number* argument. If you do not specify a remote host, the session begins without an open connection. The `telnet` command issued with these arguments has the following format.

```
telnet host [port_number]
```

- During a TELNET session, you can open a connection by issuing the `open` subcommand. The `open` subcommand has the following format.

```
telnet> open host [port_number]
```

The *host* argument specifies the remote host. The *port_number* argument specifies the TELNET command port on the remote host.

The value for the *host* argument can be either a host name or an Internet address. A *host name* is a descriptive name that uniquely identifies a host on an Internet network; an *Internet address* is a numeric identifier that uniquely identifies a remote host on an Internet network.

The following example illustrates a host name.

```
system_a
```

The Internet address format is illustrated in the following example. You will see the address in this format in certain error and informational messages. The integer following the comma specifies a port number.

```
192.9.200.1,23
```

Host names are resolved to Internet addresses either locally through the `hosts` file (which has the path name `(master_disk)>system>stcp>hosts`) or externally through the Domain Name Service. The system administrator determines whether your implementation of STCP uses the `hosts` file or the Domain Name Service to resolve host names.

If the host name for a remote host cannot be resolved to an Internet address, you cannot use the host name to open a connection with that remote host. However, you can still open a connection with the remote host by specifying its Internet address.

The following example illustrates the `telnet` command issued with a host name specified for the `host` argument. The example includes the messages that appear on the screen while the connection is being opened. Note that the remote host is a non-Stratus computer that uses the UNIX operating system.

```
telnet system_a
Trying...
Connected to system_a.
Escape character is '^]'.
```

```
SunOS UNIX (system_a)
```

```
login:
```

The following example illustrates the `open` subcommand issued with an Internet address specified for the `host` argument and a port number specified for the `port_number` argument. You can specify a port number only if you specify a host.

```
telnet> open 192.9.200.1 23
```

The port number specifies the port on the remote host that receives incoming TELNET protocol commands. The default TELNET command port is 21, and most remote hosts use this port. The `services` file on the remote host specifies the port that the remote host uses for TELNET. On Stratus hosts, the path name for this file is `(master_disk)>system>stcp>services`. For information about ports, well-known ports, and the `services` file, see the *OpenVOS STREAMS TCP/IP Administrator's Guide* (R419).

If the remote host does not respond to a connection request, the TELNET client returns the following error message.

```
telnet: Unable to connect to remote host: Connection timed out
```

In response to this error message, you should check the value you have specified for the *host* argument and then try again to make the connection, using the correct name or address. If you continue to receive this error message, it is likely that the remote host you are trying to reach has been disconnected from the network.

If the TELNET server is disabled on the remote host, TELNET returns the following error message.

```
telnet: Unable to connect to remote host: Connection refused
```

If you receive an error message indicating that there may be problems with the remote host's server, you can contact the system administrator of the remote host.

NOTE

You cannot break out while TELNET is trying to make a connection; you must wait for a timeout or a failure-to-connect message. After TELNET makes a connection, you can issue the escape request (`[Ctrl]]`) to display the TELNET prompt and quit the session.

Closing a Connection

The `close` subcommand closes a connection with a remote host. If you opened the connection by issuing the `telnet` command with the *host* argument, the `close` subcommand ends the session and returns you to operating-system command level. Otherwise, TELNET returns you to subcommand mode, where you can open another connection or issue another subcommand.

The following example illustrates the `close` subcommand.

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

While a connection is open, you are in input mode, and your input to the terminal is sent to the operating system or application software on the remote host. Before you can issue the `close` subcommand, you must issue the TELNET escape characters. The default escape characters are `[Ctrl]]`.

Working in Input Mode

After you open a connection with a remote host, you are in input mode. You begin by logging in to the remote host. Once you have logged in, you must set the terminal type for your terminal on the remote host. While in input mode, you function within the remote host's operating-system environment, using the commands and conventions of that host's operating system. You can issue subcommands by escaping to subcommand mode using the escape characters. You can also communicate with the remote host's TELNET server by issuing input requests.

This section discusses the following topics.

- [“Logging In to the Remote Host”](#)
- [“Setting the Terminal Type”](#)
- [“Issuing Subcommands”](#)
- [“Issuing Input Requests”](#)
- [“Defining Key Sequences for Input Requests”](#)

Logging In to the Remote Host

After you open a connection with a remote host, you can log in to the remote host and begin working. You must have a registered account on the remote host. The remote host determines if you are required to supply a password. If the remote host requires a password and you do not supply it when you log in, a prompt for entering your password appears on the screen.

NOTE

The client does not encrypt passwords; it sends them to the server as ordinary data.

The TELNET server on the remote host determines the directory in which your process is located when you log in. The login procedure and host environment are controlled by the remote host; therefore, you should see the documentation for TELNET as it has been implemented on the remote host for additional information. If you are connected to a Stratus remote host, see the section [“Connecting to a Stratus Remote Host”](#) later in this chapter.

Setting the Terminal Type

Before working on the remote host, you must set your terminal type, using the method that is appropriate on the remote host. (On Stratus remote hosts, issue the `set_terminal_parameters` command.) The specified terminal type must be defined on the remote host. If the remote host is a Stratus computer, it is likely that there is a terminal-type definition for the terminal you are using. Otherwise, you specify

the terminal-type definition on the remote host that most closely matches the terminal-type definition for your terminal. On most systems, there is an American National Standards Institute (ANSI) terminal type that is compatible with Stratus terminals.

NOTE

Your terminal need not be a Stratus terminal as long as it matches the physical terminal or terminal emulator being used.

The Stratus implementation of the TELNET client supports the `TERMINAL_TYPE` command, which is an optional TELNET protocol command. If the TELNET server on the remote host also supports this command, the local host's client and the remote host's server negotiate to set the terminal type automatically when the connection is established.

The negotiation is invisible to the user. However, if you have turned on the `options` operating mode, you can view the messages that are passed between the client and server during the negotiation. To turn on the `options` operating mode, issue the `toggle` subcommand, as illustrated in the following example.

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

The following example illustrates the messages that appear on the screen if the remote host's server supports the `TERMINAL_TYPE` command.

```
SENT will TERMINAL_TYPE (reply)
RCVD do TERMINAL_TYPE (don't reply)
```

The following example illustrates the messages that appear on the screen if the remote host's server does not support the `TERMINAL_TYPE` command.

```
SENT will TERMINAL_TYPE (reply)
RCVD dont TERMINAL_TYPE (reply)
SENT wont TERMINAL_TYPE (reply)
```

If the remote host is not a Stratus computer, it may not recognize the terminal type specified in the `TERMINAL_TYPE` command. In this case, it is likely that the remote host will assign a default terminal type to the connection. You can ask the system administrator of the remote host to define a terminal type on the remote host that matches your terminal type.

Issuing Subcommands

From input mode, you can issue TELNET subcommands by escaping to subcommand mode. To do this, issue the escape request. The default key sequence for the escape request is `[Ctrl]]`.

At the `telnet>` prompt, you can issue any subcommand. After you issue a subcommand, you return to input mode automatically unless you issued one of the help subcommands, `?` or `help`. After the help information appears on the screen, the `telnet>` prompt appears and you can issue one additional subcommand.

To return to input mode without issuing a subcommand, press the `[Return]` key at the `telnet>` prompt.

Issuing Input Requests

While a connection is open, you can issue input requests. These requests provide functions such as editing a command line, checking the connection with the remote host, stopping the display of output, ending the current process, and escaping to subcommand mode. When you issue an input request, the client on the local host passes the corresponding TELNET protocol command to the server on the remote host. The remote host's response depends on how the remote host's server interprets the command.

You can issue any of the input requests by escaping to subcommand mode and issuing the `send` subcommand. This subcommand has the following format.

```
telnet> send input_request
```

The `input_request` argument specifies the input request that you want to issue. If the `localchars` operating mode is turned on, you can issue many of the input requests directly from input mode.

[Table 3-2](#) lists and describes the TELNET input requests and provides the corresponding argument for issuing each input request with the `send` subcommand. The table also lists the default key sequences for input requests that you can issue directly from input mode.

Table 3-2. TELNET Input Requests

Input Request	<code>send</code> <i>input_request</i> Argument	Default Key Sequence	Description
Abort-output	ao	Ctrl O	Continues to run the current process but stops sending output to the screen
Are-you-there	ayt	None	Queries the remote host to ensure that the connection is still open
Break	brk	Ctrl \	Sends the TELNET break sequence
Erase-character	ec	Ctrl H	Erases the character to the left of the cursor
Erase-line	el	Ctrl U	Erases all characters to the right of the cursor on the current command line
Escape	escape	Ctrl]	Leaves input mode and enters subcommand mode while a connection is open
Go-ahead	ga	None	Sends a go-ahead signal to the remote server; useful for operating certain terminals (for example, a physically half-duplexed terminal with a lockable keyboard)
Help	?	None	Displays a list of the input requests that you can issue with the <code>send</code> subcommand
Interrupt-process	ip	Ctrl C	Stops the current process and returns the user to operating-system command level on the remote host
No-operation	nop	None	Sends a nonexecutable command to the remote host; useful for testing successful transmission over a network
Synchronize	synch	None	Discards previously typed text that has not yet been read and processed by the remote host (requires BSD™ UNIX Version 4.3 or a subsequent version on the remote host)

You can specify multiple arguments with the `send` subcommand. The following example illustrates the `send` subcommand issued with two arguments, `intr` and `synch`. These arguments instruct the remote host to interrupt the current process and discard any previously typed text that has not yet been read and processed.

```
telnet> send ip synch
```

Defining Key Sequences for Input Requests

As shown in [Table 3-2](#), the input requests that you can issue directly from input mode have predefined default key sequences. When you issue one of these key sequences during a TELNET session, it is interpreted by the TELNET server on the remote host. The actual response to the key sequence depends on whether the remote host supports the input request and how it interprets the input request.

You can redefine the input-request key sequences by issuing the `set` subcommand. The `set` subcommand has the following format.

```
telnet> set input_request key_sequence
```

The `input_request` argument specifies the input request for which you want to define a key sequence. The `key_sequence` argument specifies the keys that you want to define for the input request. If you do not define a key sequence for an input request, TELNET uses the default key sequence.

To display the current key sequences for input requests, issue the `display` subcommand.

[Table 3-3](#) lists the input requests that you can issue directly from input mode and the corresponding `set` arguments and default key sequences.

Table 3-3. Input Requests and Corresponding `set` Arguments

Input Request	set input_request Argument	Default Key Sequence
Abort-output	flushoutput	Ctrl O
Break	quit	Ctrl \
Echo	echo	Ctrl E
End-of-file	eof	Ctrl @
Erase-character	erase	Ctrl H
Erase-line	kill	Ctrl U
Escape	escape	Ctrl]
Interrupt-process	interrupt	Ctrl C

When you define a key sequence that begins with the `Ctrl` key, you type the circumflex character (^) to represent the `Ctrl` key. The following example illustrates a `set` subcommand that defines `Ctrl B` for the break input request.

```
telnet> set quit ^B
quit character is '^B'.
```

You can remove the key sequence that has been defined for an input request. To do this, you specify `off` as the value for a `set` argument. You can still issue the input request from subcommand mode by issuing the `send` subcommand. The following example illustrates the use of the `set` subcommand to remove the key sequence for the break input request.

```
telnet> set quit off
quit character is 'off'.
```

Setting Operating Modes

TELNET operating modes determine how the TELNET client responds to TELNET events. The `toggle` subcommand turns these operating modes on and off. The `toggle` subcommand has the following format.

```
telnet> toggle operating_mode
```

The `operating_mode` argument specifies the operating mode that you want to set. To see a list of the operating modes, issue the `toggle` subcommand with the `?` argument, as illustrated in the following example.

```
telnet> toggle ?
autoflush  toggle flushing of output when sending interrupt characters
autosynch  toggle automatic sending of interrupt characters in urgent
binary     toggle sending and receiving of binary data
crlf       toggle sending carriage returns as telnet CR LF
crmod      toggle mapping of received carriage returns
localchars  toggle local recognition of certain control characters
localflow  toggle local xon/xoff flow control

debug      (debugging) toggle debugging
netdata     (debugging) toggle printing of hexadecimal network data
options     (debugging) toggle viewing of options processing

?          display help information
```

The following example illustrates the `toggle` subcommand issued with the `localchars` argument. If the `localchars` operating mode is off when you issue the

`toggle` subcommand, the subcommand turns it on; if it is on when you issue the `toggle` subcommand, the subcommand turns it off.

```
telnet> toggle localchars
Will recognize certain control characters.
telnet> toggle localchars
Won't recognize certain control characters.
```

To display the current settings for operating modes, issue the `display` subcommand.

[Table 3-4](#) lists the `toggle` arguments and their initial settings and describes their effects when the corresponding operating mode is turned on.

Table 3-4. TELNET Operating Modes *(Page 1 of 2)*

<code>toggle</code> <i>operating_mode</i> Argument	Initial Setting	Effect When Turned On
<code>autoflush</code>	On	If the <code>localchars</code> operating mode is turned on and a user issues the <code>abort-output</code> , <code>break</code> , or <code>interrupt-process</code> input request directly from input mode, <code>autoflush</code> prevents the display of data on the screen until the remote host acknowledges that it has processed the input request.
<code>autosynch</code>	Off	If the <code>localchars</code> operating mode is turned on and a user issues the <code>break</code> or <code>interrupt-process</code> input request directly from input mode, <code>autosynch</code> automatically sends the <code>synchronize</code> input request. The <code>synchronize</code> input request discards previously typed text that has not yet been read and processed by the remote host.
<code>debug</code>	Off	Enables debugging mode to be turned on or off
<code>crmod</code>	Off	Maps a carriage-return character received from the remote host to a carriage-return character followed by a line-feed character. This operating mode is useful if the remote host sends carriage-return characters without line-feed characters.
<code>localchars</code>	Off	Enables the key sequences defined for input requests so that users can issue input requests directly from input mode
<code>localflow</code>	Off	Enables local flow control to be turned on or off
<code>netdata</code>	Off	Displays network data in hexadecimal format

Table 3-4. TELNET Operating Modes (Page 2 of 2)

toggle operating_mode Argument	Initial Setting	Effect When Turned On
options	Off	Displays the TELNET protocol commands that are passed between the TELNET client and server once a connection has been opened
?	None	Displays the toggle arguments

Displaying Information

This section explains how to display help for subcommands, current settings, and TELNET status. It discusses the following topics.

- [“Displaying Online Help for Subcommands”](#)
- [“Displaying the Current Settings”](#)
- [“Displaying the Status of the TELNET Session”](#)

Displaying Online Help for Subcommands

To display online help for the TELNET subcommands, you issue one of the help subcommands, `?` or `help`. These subcommands have the following format.

```
telnet> ? [subcommand]
```

```
telnet> help [subcommand]
```

The *subcommand* argument specifies the TELNET subcommand for which you want information. If you specify a subcommand, TELNET displays the syntax of the subcommand and a brief description of its function. If you do not specify a subcommand, TELNET lists and briefly describes each of the TELNET subcommands.

The following example illustrates the `?` subcommand and the resulting output when you do not specify a subcommand. The `help` subcommand generates the same output.

```
telnet> ?
```

```
close      close current connection
display    display operating modes
open       connect to a site
quit       exit telnet
send       transmit special characters ('send ?' for more)
set        set operating modes ('set ?' for more)
status     display status information
```

```
toggle      toggle operating parameters ('toggle ?' for more)
?           display help information
```

As shown in this example, you can get additional online help for the `send`, `set`, and `toggle` subcommands by specifying `?` as an argument when you issue the subcommand.

Displaying the Current Settings

To display the current settings for an input request or an operating mode, you issue the `display` subcommand. The `display` subcommand has the following format.

```
telnet> display [setting]
```

The *setting* argument specifies the input request or operating mode for which you want to display the current setting. If you issue the `display` subcommand with an argument, TELNET displays the current setting for the specified input request or operating mode. To specify an input request, you use the corresponding *set input_request* argument; to specify an operating mode, you use the corresponding *toggle operating_mode* argument. If you issue the `display` subcommand without an argument, TELNET displays a list of **all** of the input requests and operating modes and indicates the current setting for each.

You can specify multiple arguments for a single `display` subcommand.

The following example illustrates the output that results when you issue the `display` subcommand with two arguments. The `escape` argument indicates the key sequence for escaping from input mode to subcommand mode; the `localchars` argument indicates whether you can issue certain input requests directly from input mode.

```
telnet> display escape localchars
[^^] escape
Will recognize certain control characters.
```

Displaying the Status of the TELNET Session

To display information about the status of the current TELNET session, you issue the `status` subcommand, as illustrated in the following example.

```
telnet> status
```

The output from this subcommand indicates whether a connection is open. If a connection is open, it also specifies the name or Internet address of the remote host as it was specified when the connection was opened. The output also indicates the current key sequence defined for the escape request.

The following example illustrates a `status` subcommand and the status information that TELNET returns.

```
telnet> status
Connected to system_b.
Escape character is '^['.
```

Connecting to a Stratus Remote Host

This section provides information for users who are connecting to a Stratus remote host running STCP. This information is applicable whether your local host is a Stratus computer or another vendor's computer. This section discusses the following topics.

- [“Logging In to the VOS Operating System”](#)
- [“Setting the Terminal Type”](#)
- [“Determining the Initial Current Directory”](#)
- [“The OpenVOS Command Interface”](#)
- [“Providing Connections for Applications”](#)
- [“Issuing TELNET Input Requests”](#)

When you open a connection with a Stratus remote host, the TELNET client on your local host is connected to the TELNET server on the Stratus remote host. The system administrator of the Stratus remote host installs and maintains the TELNET server and registers users. If you need to resolve issues related to the TELNET server on the remote host, contact the system administrator. For additional information about the Stratus TELNET server, see the *OpenVOS STREAMS TCP/IP Administrator's Guide* (R419).

Logging In to the VOS Operating System

Before logging in to a Stratus remote host, you must be registered as a user on the remote host. Once you open a connection, a Stratus login prompt appears on the screen. At this prompt, you enter the Stratus login command, `login`, followed by your person name or alias as registered on the Stratus remote host. After you press the Return key, the password prompt appears on the screen. At this prompt, you enter your password as registered on the remote host.

The following example illustrates a typical login procedure. When you enter your password, the characters are not echoed to the screen.

```
Please login 10:15:45
login Susan_Smith
Password:
```

Setting the Terminal Type

After you establish a connection with a Stratus remote host, you must set the terminal type for the terminal you are using. To set the terminal type, you issue the `set_terminal_parameters` command. The Stratus TELNET server does not support the optional TELNET protocol command `TERMINAL_TYPE`, which attempts to set the terminal type automatically.

To set the terminal type, you must specify a terminal type that has been defined on the Stratus remote host. On a Stratus computer, there is a terminal-type definition file (which has the path name `(master_disk)>system>sample_programs>supported_ttps>terminal_type.ttp`) for each supported terminal type. There is a corresponding directory for unsupported terminal types (which has the path name `(master_disk)>system>sample_programs>unsupported_ttps>terminal_type.ttp`). When you issue the `set_terminal_parameters` command, specify the terminal type that corresponds to the type of terminal or emulator you are using. For information about terminal-type definition files, see the manual *VOS Communications Software: Defining a Terminal Type* (R096).

Determining the Initial Current Directory

When you log in to a Stratus remote host, the OpenVOS operating system checks the registration database to determine whether a home directory has been specified for your person name. If a home directory has been specified, that directory is your initial current directory.

If a home directory has not been specified for your person name, the OpenVOS operating system uses a group name to determine your initial current directory. There are two possibilities.

- If you specify a group name when you log in (*person_name.group_name*), the server uses your home directory in the specified group.
- If you do not specify a group name when you log in, the server uses your home directory in your default group (the group specified in the first `Groups` field in the registration database).

The OpenVOS Command Interface

When you access the VOS command interface through TELNET, you access the STCP TELNET server. To determine which server you have accessed, issue the following command.

```
display_line (terminal_info device_type)
```


If you have accessed the STCP TELNET server, the command displays the following:

```
window_term
```

The STCP TELNET Server

The STCP TELNET server consists of a user-mode server and a kernel-mode access-layer driver that provides access to the Stratus window terminal software. All references to the STCP TELNET server imply the combined processing of the user-mode server and the kernel-mode access-layer driver.

The Stratus window terminal software supports character-mode terminals (for example, video display terminals) in a window environment. It also provides all of the OpenVOS command-line editing functions, such as the arrow keys, the display-form key, and recall of previous commands. For information about the OpenVOS command-line editing functions, see the *OpenVOS Commands User's Guide* (R089). For information about the command interface that the window terminal software provides, see the *Window Terminal User's Guide* (R256).

Providing Connections for Applications

For information about providing connections for applications to login terminals or slave terminals, see the *OpenVOS STREAMS TCP/IP Administrator's Guide* (R419).

Issuing TELNET Input Requests

To issue an input request, you must escape to the subcommand mode of the TELNET client on your local host. To escape to subcommand mode, issue the escape request to display the TELNET prompt. (The default key sequence for the escape request is `[Ctrl]]`.) At the TELNET prompt, issue the input-request subcommand. If your local host is a Stratus computer, the `send` subcommand issues input requests, as illustrated in the following example.

```
telnet> send input_request
```

Press the `[Return]` key to execute the input request and return to the VOS command interface. If you have accessed the window terminal software through the STCP TELNET server, the input-request subcommand remains on the terminal screen. To clear the subcommand from the terminal screen, press the redisplay function key (for example, `[Shift]-[F19]` on the V103 terminal).

[Table 3-5](#) lists and describes the TELNET input requests supported by the STCP TELNET server. For each input request, [Table 3-5](#) also lists the corresponding `send` argument and the TELNET protocol command, and describes the response.

Table 3-5. Responses to Client Requests Supported by STCP TELNET Server

Client Input Request	send <i>input_request</i> Argument	TELNET Protocol Command	Response
Abort-output	ao	AO	Server allows the current process to finish executing but stops sending data until it receives a read request
Are-you-there	ayt	AYT	Server sends a message: [Yes]
Break	brk	BRK	Client sends a break request; OpenVOS stops the current process and sends a message (which is application dependent): Request? (stop, continue, keep, login, re-enter)
Erase-character	ec	EC	Server erases the character to the left of the cursor
Erase-line	el	EL	Server erases the current command line
Escape	escape	ESCAPE	Client sends the current key sequence for the escape request
Go-ahead	ga	GA	Request is not supported; server ignores the request
Help	? or help	(None)	Client displays a list of send arguments
Interrupt-process	ip	IP	Client sends an interrupt-process request; OpenVOS stops the current process and sends a message (which is application dependent): Request? (stop, continue, keep, login, re-enter)
No-operation	nop	NOP	Server makes no response because the request is nonexecutable
Synchronize	synch	SYNCH	Server reads and discards all data until it receives the synchronizing mark created by the TELNET DATA_MARK command

Sample TELNET Session

This section provides an example of a typical TELNET client session with a non-Stratus remote host. In this sample session, a TELNET user on a Stratus local host opens a connection with, and logs in to, a remote host running the UNIX operating system.

In the sample session that follows, the user performs the following tasks.

1. begins the session without opening a connection
2. turns on the `options` and the `localchars` operating modes
3. opens a connection with the remote host
4. logs in to the remote host
5. lists the contents of the current directory by issuing the remote host's operating-system command `ls`
6. lists the current users by issuing the remote host's operating-system command `who`
7. escapes to subcommand mode by issuing the escape input request
8. queries the server on the remote host by issuing the are-you-there input request
9. displays the `board_list` file by issuing the remote host's operating-system command `cat`
10. stops the display of the `board_list` file by issuing the interrupt-process input request
11. logs out from the remote host, which closes the connection and ends the session

NOTE

In the sample session, the key sequences for input requests that do not appear on the screen are explained in the comments. (Comments are enclosed by the delimiters `/*` and `*./`.) Also, for readability, a blank line precedes each TELNET subcommand. In an actual session, these blank lines do not appear on the screen.

telnet

```
telnet> toggle options localchars
Will show option processing.
Will recognize certain control characters.
telnet> open system_a

Trying...
Connected to system_a.
Escape character is '^['.
SENT do SUPPRESS GO AHEAD.
SENT will TERMINAL TYPE (don't reply)
RCVD do TERMINAL TYPE (don't reply)
RCVD will SUPPRESS GO AHEAD (don't reply)
Received suboption Terminal type - request to send.
```

(Continued on next page)

```
RCVD will ECHO (reply)
SENT do ECHO (reply)
RCVD do ECHO (reply)
SEND wont ECHO (don't reply)
RCVD dont ECHO (don't reply)
```

4.3 BSD UNIX (system_a)

```
login: Susan_Smith
Password:          /* Password is not echoed to screen.*/
```

```
Last login: Wed Jan  1 10:17:01 from system_b
```

```
system_a> ls -la
total 36
```

```
-rw-r--r--  1 susan_smith    88 Dec  4 14:32 append_file.cm
-rw-r--r--  1 susan_smith  103 Dec 11 15:49 send_memo.cm
-rw-r--r--  1 susan_smith  541 Dec 26 12:47 board_list
-rw-r--r--  1 susan_smith  737 Dec 24 11:22 may_reports
```

```
system_a> who
smith      console Dec 31 15:20
jones      tty0     Dec 31 15:41
clark      tty1     Dec 31 15:41
clark      tty2     Dec 31 15:41
brown      tty3     Dec 31 15:41
```

```
/* User issues escape request, [Ctrl] ], to escape to subcommand
mode. */
```

```
telnet> send ayt
[Yes]
```

```
/* User presses [Return] to display the system prompt. */
```

```
system_a> cat board_list
```

```
BOARD OF DIRECTORS  -- March, 1990
-----
```

(Continued on next page)

Officers

Nancy Jones, Chairperson

Robert Baker, Treasurer

William Anderson, Secretary

(Continued on next page)

Members

/* User issues interrupt-process input request, Ctrl C. */

SENT do TIMING MARK

RCVD wont TIMING MARK (don't reply)

system_a> **logout**

Connection closed by foreign host.

ready 10:19:09

Chapter 4

The TFTP Utility

The TFTP utility, referred to as TFTP throughout this manual, is a user interface for the Stratus implementation of the STREAMS TFTP protocol for remote file transfer. TFTP allows you to communicate on a TCP/IP network with other hosts that support this protocol. TFTP is a “trivial” file transfer protocol similar to FTP, but with fewer functions.

This chapter describes TFTP and contains the following sections.

- [“Using TFTP”](#)
- [“TFTP Example”](#)

The FTP subcommands are summarized in [Appendix C](#).

Using TFTP

This section provides an overview for using TFTP. TFTP transfers a file without user authentication; that is, it does not check whether the user has a valid account or a valid password. TFTP only transfers files that are readable by anyone to a directory that is writable by anyone.

You use the `tftp` command to transfer files. The `tftp` command allows you to put a file (using the `put` command) to a directory on a remote host, or to get a file (using the `get` command) from a remote host. When using the `tftp` command, you must specify the host, the transfer mode, and the name of the file that you want to transfer. The default transfer mode is `ascii`, but you can specify the `binary` transfer mode. You can also specify a file as the target of the transfer operation. If you do not specify a target for the transfer operation, a file will be created using the same file name as the file that you want to transfer. If you do not specify a full path name, the default path for a transfer to an STCP host is `(master_disk)>system>stcp>tftp_default`.

The `tftp` command is often used to download a boot file to a diskless client host so that the boot file can then be executed. When you specify the `tftp` command at VOS command level, TFTP prompts you, in succession, for the remote host, the transfer mode, the `put` or `get` subcommand, and the name of the file you want to transfer. The following example shows the TFTP prompts with the user input in boldface.

```
tftp
host: m11
transfer_mode: binary
command: put
from_file: boot_sys.pm
```

You can specify the `trace` option if you want messages concerning the transfer displayed on the monitor screen. For further information about the `tftp` command, see [Appendix C](#).

TFTP Example

In the following example, the `tftp` command transfers the file `spec1` from the remote host `172.16.3.35` to the target file `spec1.new` on the local host.

```
tftp 172.16.3.35 ascii get %sys2#m7-1>Admin>Specs>spec1
-to_file %sys1#m1-1>specs>spec1.new
```

If the transfer succeeds, the command shows how much data was transferred. If the transfer fails, the command generates an error message in the `>stcp>logs>inetd.out` file.

Appendix A

The `ftp` Command, Subcommands, and Server Messages

This appendix describes the `ftp` command, its subcommands, and FTP server messages. It contains the following sections.

- [“`ftp`”](#)
- [“The FTP Subcommands”](#)
- [“The FTP Server Messages”](#)

ftp

Purpose

The `ftp` command begins an FTP session. FTP allows you to transfer files to and from a remote host and perform related operations.

Display Form

```
----- ftp -----
host:      █
-debug:    no
-verbose:  yes
-globbing: yes
-interactive: yes
-autologin: yes
-os_break: no
-passive:  no
```

Command-Line Form

```
ftp [host]
    [-debug]
    [-no_verbose]
    [-no_globbing]
    [-no_interactive]
    [-no_autologin]
    [-os_break]
    [-passive]
```

Arguments

► *host*

Specifies the remote host with which you want to open a connection for the FTP session. The value for the *host* argument can be either a host name or an IP address. A host name is a descriptive name associated with an Internet address. An Internet address uniquely identifies a remote host on an Internet network.

If you do not specify a host, FTP begins the session in subcommand mode without opening a connection with a remote host. In FTP subcommand mode, you can open a connection to a remote host by issuing the `open` subcommand.

► `-debug`

CYCLE

Turns on debugging mode. When debugging mode is on, each time you issue a subcommand, FTP displays the actual FTP protocol commands that it sends to the remote host. Each command is preceded by the string `-->`.

Within an FTP session, you can change the setting for this mode by issuing the `debug` subcommand.

► `-no_verbose`

CYCLE

Turns off verbose mode. When verbose mode is on, FTP displays responses from the server and file-transfer statistics.

Within an FTP session, you can change the setting for this mode by issuing the `verbose` subcommand.

► `-no_globbing`

CYCLE

Turns off global file-name-expansion mode. When global file-name-expansion mode is on, you can use star names and abbreviations. FTP expands each name and processes all files with names that match the star name or abbreviation. When global file-name-expansion mode is off, FTP treats every name as a literal.

Within an FTP session, you can change the setting for this mode by issuing the `glob` subcommand.

NOTE

If you use a star name with the subcommands `mdelete` or `mget`, the star name is expanded on the remote host according to the remote host's rules for star-name expansion. You can use abbreviations for specifying names on the local host only.

► `-no_interactive`

CYCLE

Turns off interactive-prompting mode. Interactive prompting occurs primarily during file transfer or deletion if multiple files are affected, or if a file will overwrite an existing file on the local host. If interactive-prompting mode is on, you are prompted to verify that you want to apply the current subcommand to a specified file. When interactive-prompting mode is off, file transfer or deletion occurs without prompting.

Within an FTP session, you can change the setting for this mode by issuing the `prompt` subcommand.

- ▶ `-no_autologin` CYCLE
Disables auto-login mode and prevents FTP from checking your `.netrc` file, which provides the information required to perform an automatic login to the remote host. By default, auto-login mode is enabled, which enables FTP to check your `.netrc` file to handle user ID/password verification for the remote connection. If it can find the file, it will automatically log you in.

- ▶ `-os_break` CYCLE
Turns on default break handling mode. When this mode is on, FTP always uses the default break handler, which is the operating-system break handler; when this mode is off, FTP uses its own break handlers during data transfers. The FTP break handlers allow you to abort the current data transfer without terminating the `ftp` process. For example, when you issue your terminal's break command, such as `Ctrl Break`, during a file transfer, the FTP break handlers abort the file transfer and return you to the `ftp>` prompt.

When default break mode is on, you cannot stop the current data transfer and return to the `ftp>` prompt. If you issue the break command and select the `stop` argument, the operating system stops the current `ftp` process and returns you to operating-system command level. Stratus recommends that you leave default break mode off because the default break handler may violate FTP protocol when terminating data transfers, leaving the FTP server in an unknown state on the remote host.

- ▶ `-passive` CYCLE
Turns on passive mode. In active mode FTP (the default), the client opens the command channel and the server opens the data channel. In passive mode FTP, the client opens the command and data channels.

The FTP Subcommands

This section describes the subcommands that you can issue from an FTP session. You can issue a subcommand when the `ftp>` prompt is displayed. Each of these subcommands is explained in more detail in [Chapter 2, “The FTP Utility.”](#)

The arguments for some of the subcommands specify file names. When specifying a file name, you can use a full or relative path name as defined on the system on which the file resides.

The file-transfer subcommands transfer files between the local host and a remote host. When FTP transfers a file, it sends or receives a copy of the file. The source file remains on the source system.

- ▶ `! [command_line]`
Creates a subprocess from which you can issue one or more operating-system commands. The `command_line` argument specifies the command line that you

want to issue. You can issue any type of operating-system command (internal or external) with this subcommand.

If you issue the `!` subcommand without specifying a command line, the operating system suspends the FTP session and creates an interactive subprocess in which you can execute one or more operating-system commands. To leave the subprocess and return to the FTP session, you issue the `logout` command.

If you issue the `!` subcommand and specify a command line, the operating system creates a subprocess, executes the commands on the command line, and then returns to the FTP session.

- ▶ `$ macro_name [argument...]`
Executes a macro. The *macro_name* argument specifies the name of the macro that you want to execute. The name can be one to eight characters long. The *argument* argument specifies one or more arguments that FTP passes to a macro that has been defined to accept arguments. Arguments are passed to the macro without global file-name expansion, which means that you cannot use star names or abbreviations.

(For information about defining macros, see the description of the `macrodef` subcommand later in this appendix.)

- ▶ `.. command_line`
Issues an operating-system command line, then returns to the FTP session. The *command_line* argument specifies the command line that you want to issue. Only internal commands (commands that are built into the operating system) can be issued with this subcommand.

This subcommand is faster than the `!` subcommand because it does not have to create a subprocess.

- ▶ `? [subcommand]`
Displays help information for FTP subcommands. The *subcommand* argument specifies the FTP subcommand for which you want information. If you specify a subcommand, FTP displays the syntax of the subcommand and a brief description of its function. If you do not specify a subcommand, FTP displays a list of all FTP subcommands.

This subcommand performs the same function as the `help` subcommand; you can use them interchangeably.

- ▶ `append local_file [remote_file]`
Transfers a file from the local host to the remote host and appends the file to the end of an existing file. The *local_file* argument specifies the name of the source file that you want to transfer. The *remote_file* argument specifies the name of the destination file.

If you specify a value for the *remote_file* argument, and a destination file with that name already exists, FTP appends the transferred file to the specified file. If the destination file does not exist, FTP transfers the source file to the current directory on the remote host, creating a destination file with the name specified in the *remote_file* argument.

If you do not specify a value for the *remote_file* argument, FTP looks in the current directory on the remote host for a destination file with a name that matches the *local_file* argument, and then appends the transferred file to it. If the destination file does not exist, FTP transfers the source file to the current directory on the remote host, creating a destination file with the name specified in the *local_file* argument.

► **ascii**

Sets the file-transfer type to ASCII. This is the default setting. You use this file-transfer type for transferring text files. Other file-transfer types include image, binary, and Tenex. For information about setting these file-transfer types, see the descriptions of the *image*, *binary*, and *tenex* subcommands in this appendix and in “File-Transfer Modes” in [Chapter 2](#).”

► **bell**

Changes the setting for bell mode. The default setting is off. When bell mode is on, your terminal beeps whenever a file transfer is completed.

► **binary** [*record_size*]

Sets the file-transfer type to binary. You use this file-transfer type for transferring nontext files. The *record_size* argument specifies the record size for the destination file.

This subcommand performs the same function as the *image* subcommand; you can use them interchangeably.

► **bye**

Ends the FTP session, returning your process to operating-system command level. If a connection with a remote host is open when you issue this subcommand, FTP closes the connection.

This subcommand performs the same function as the *quit* subcommand; you can use them interchangeably.

► **case**

Changes the setting for case-mapping mode. The default setting is off. When case-mapping mode is on, FTP assigns file names with all lowercase characters when transferring files to the local host with the *mget* subcommand. If a source file name contains uppercase characters, FTP changes them to the corresponding lowercase characters in the destination file name.

- ▶ `cd remote_directory`
Changes the current directory on the remote host. The *remote_directory* argument specifies the directory that you want to use. The *remote_directory* argument is required; FTP does not use the initial current directory as a default value for this argument.
- ▶ `cdup`
Changes the current directory on the remote host to the directory that is the parent of the previous current directory.
- ▶ `close`
Closes the connection with the remote host, but does not end the FTP session. This subcommand performs the same function as the `disconnect` subcommand; you can use them interchangeably.
- ▶ `cr`
The default setting is permanently on. CR-stripping mode strips out the ASCII CR characters from an ASCII file when it is received.
- ▶ `debug`
Changes the setting for debugging mode. The default setting is off. When debugging mode is on, each time you issue a subcommand, FTP displays the actual protocol commands that FTP sends to the remote host. The lines generated by debugging mode begin with the characters `--->`.
- ▶ `delete remote_file`
Deletes a file on the remote host. The *remote_file* argument specifies the file that you want to delete.
- ▶ `dir [remote_directory [local_file]]`
Lists the contents of a directory on the remote host, with details. The *remote_directory* argument specifies the name of the directory on the remote host whose contents you want to list. If you do not specify a directory, FTP lists the current directory.

The *local_file* argument specifies the name of the file on the local host to which you want to write the listing. If you specify a value for the *local_file* argument, you must also specify a value for the *remote_directory* argument. If you specify a hyphen (-) as the value for the *local_file* argument, or if you do not specify a value for the *local_file* argument, the listing is displayed on the screen.

This subcommand is similar to the `ls` and `nlist` subcommands, which provide a listing without details.

► `disconnect`

Terminates the connection with the remote host but does not end the FTP session. This subcommand performs the same function as the `close` subcommand; you can use them interchangeably.

► `form`

Displays the default file-transfer format, which is `non-print`. You cannot change this setting. The `non-print` mode specifies that the ASCII characters for vertical format control (such as formfeed and newline) are retained in the transferred file. This mode applies only when the file-transfer type is set to ASCII.

► `get remote_file [local_file]`

Transfers a file from the remote host to the local host. The `remote_file` argument specifies the name of the source file that you want to transfer. The `local_file` argument specifies the name of the destination file. If you do not specify a value for the `local_file` argument, FTP transfers the file to the current directory on the local host, using the file name specified in the `remote_file` argument.

If you specify a hyphen (-) as the value for the `local_file` argument, FTP displays the transferred file on the screen instead of writing it to a file.

If interactive-prompting mode is on and receive-unique mode is off, and the file you want to transfer has the same name as a file that already exists on the local host, FTP prompts you to verify that you want to overwrite the file on the local host. If interactive-prompting mode is off or receive-unique mode is on, FTP transfers each file without prompting. If both of these modes are off or both are on, FTP transfers each file without prompting.

This subcommand performs the same function as the `recv` subcommand; you can use them interchangeably.

► `glob`

Changes the setting for global file-name-expansion mode. The default setting is on. When global file-name-expansion mode is on, you can use star names and abbreviations. FTP expands each name and applies the current subcommand to all files with names that match the star name or abbreviation. When global file-name-expansion mode is off, FTP treats every name as a literal; it does not expand star names or abbreviations.

NOTE

If you use star names to specify file names on the remote host, the remote host expands the star names according to its rules for star-name expansion. You cannot use abbreviations to specify file names on the remote host.

- ▶ `hash`

Changes the setting for hash-mark mode. The default setting is off. When hash-mark mode is on, FTP displays a number sign (#, also called a hash mark) each time a data block (4096 bytes) is transferred.
- ▶ `help [subcommand]`

Displays information for FTP subcommands. The *subcommand* argument specifies the FTP subcommand for which you want information. If you specify a subcommand, FTP displays the syntax of the subcommand and a brief description of its function. If you do not specify a subcommand, FTP displays a list of all FTP subcommands.

This subcommand performs the same function as the `? subcommand`; you can use them interchangeably.
- ▶ `image [record_size]`

Sets the file-transfer type to image. You use this file-transfer type for transferring nontext files. The *record_size* argument specifies the record size for the destination file.

This subcommand performs the same function as the `binary` subcommand; you can use them interchangeably.
- ▶ `lappend`

The `lappend` subcommand transfers a file from the remote host to the local host and appends the file to the end of an existing local file. This command is similar to the `append` subcommand, with the following differences:

 - If the local file does not exist, `lappend` behaves in a similar manner to the `get` request.
 - If the local file does exist, `lappend` appends data from the remote file to the local file.
- ▶ `lcd [local_directory]`

Changes the current directory on the local host. The *local_directory* argument specifies the directory that you want to use. If you specify a value for the *local_directory* argument, FTP changes the current directory on the local host to the specified directory. If you do not specify a value for the *local_directory* argument, FTP changes the current directory on the local host to your home directory.
- ▶ `literal arg_1 [arg_2...]`

Sends one or more verbatim strings to the FTP server on the remote host. The *arg_n* arguments specify one or more verbatim strings that the subcommand sends to the remote host.

This subcommand performs the same function as the `quote` subcommand; you can use them interchangeably.

- ▶ `ls [remote_directory [local_file]]`
Lists the contents of a directory on the remote host, with details. The *remote_directory* argument specifies the name of the directory on the remote host whose contents you want to list. If you do not specify a directory, FTP lists the current directory. You can also specify a wildcard file name for this argument to select a group of file names that you want to list. For example, if you specify `*.cobol`, the subcommand lists all files that end with the suffix `.cobol`.

The *local_file* argument specifies the name of the file on the local host to which you want to write the listing. If you specify a value for the *local_file* argument, you must also specify a value for the *remote_directory* argument. If you specify a hyphen (-) as the value for the *local_file* argument, or if you do not specify a value for the *local_file* argument, the listing is displayed on the screen.

This subcommand performs the same function as the `nlist` subcommand; you can use them interchangeably. The `ls` and `nlist` subcommands are similar to the `dir` subcommand, which provides a more detailed listing.

- ▶ `macdef macro_name`
Defines a macro. The *macro_name* argument specifies the name that you want to assign to the macro. Once you have issued the `macdef` subcommand, a message is displayed and the cursor appears on a blank line below the message. To define the macro, you enter one or more subcommands that you want to store for later execution. To specify the end of the macro, you leave a blank line after the last entry in the definition.

NOTE _____

The `$` character is a reserved character when it is used in a macro definition. If you want to use the `$` character as a literal, you must precede it with a reverse slant (`\`).

To execute a macro, you issue the `$` subcommand, which is described earlier in this appendix.

You can define a macro that accepts argument values when you execute it. You can also store macros in the automatic login file. (For additional information about the features and limitations of FTP macros, see the sections “[Defining and Executing a Macro](#)” and “[Defining and Executing a Macro That Accepts Arguments](#)” in Chapter 2.)

► `mdelete remote_file...`

Deletes the specified files on the remote host. The *remote_file* argument specifies the names of the files that you want to delete. To specify multiple files, you separate each file name with a space. You can also use a star name to specify multiple files. If you use a star name, global file-name-expansion mode must be on. Star names are expanded according to the remote host's rules for star-name expansion.

If interactive-prompting mode is on, you are prompted to verify that you want to delete each file affected by the `mdelete` subcommand. If interactive-prompting mode is off, `mdelete` deletes each affected file without prompting.

► `mdir remote_directory... local_file`

Lists the contents of one or more directories on the remote host, with details. The *remote_directory* argument specifies the names of the directories on the remote host whose contents you want to list. To specify multiple directories, you separate each directory name with a space.

The *local_file* argument specifies the name of the file on the local host to which you want to write the listing. If you specify a hyphen (-) as the value for the *local_file* argument, the listing is displayed on the screen. The *local_file* argument is required, and it must follow the *directory_name* argument, which is also required.

If interactive-prompting mode is on, a prompt is displayed before the subcommand is executed. If you issued the subcommand with only one argument, you are prompted to specify the name of the local file. If you issued the subcommand with multiple arguments, you are prompted to verify that the last argument specifies the name of the local file to which you want to write the listing. If you specify `no` in response to the prompt, you must reissue the subcommand with the *local_file* argument.

This subcommand is similar to the `mls` subcommand, which provides a listing without details.

► `mget remote_file...`

Transfers one or more files from the remote host to the current directory on the local host. The *remote_file* argument specifies the names of the files that you want to transfer. To specify multiple files, you separate each file name with a space. You can also use a star name to specify multiple files. If you use a star name to specify multiple files, global file-name-expansion mode must be on. Star names are expanded according to the remote host's rules for star-name expansion.

If interactive-prompting mode or receive-unique mode is on, and the file you want to transfer has the same name as a file that already exists on the local host, FTP prompts you to verify that you want to delete (overwrite) the file on the local host. If both of these modes are off, FTP transfers each file without prompting.

The `mget` subcommand transfers only the files in the specified directories; it does not transfer any subdirectories.

► `mkdir remote_directory`

Creates a directory on the remote host. The `remote_directory` argument specifies the name of the directory that you want to create.

► `mls remote_directory... local_file`

Lists the contents of one or more directories on the remote host, without details. The `remote_directory` argument specifies the names of the directories on the remote host whose contents you want to list. To specify multiple directories, you separate each directory name with a space.

The `local_file` argument specifies the name of the file on the local host to which you want to write the listing. If you specify a hyphen (-) as the value for the `local_file` argument, the listing is displayed on the screen. The `local_file` argument is required, and it must follow the `directory_name` argument, which is also required.

If interactive-prompting mode is on, a prompt is displayed before the subcommand is executed. If you issued the subcommand with only one argument, you are prompted to specify the name of the local file. If you issued the subcommand with multiple arguments, you are prompted to verify that the last argument specifies the name of the local file to which you want to write the listing. If you specify `no` in response to the prompt, you must reissue the subcommand with the `local_file` argument.

This subcommand is similar to the `mdir` subcommand, which provides a more detailed listing.

► `mode`

Displays the file-transfer transmission mode, which is `stream`. You cannot change this setting. The `stream` transmission mode specifies that data is transmitted as a stream of bytes when a file is transferred.

► `modtime file_name`

Indicates the time at which a file on the remote host was last modified. The `file_name` argument specifies the file that you want to check. The time is displayed as Greenwich Mean Time.

► `mput local_file...`

Transfers one or more files from the local host to the current directory on the remote host. The `local_file` argument specifies the names of the source files that you want to transfer. To specify multiple files, you separate each file name with a space. If global file-name-expansion mode is on, you can also use a star name to specify multiple files.

If store-unique mode is on, and the file you want to transfer has the same name as a file that already exists on the remote host, FTP prompts you to verify that you want to delete and replace the file on the remote host. If store-unique mode is off, FTP transfers each file without prompting.

The `mput` subcommand transfers only the files contained in a specified directory; it does not transfer any subdirectories.

- `nlist [remote_directory [local_file]]`
Lists the contents of a directory on the remote host, without details. The *remote_directory* argument specifies the name of the directory on the remote host whose contents you want to list. If you do not specify a directory, FTP lists the current directory.

The *local_file* argument specifies the name of the file on the local host to which you want to write the listing. If you specify a value for the *local_file* argument, you must also specify a value for the *remote_directory* argument. If you specify a hyphen (-) as the value for the *local_file* argument, or if you do not specify a value for the *local_file* argument, the listing is displayed on the screen.

This subcommand performs the same function as the `ls` subcommand; you can use them interchangeably. The `nlist` and `ls` subcommands are similar to the `dir` subcommand, which provides a more detailed listing.

- `nmap [inpattern outpattern]`
Changes the setting for file-name-mapping mode. The default setting is off. The *inpattern* argument specifies the components in the source file name. The *outpattern* argument specifies the mapping of the corresponding components in the destination file name. To turn file-name-mapping mode off after using this mode, you issue the `nmap` subcommand without arguments.

When you specify a pattern, you use the character sequences \$1 through \$9 to indicate file-name components in the *inpattern* and *outpattern* arguments.

NOTE _____

The characters \$, [,], and , are reserved characters when used in `nmap` arguments. If you want to use a reserved character as a literal, you must precede the character with a reverse slant (\).

For additional information about the `nmap` subcommand, see the section [“File-Name-Mapping Mode”](#) in Chapter 2.

- ▶ `ntrans [inchars [outchars]]`

Changes the setting for character-translation mode. The default setting is off. The *inchars* argument specifies the characters in the source file name that you want to translate. The *outchars* argument specifies the corresponding translation characters in the destination file name. To turn character-translation mode off after using this mode, you issue the `ntrans` subcommand without arguments.
- ▶ `open host [port_number]`

Opens an FTP connection with a remote host. The *host* argument specifies the remote host. The value for the *host* argument can be either a host name or an Internet address. A host name is a descriptive name that uniquely identifies a host on an Internet network; an Internet address is a numeric identifier that uniquely identifies a remote host on an Internet network.

The *port_number* argument specifies the FTP command port on the remote host. You can specify a port number only if you specify a host. The default FTP command port is 21, and most remote hosts use this port.
- ▶ `passive`

Allows you to enter passive mode. In passive mode, the server waits for the client to establish a connection with it instead of attempting to connect to a client-specified port. The server responds with the IP address and the port number it is listening on. Active mode is the default; if you are in passive mode, enter the `passive` subcommand again to switch back into active mode.
- ▶ `prompt`

Changes the setting for interactive-prompting mode. The default setting is on. Interactive prompting occurs during file transfer or deletion if multiple files are affected or if a file overwrites an existing file on the local host. Before each transfer or deletion, you are prompted to verify that you want to apply the current subcommand to a specified file. When interactive-prompting mode is off, transfer or deletion occurs without prompting.
- ▶ `proxy ftp_subcommand`

Allows you to issue subcommands relevant to the proxy connection. The *ftp_subcommand* argument specifies the subcommand that you want to issue for the proxy connection. The subcommands that you can issue are a subset of the FTP subcommands. To display a list of the subcommands that you can issue for a proxy connection, issue the subcommand `proxy ?`. Some of the subcommands function differently when you issue them with the `proxy` subcommand. (For additional information about proxy connections, see [“Opening a Proxy Connection” in Chapter 2](#).)

NOTE

If you use the `proxy` subcommand, the remote host for the proxy connection must support the FTP protocol command `PASV`.

► `put local_file [remote_file]`

Transfers a file from the local host to the remote host. The `local_file` argument specifies the name of the source file that you want to transfer. The `remote_file` argument specifies the name of the destination file. If you do not specify a value for the `remote_file` argument, FTP transfers the file to the current directory on the remote host, using the file name specified in the `local_file` argument.

If the file name you specify for the destination file matches an existing file name on the remote host, the transferred file deletes and replaces the existing file unless store-unique mode is on. If store-unique mode is on, FTP changes the destination file name to make it unique.

This subcommand performs the same function as the `send` subcommand; you can use them interchangeably.

► `pwd`

Displays the name of the current directory on the remote host. (The name `pwd` stands for *print working directory*. The terms *working directory* and *current directory* are synonymous.)

► `quit`

Ends the FTP session and returns your process to operating-system command level. If a connection with a remote host is open when you issue this subcommand, FTP closes the connection.

This subcommand performs the same function as the `bye` subcommand; you can use them interchangeably.

► `quote arg_1 [arg_2...]`

Sends one or more verbatim strings to the FTP server on the remote host. The `arg_n` arguments specify one or more verbatim strings that the subcommand sends to the remote host.

► `recv remote_file [local_file]`

Transfers a file from the remote host to the local host. The `remote_file` argument specifies the name of the source file that you want to transfer. The `local_file` argument specifies the name of the destination file. If you do not specify a value for the `local_file` argument, FTP transfers the file to the current directory on the local host, using the file name specified in the `remote_file` argument.

If you specify a hyphen (-) as the value for the *local_file* argument, FTP displays the transferred file on the screen instead of writing it to a file.

If interactive-prompting mode or receive-unique mode is on, and the file you want to transfer has the same name as a file that already exists on the local host, FTP prompts you to verify that you want to delete (overwrite) the file on the local host. If both of these modes are off, FTP transfers each file without prompting.

This subcommand performs the same function as the *get* subcommand; you can use them interchangeably.

► *remotehelp* [*command*]

Displays information about the FTP protocol commands that the remote host supports. The *command* argument specifies the FTP protocol command for which you want information. If you specify an FTP protocol command, FTP displays the syntax of the command and a brief description of its function. If you do not specify a command, FTP displays a list of all FTP protocol commands that the remote host supports.

This subcommand performs the same function as the *rhel*p subcommand; you can use them interchangeably.

► *rename from to*

Renames a file on the remote host. The *from* argument specifies the current file name, and the *to* argument specifies the new file name. The *rename* subcommand changes the name specified in the *from* argument to the name specified in the *to* argument. You cannot specify a star name with this subcommand.

► *reset*

Not implemented in the STCP version of FTP.

► *rhel*p *command*

Displays information about the FTP protocol commands that the remote host supports. The *command* argument specifies the FTP protocol command for which you want information. If you specify an FTP protocol command, FTP displays the syntax of the command and a brief description of its function. If you do not specify a command, FTP displays a list of all FTP protocol commands that the remote host supports.

This subcommand performs the same function as the *remotehelp* subcommand; you can use them interchangeably.

► *rmdir remote_directory*

Deletes a directory on the remote host. The *remote_directory* argument specifies the directory that you want to delete. If the *remote_directory* is not

empty, FTP will issue a warning that it is not empty; FTP will not delete the directory unless it is empty.

► **rstatus**

Displays the status of the current connection with the remote host. The status message indicates the name of the remote host, the login name of the current user, the settings for certain modes, and the data connection if one exists.

► **runique**

Changes the setting for receive-unique mode. The default setting is off. When receive-unique mode is on, FTP ensures that each file that is transferred to the local host is assigned a unique file name. If the name of the file transferred to the local host matches the name of an existing file on the local host, FTP appends the characters `.n` to the name of the transferred file, where *n* represents an integer from 1 to 99.

► **send *local_file* [*remote_file*]**

Transfers a file from the local host to the remote host. The *local_file* argument specifies the name of the source file that you want to transfer. The *remote_file* argument specifies the name of the destination file. If you do not specify a value for the *remote_file* argument, FTP transfers the file to the current directory on the remote host, using the file name specified in the *local_file* argument.

If the file name you specify for the destination file matches an existing file name on the remote host, the transferred file overwrites the existing file unless store-unique mode is on. If store-unique mode is on, FTP changes the destination file name to make it unique.

This subcommand performs the same function as the `put` subcommand; you can use them interchangeably.

► **sendport**

Changes the setting for sendport mode. The default setting is on. When sendport mode is on, FTP sends the FTP protocol command `PORT` to the remote host when establishing a connection for a file transfer. The `PORT` command tells the server on the remote host which port to use for transferring data. If sendport mode is off, FTP does not send the `PORT` command and the default data port is used.

NOTE _____

On some remote hosts, FTP does not accept the `PORT` command. If you open a connection with a remote host that does not accept this command, you must turn off sendport mode before issuing a file-transfer subcommand.

- ▶ `sequential`
Sets the file-transfer type to sequential. The default file-transfer type is `ASCII`. Stratus provides a nonstandard file-transfer type, *sequential*, which supports transfers of sequential save files. This file-transfer type is for file transfers where the local host and the remote host are both Stratus computers.
- ▶ `size file_name`
Indicates the size, in bytes, of a file on the remote host. The *file_name* argument specifies the name of the file that you want to check.
- ▶ `status`
Displays information about the current FTP session. The status message indicates the name of the remote host attached to the primary connection and, if applicable, the name of the remote host attached to the proxy connection. Status information also includes the name of the TCP device and the current settings for all of the FTP modes.
- ▶ `struct`
Displays the default file-transfer structure, which is `file`. You cannot change this setting. The `file` mode specifies that a file is transmitted as a stream of bytes with no recognition of internal structure, such as records or pages.
- ▶ `sunique`
Changes the setting for store-unique mode. The default setting is off. When store-unique mode is on, FTP ensures that each file that is transferred to the remote host is assigned a unique file name. If the name of the file transferred to the remote host matches the name of an existing file on the remote host, the FTP server on the remote host changes the file name to make it unique and returns a message indicating the unique file name.

NOTE _____

If you turn on store-unique mode, the remote host must support the FTP protocol command `STOU`. If the remote host does not support this command, the `put`, `send`, and `mput` subcommands will not work when store-unique mode is on.

- ▶ `system`
Indicates the system type of the remote host.
- ▶ `tenex`
Sets the file-transfer type to Tenex. You use this file-transfer type for transferring files to or from a Tenex machine.

► `type [type_name] [record_size]`

Sets the file-transfer type. The *type_name* argument specifies the file-transfer type. You can specify one of the following values for this argument.

```
ascii
binary
image
sequential
tenex
```

NOTICE

The `seq` file-transfer type is nonstandard. If you attempt to use this file-transfer type when the transfer is not between Stratus computers, the results are unpredictable and may be undesirable.

The *record_size* argument specifies the record length when you specify `binary` or `image` as the file-transfer type. This argument specifies the record size for the destination file. You can specify a record size of 0 through 32767. Once you specify a record size, it is applied to all file transfers until you change it.

If you issue the `type` subcommand without specifying a file-transfer type, FTP displays the current file-transfer type. See the section “[File-Transfer Modes](#)” in [Chapter 2](#) for a description of each file-transfer type.

► `user user_name [password]`

Sends login data to the remote host. The *user_name* argument is required and specifies your user name. The *password* argument specifies your password. When logging in, specify your user name and password as registered on the remote host. If you do not specify a required password, you will be prompted to supply it.

When logging in to a Stratus remote host, you specify your full user name (consisting of your person name and your group name) as registered on the remote host. You can also specify an alias for your user name. Stratus systems require passwords.

► `verbose`

Changes the setting for verbose mode. The default setting is on. When verbose mode is on, FTP displays the responses that the remote host returns when you issue a subcommand. Responses include informational and error messages.

The FTP Server Messages

The messages that you receive from the remote host's FTP server provide you with information about how the server responds to the subcommands that you issue. These messages consist of a three-digit number followed by text. This section explains each value that can appear as the first digit on the left. This value can be an integer from 1 to 5.

NOTE _____

You receive server messages only when verbose mode is on.

The following list explains how to interpret the first digit on the left.

- 1 indicates a positive preliminary reply. This means that you will probably receive another message before you can issue another subcommand. For example, when you issue the `dir` subcommand, you receive the following message.

```
150 Opening data connection.
```

This is the first part of the server's response, which indicates that it has taken the first step in providing a list of the directory contents.

- 2 indicates a positive completion reply. This means that the server was successful in executing the subcommand, and that you can now issue another subcommand. For example, if you issue the subcommand `delete file_a`, when the server deletes the file, it sends the following message.

```
250 DELE command successful.
```

- 3 indicates a positive intermediate reply. This means that the server needs more information before it can execute the subcommand. For example, if the server requires a password, it sends the following message.

```
331 Password required for Susan_Smith.
```

- 4 indicates that the server did not accept the subcommand, but that this error condition is temporary and you can issue the subcommand again without changing it. For example, if you issued the `put` subcommand correctly, but the server could not make a data connection, the server sends the following message.

```
425 Can't build data connection.
```

Issue the `put` subcommand again.

- 5 indicates that you made an error when entering a subcommand or responding to a prompt. For example, if you make an error when entering login data, the server sends the following message.

530 Login incorrect.

Appendix B

The `telnet` Command and Subcommands

This appendix describes the `telnet` command and its subcommands. It contains the following sections.

- “[telnet](#)”
- “[The TELNET Subcommands](#)”

telnet

Purpose

The `telnet` command begins a TELNET session. TELNET allows you to log in to a remote host.

Display Form

```
----- telnet -----  
host: █  
port_number:  
-address: ipv4  
-lookup: yes
```

Command-Line Form

```
telnet host  
      [port_number]  
      [-address type]  
      [-no_lookup]
```

Arguments

► *host*

Required

Specifies the remote host with which you want to open a connection. The value for the *host* argument can be either a host name or an IP address. You can specify either IPv4 or IPv6 addresses. A host name is a descriptive name that uniquely identifies a host on an Internet network; an Internet address is a numeric identifier that uniquely identifies a remote host on an Internet network.

If you specify a host, TELNET attempts to open a connection. Once the connection is open, you are in input mode. If you do not specify a host, TELNET begins the session without opening a connection and puts you in subcommand mode. In subcommand mode, you can open a connection by issuing the `open` subcommand.

- ▶ *port_number*
Specifies the TELNET command port on the remote host. You can specify a port number only if you specify a value for the *host* argument. The default TELNET command port is 21, and most remote hosts use this port.
- ▶ *-address type* CYCLE
Specifies the type of address, IPv4 or IPv6, to use to access the host. The default is IPv4.
- ▶ *-no_lookup* CYCLE
Specifies that the specified hostname is a numeric IP address rather than a name. By default (**yes**), the command interprets the hostname argument as a name, if the name can be found; otherwise, the command interprets it as a numeric IP address.

The TELNET Subcommands

This section describes each of the subcommands that you can issue from subcommand mode during a TELNET session. You can issue a subcommand when the `telnet>` prompt is displayed. Each of these subcommands is explained in more detail in [Chapter 3, “The TELNET Utility.”](#)

- ▶ `? [subcommand]`
Displays help information for TELNET subcommands. The *subcommand* argument specifies the TELNET subcommand for which you want information. If you specify a subcommand, TELNET displays the syntax of the subcommand and a brief description of its function. If you do not specify a subcommand, TELNET lists and briefly describes all TELNET subcommands.

This subcommand performs the same function as the `help` subcommand; you can use them interchangeably.

- ▶ `close`
Closes a connection with a remote host. If you opened the connection at `telnet` command level, TELNET ends the session and returns you to operating-system command level. Otherwise, TELNET returns you to subcommand mode, where you can issue the `quit` subcommand to end the session.

- ▶ `display [setting]`
Displays the current settings for input requests and operating modes. The *setting* argument specifies the input request or operating mode for which you want to display the current setting. If you issue the `display` command with an argument, TELNET displays the current setting for the specified input request or operating mode. To specify an input request, you use the corresponding *set input_request* argument (see [Table 3-3](#)); to specify an operating mode, you use the corresponding *toggle operating_mode* argument (see [Table 3-4](#)). If you issue the `display` subcommand without an argument, TELNET displays a list of all input requests and operating modes and indicates the current setting for each.

- ▶ `help [subcommand]`
Displays information about TELNET subcommands. The *subcommand* argument specifies the TELNET subcommand for which you want information. If you specify a subcommand, TELNET displays the syntax of the subcommand and a brief description of its function. If you do not specify a subcommand, TELNET displays a list of all TELNET subcommands.

This subcommand performs the same function as the `?` subcommand; you can use them interchangeably.

- ▶ `open host [port_number]`
Opens a connection with a remote host. The *host* argument specifies the remote host. The value for the *host* argument can be either a host name or an Internet

address. A *host name* is a descriptive name that uniquely identifies a host on an Internet network; an *Internet address* is a numeric identifier that uniquely identifies a remote host on an Internet network.

The *port_number* argument specifies the TELNET command port on the remote host. You can specify a port number only if you specify a value for the *host* argument. The default TELNET command port is 23, and most remote hosts use this port.

Once the connection is open, you are in input mode.

► `quit`

Ends a TELNET session and returns you to operating-system command level. If a connection is open when you issue this subcommand, TELNET performs the logout procedure on the remote host, closes the connection, and returns you to operating-system command level.

► `send input_request`

Issues one or more input requests to the server on the remote host. The *input_request* argument specifies the input request that you want to issue. You can specify multiple input requests. If you specify `?` for the *input_request* argument, TELNET displays a list of the input requests that you can issue with the `send` subcommand.

If the `localchars` operating mode is turned on, you can issue many of the input requests directly from input mode. (For information about the `localchars` operating mode, see the description of the `toggle` subcommand later in this appendix; for information about how to define key sequences for issuing input requests directly from input mode, see the description of the `set` subcommand later in this appendix.)

For a list of the `send` arguments, their corresponding default key sequences, and a description of the functions they provide, see [Table 3-2](#).

► `set input_request key_sequence`

Defines key sequences for issuing input requests directly from input mode. The *input_request* argument specifies the input request for which you want to define a key sequence. The *key_sequence* argument specifies the keys that you want to define for the input request. If you do not define a key sequence for an input request that you can issue directly from input mode, TELNET uses the default key sequence. To display the current key sequences, issue the `display` subcommand, which is described earlier in this appendix.

For a list of the input requests that you can issue directly from input mode, the corresponding `set` arguments, and the default key sequences, see [Table 3-3](#).

You can remove the key sequence that has been defined for an input request by specifying `off` for the *key_sequence* argument. You can still issue the input request from subcommand mode by issuing the `send` subcommand.

► `status`

Displays status information for the current TELNET session. The output from this subcommand indicates whether a connection is open. If a connection is open, it also specifies the name or Internet address of the remote host as it was specified when the connection was opened. The output also indicates the current key sequence defined for the escape request.

► `toggle operating_mode`

Turns an operating mode on or off. The *operating_mode* argument specifies the operating mode that you want to set. Operating modes determine how the TELNET client responds to TELNET events. To display the current settings for operating modes, issue the `display` subcommand, which is described earlier in this appendix.

For a list of the `toggle` arguments, their initial settings, and their effect when turned on, see [Table 3-4](#).

Appendix C

The `tftp` Command

This appendix documents the `tftp` command and its arguments.

tftp

Purpose

The `tftp` command, which implements standard TFTP, enables you to transfer files to and from a local network site without supplying a password. The `tftp` command is a general-user command.

Display Form

-----tftp-----

host:

transfer_mode:

ascii

command:

put

from_file:

-to_file:

-trace:

no

Command-Line Form

```
tftp host transfer_mode command from_file
      [-to_file string]
      [-trace]
```

Arguments

- host

Required

Specifies the host name or IP address of the remote host.
- transfer_mode

CYCLE Required

Sets the transfer mode for the file. You can specify either `ascii` or `binary`. By default, `tftp` uses the `ascii` transfer mode, which is suitable for text files (ASCII files). Use the `binary` mode if you are transferring a nontext file (for example, compiled code).
- command

CYCLE Required

Specifies the operation to be performed, either `put` or `get`. Specify `put` to transfer the specified source file to the remote host; specify `get` to retrieve the specified source file from the remote host. If you do not specify a command, `tftp` uses `put`.

► *from_file***Required**

The path name of the file to be transferred. If the operation is `put`, the source file resides on the local host and will be transferred to the remote host. If the operation is `get`, the source file resides on the remote host and will be transferred to the local host.

► *-to_file string*

Specifies the name assigned to the transferred file (the copy). You should specify the full path name.

- If you do not specify a full path name, the default path for a transfer to an STCP host is `(master_disk)>system>stcp>tftp_default`.
- If you do not specify a file name for the transferred file with the `-to_file` argument, `tftp` simply uses the name of the source file.

► *-trace***CYCLE**

Enables tracing, which sends messages concerning the transfer to the console. By default, `tftp` does not enable tracing.

For related information and an example, see [Chapter 4, “The TFTP Utility.”](#)

Index

Misc.

- . . subcommand, 2-53, 2-54, A-5
- ! subcommand, 2-53, 2-54, A-5
- \$ subcommand, 2-45, 2-46, A-5, A-10
- ? subcommand
 - FTP, 2-50, 2-51, A-5, A-9, A-10
 - TELNET, 3-3, 3-10, 3-15, B-4

A

- Abbreviations, 2-3, 2-22, 2-23, 2-46, A-3, A-8
- Abort-output input request, 3-13
- Accessing
 - directories, 2-1, 2-40
 - files, 2-1, 2-31
- account
 - keyword, 2-9
 - subcommand, 2-5, 2-56
- Account passwords, 2-7, 2-8, 2-56, A-19
- Active mode
 - FTP, A-14
- Addressing
 - resolving host names to Internet addresses, 2-6, 3-6
 - using host names, 2-5, 3-5
 - using Internet addresses, 2-5, 2-6, 3-5, 3-6
- Aliases, 2-8, 2-56, 3-17
- append subcommand, 2-12, 2-32, 2-33, 2-34, 2-56, A-5
- Appending a file, 2-33, 2-34, 2-56, A-5
- Applications, providing connections for, 3-19
- Arguments in macros, 2-46, 2-48
- ASCII file-transfer type, 2-18, A-6, A-19
- ascii subcommand, 2-18, A-6
- autoflush operating mode, 3-15
- Automatic login, 2-7, 2-9, 2-10, 2-47
- Automatic login files
 - account keyword, 2-9
 - creating, 2-9
 - login keyword, 2-9

- macdef keyword, 2-9
- machine keyword, 2-9
- password keyword, 2-9
- autosynch operating mode, 3-15

B

- Beginning a session
 - FTP, 2-3, A-2
 - TELNET, 3-4, B-2
- Bell mode, 2-21, A-6
- bell subcommand, 2-21, A-6
- Binary file-transfer type, A-6, A-19
- binary subcommand, 2-18, A-6, A-9
- Block transfers, 2-30, A-9
- Break handlers, 2-3
- Break input request, 3-13
- Break mode, A-4
- bye subcommand, 2-4, A-6, A-15

C

- case subcommand, 2-25, A-6
- Case-mapping mode, 2-25, A-6
- cd subcommand, 2-41, 2-42, A-7
- cdup subcommand, 2-41, 2-42, A-7
- Changing
 - bell mode, 2-21, A-6
 - case-mapping mode, 2-25, A-6
 - character-translation mode, 2-25, A-14
 - CR-stripping mode, 2-25, A-7
 - current directory, 2-42, A-9
 - debugging mode, 2-29, A-7
 - file-name-mapping mode, 2-27, A-13
 - file-transfer type, 2-17, 2-20, A-6, A-9, A-18, A-19
 - global file-name-expansion mode, 2-23, A-8
 - hash-mark mode, 2-30, A-9
 - interactive-prompting mode, 2-22, A-14

- key sequences for input requests, 3-12, B-5
- receive-unique mode, 2-24, A-17
- sendport mode, 2-30, A-17
- store-unique mode, 2-24, A-18
- verbose mode, 2-21, A-19
- Character-translation mode, 2-25, A-14
- Client program, 1-2
 - FTP, 2-1, 2-2, 2-5
 - TELNET, 3-1, 3-2
- Client/server connections, 1-2
- close subcommand
 - FTP, 2-5, 2-12, 2-47, A-7, A-8
 - TELNET, 3-3, 3-4, 3-7, B-4
- Closing a connection
 - FTP, 2-4, 2-12, A-7, A-8
 - TELNET, 3-4, 3-5, 3-7, B-4, B-5
- Command interface, VOS, 3-18
- Command port
 - FTP, 2-6, 2-30
 - TELNET, 3-6
- Command-line editing, 3-10
- Commands
 - ftp, 2-2, A-2
 - telnet, 3-4, B-2
 - tftp, 4-2, C-1
- Concurrent open connections, 2-11
- Confirming block transfers, 2-30, A-9
- Connections
 - closing, 2-12, 3-5, 3-7, A-7, A-8, B-4, B-5
 - displaying the current status, 2-14, A-17
 - error messages, 2-6, 3-7
 - FTP, 2-2
 - opening, 2-5, 3-5, A-2, A-14, B-2, B-5
 - providing for applications, 3-19
 - proxy, 2-10, A-14
 - status, 2-14, 3-16
 - TELNET, 3-2
 - to a Stratus remote host, 2-55, 3-17
 - virtual terminal, 3-2
- Contents of a directory, 2-43
- Control key, 3-13
- cr subcommand, 2-25, A-7
- Creating
 - automatic login files, 2-9
 - directories, 2-44, A-12
 - sequential save files, 2-18
- crmod operating mode, 3-15
- CR-stripping mode, 2-25, A-7

- Current
 - directory, 2-40, 2-41, 2-42
 - macros, 2-47
 - status
 - FTP, 2-13, 2-14
 - TELNET, 3-16

D

- Data port, 2-30
- Date/time of last file modification, 2-40, A-12
- debug subcommand, 2-29, A-3, A-7
- Debugging mode, 2-3, 2-29, A-3, A-7
- Defaults
 - bell mode, 2-21
 - break mode, 2-3, A-4
 - case-mapping mode, 2-25
 - character-translation mode, 2-25
 - command port, 2-6, 2-30, 3-6
 - CR-stripping mode, 2-25
 - data port, 2-30
 - debugging mode, 2-29
 - escape request, 3-10
 - file-name-mapping mode, 2-27
 - file-transfer type, 2-17
 - ftp command, 2-3
 - global file-name-expansion mode, 2-22
 - hash-mark mode, 2-30
 - interactive-prompting mode, 2-22
 - key sequences for input requests, 3-12
 - receive-unique mode, 2-24
 - sendport mode, 2-30
 - store-unique mode, 2-24
 - telnet command, 3-4
 - terminal type, 3-9
 - verbose mode, 2-21
- Defining
 - key sequences for input requests, 3-12, B-5
 - macros, 2-45, A-10
 - in an automatic login file, 2-9
 - with arguments, 2-48
 - without arguments, 2-45
 - terminal types, 3-9
- delete subcommand, 2-32, 2-39, A-7
- Deleting
 - directories, 2-44, A-17
 - files, 2-21, 2-31, 2-39, A-7, A-11

Determining the initial current directory, 2-56, 3-18

`dir` subcommand, 2-33, 2-41, 2-43, A-7, A-10, A-13

Directories

- changing, 2-42, A-7, A-9
- contents of, 2-43
- creating, 2-44, A-12
- current, 2-40, 2-41, 2-42, A-15
- deleting, 2-44, A-17
- listing, 2-43, 2-44, A-7, A-10, A-11, A-12, A-13
- moving up one level, 2-42, A-7

Directory operations, 2-40

`disconnect` subcommand, 2-5, 2-12, 2-47, A-7, A-8

`display` subcommand, 3-3, 3-12, 3-14, 3-16, B-4, B-6

Displaying

- current directory, 2-41, A-15
- current file-transfer type, 2-17, A-19
- directories, 2-43, A-7, A-10, A-12, A-13
- escape input request, 3-16, B-6
- file size, 2-40, A-18
- files, 2-31, A-8, A-16
- FTP messages, 2-3, 2-21, A-19
- FTP protocol commands, 2-3, 2-29, A-7
- help information
 - FTP, 2-51, A-5, A-9
 - TELNET, 3-10, 3-15, B-4
- input requests, 3-12, 3-16, B-4
- last modification date/time, 2-40, A-12
- operating modes, 3-14, 3-16, B-4
- output of `get` and `recv`
 - subcommands, 2-36
- status
 - FTP, 2-13, 2-14, A-18
 - TELNET, 3-16, B-6
- TELNET output, 3-10

Domain Name Service, 2-6, 3-6

E

Editing a TELNET command line, 3-10

Encryption of passwords, 2-7, 3-8

Ending a session

- FTP, 2-3, 2-4, A-6, A-15
- TELNET, 3-4, B-5

Ensuring unique file names, 2-24

Erase-character input request, 3-13

Erase-line input request, 3-13

Error messages

- connection requests, 2-6, 3-7
- FTP server, A-20

Escape input request, 3-10, 3-13

Escaping to subcommand mode, 3-3, 3-4, 3-8, 3-10

Establishing communications, 2-4, 3-5

Ethernet, 1-1

Executing a macro, 2-45, A-5

Extended sequential files, 2-18, 2-19, 2-35

External operating-system commands, 2-53, A-5

F

`file` file-transfer structure, 2-21, A-18

File Transfer Protocol (FTP), 2-1

File transfers

- local host to remote host, 2-32, A-5, A-12, A-15, A-17
- modes, 2-17
 - `form`, 2-21
 - `mode`, 2-21
 - `struct`, 2-21
- remote host to local host, 2-36, A-8, A-11, A-15
- subcommands, 2-32, 2-36
- types, 2-17
 - ASCII, 2-20, A-6, A-19
 - between Stratus hosts, 2-20
 - binary, A-6, A-19
 - displaying the current setting, 2-17, A-19
 - fixed-length records, 2-18, A-19
 - image, 2-20, A-9, A-19
 - sequential, 2-17, 2-18, 2-20, A-19
 - Tenex, A-18, A-19

File-name-mapping mode, 2-27, A-13

Files, 2-31

- appending, 2-34, 2-56, A-5
- deleting, 2-39, A-7
- displaying the size, 2-40, A-18
- last modification date/time, 2-40, A-12
- renaming, 2-38, A-16
- size of, 2-40, A-18
- specifying path names, 2-31, A-4

- summary of subcommands, 2-31
 - transferring, 2-1, 2-20, 2-31, 2-35, A-2, A-4, A-8, A-9, A-11, A-12, A-15
- Fixed-length-record files, 2-18
- form subcommand, 2-21, A-8
- Format mode, 2-15
- FTP, 1-1, 2-1
- addressing, 2-5, A-2
 - appending a file, 2-32, A-5
 - beginning a session, 2-3, A-2
 - changing the current directory, 2-42, A-7, A-9
 - client program, 2-1, 2-2, 2-5
 - closing a connection, 2-12, A-7, A-8
 - connecting to a Stratus remote host, 2-55
 - creating
 - automatic login files, 2-9
 - directories, 2-44, A-12
 - sequential save files, 2-18
 - default break mode, A-4
 - defining a macro, 2-45, A-10
 - deleting
 - directories, 2-44, A-17
 - files, 2-39, A-7, A-11
 - determining the initial current directory, 2-56
 - displaying information
 - current directory, 2-41, A-15
 - file modification date/time, 2-40, A-12
 - file size, 2-40, A-18
 - file-transfer type, 2-17, A-19
 - help, 2-50, A-5, A-9, A-16
 - status, 2-13, 2-14, A-17, A-18
 - system type of the remote host, 2-14, A-18
 - ending a session, 2-3, A-6, A-15
 - establishing communications, 2-4
 - executing a macro, 2-45, A-5
 - file-transfer types, 2-17, A-19
 - ftp command, A-2
 - getting help, 2-50, A-5, A-9
 - illustration of a connection, 2-2
 - issuing
 - operating-system commands, 2-53, A-5
 - protocol commands, 2-52, A-9, A-15
 - listing directory contents, 2-43, A-7, A-10, A-11, A-12, A-13
 - logging in to the remote host, 2-4, 2-7, A-19
 - opening a connection
 - primary, 2-4, 2-5, A-2, A-14
 - proxy, 2-10, A-14
 - overview, 2-2
 - performing directory operations, 2-40
 - performing file operations, 2-31
 - primary connections, 2-11
 - prompt, 2-3
 - proxy connections, 2-11
 - renaming files, 2-38, 2-39, A-16
 - sample sessions, 2-57
 - sending verbatim strings to the server, 2-52, A-9, A-15
 - server messages, A-20
 - server program, 2-1, 2-2, 2-5
 - session, 2-2
 - setting modes, 2-14
 - specifying
 - command port, 2-6, A-14
 - file-transfer type, 2-17, A-6, A-9, A-18, A-19
 - passive mode, A-14
 - password, 2-7, A-19
 - remote host, 2-5, A-2
 - transferring files, 2-32, 2-36, A-8, A-11, A-12, A-15, A-17
- ftp command, 2-2, 2-3, A-2
- arguments
- debug, 2-3, 2-29, A-3
 - host, 2-3, 2-5, A-2
 - no_globbing, 2-3, 2-23, A-3
 - no_interactive, 2-3, 2-22, A-3
 - no_verbose, 2-3, 2-21, A-3
 - os_break, 2-3, A-4
 - passive, A-4
- command-line form, A-2
- corresponding FTP subcommands, 2-3
- default argument values, 2-3
- display form, A-2
- opening a connection, 2-11
- remote login procedure, 2-7
- setting modes, 2-15
- debugging, 2-29, A-3, A-4
 - default break, A-4
 - global file-name-expansion, 2-23, A-3
 - interactive-prompting, 2-22, A-3
 - verbose, 2-21, A-3

FTP modes

- bell, 2-21, A-6
- case-mapping, 2-25, A-6
- character-translation, 2-25, A-14
- CR-stripping, 2-25, A-7
- debugging, 2-3, 2-29, A-3, A-7
- default break, 2-3, A-4
- file-name-mapping, 2-27, A-13
- file-transfer, 2-17
 - format, 2-15, 2-21, A-8
 - structure, 2-15, 2-21, A-18
 - transmission, 2-15, 2-21, A-12
 - type, 2-17, A-19
- global file-name-expansion, 2-3, 2-22, 2-34, 2-37, 2-38, A-3, A-8
- hash-mark, 2-30, A-9
- interactive-prompting, 2-3, 2-21, 2-35, 2-44, A-3, A-14
- receive-unique, 2-24, 2-38, A-17
- sendport, 2-30, A-17
- store-unique, 2-24, A-18
- verbose, 2-3, 2-21, A-3, A-19

FTP protocol commands

- getting help, 2-52, A-16
- PASV, 2-12, A-15
- PORT, 2-29, 2-30, A-17
- STOR, 2-29
- STOU, 2-25, A-18

FTP subcommands, A-4

- ., 2-53, 2-54, A-5
- !, 2-53, 2-54, A-5
- \$, 2-45, 2-46, A-5
- ?, 2-50, 2-51, A-5
- account, 2-5
- append, 2-12, 2-32, A-5
- ascii, 2-18, A-6
- bell, 2-21, A-6
- binary, 2-18, A-6
- bye, 2-4, A-6
- case, 2-25, A-6
- cd, 2-41, 2-42, A-7
- cdup, 2-41, 2-42, A-7
- close, 2-5, 2-12, A-7
- cr, 2-25, A-7
- debug, 2-29, A-7
- delete, 2-32, 2-39, A-7
- dir, 2-41, 2-43, A-7
- disconnect, 2-5, 2-12, A-8
- for changing the current directory, 2-42

- for defining and using macros, 2-45
- for directory operations, 2-40
- for establishing communications, 2-4
- for file operations, 2-31
- for file-transfer types, 2-17, A-4
- for getting help, 2-50
- for issuing operating-system commands, 2-53
- for proxy connections, 2-11
- form, 2-15, A-8
- get, 2-12, 2-32, 2-36, A-8
- glob, 2-23, A-8
- hash, 2-30, A-9
- help, 2-50, 2-51, A-9
- image, 2-18, A-9
- lappend, 2-35, A-9
- lcd, 2-41, 2-42, A-9
- ls, 2-41, 2-43, A-10
- macdef, 2-45, A-10
- mdelete, 2-32, 2-39, A-11
- mdir, 2-41, 2-43, A-11
- mget, 2-32, 2-36, A-11
- mkdir, 2-41, 2-44, A-12
- mls, 2-41, 2-43, A-12
- mode, 2-15, A-12
- modtime, 2-32, 2-40, A-12
- mput, 2-12, 2-32, A-12
- nlist, 2-41, 2-43, A-13
- nmap, 2-27, A-13
- ntrans, 2-25, A-14
- open, 2-5, 2-12, A-14
- passive, 2-30
- prompt, 2-22, A-14
- proxy, 2-5, 2-10, A-14
- put, 2-12, 2-32, A-15
- pwd, 2-41, A-15
- quit, 2-4, A-15
- quote, 2-52, A-9, A-15
- recv, 2-12, 2-32, 2-36, A-15
- remotehelp, 2-50, 2-52, A-16
- rename, 2-32, 2-38, A-16
- reset, 2-5
- rhelph, 2-50, 2-52, A-16
- rmdir, 2-41, 2-44, A-17
- rstatus, 2-13, 2-14, A-17
- runique, 2-24, A-17
- send, 2-12, 2-32, A-17
- sendport, 2-30, A-17
- size, 2-32, 2-40, A-18

- status, 2-13, A-18
- struct, 2-15, A-18
- sunique, 2-24, A-18
- system, 2-13, 2-14, A-18
- tenex, 2-18, A-18
- type, 2-18, A-19
- user, 2-5, 2-7, 2-8, A-19
- verbose, 2-21, A-19
- ftp> prompt, 2-3, A-4
- ftp.pm program, 2-2

G

- get subcommand, 2-12, 2-32, 2-36, A-8, A-16
- Getting help
 - FTP, 2-50, 2-51, 2-52, A-5, A-9
 - TELNET, 3-15, B-4
- glob subcommand, 2-23, A-3, A-8
- Global file-name-expansion mode, 2-3, 2-22, 2-34, 2-37, 2-38, 2-39, 2-46, A-3, A-8
- Globbering, 2-22
- Group names, 2-56, 3-18

H

- hash subcommand, 2-30, A-9
- Hash-mark mode, 2-30, A-9
- Help
 - FTP, 2-50
 - protocol commands, 2-52, A-16
 - subcommands, 2-51, A-5, A-9
 - TELNET, 3-15
 - input requests, 3-16, B-4
 - operating modes, 3-16, B-4
 - subcommands, 3-15, B-4
- help subcommand
 - FTP, 2-50, 2-51, A-5, A-9
 - TELNET, 3-3, 3-10, 3-15, B-4
- Home directories, 2-42, 2-56, 3-18, A-9
- host argument
 - ftp command, 2-5, A-2
 - telnet command, 3-4, 3-5, B-2
- Host names, 3-6
- Hosts, 1-2
 - FTP, 2-5
 - TELNET, 3-5
- hosts file, 2-6, 3-6

I

- Image file-transfer type, 2-18, A-9, A-19
- image subcommand, 2-18, A-6, A-9
- Indexed files, 2-18
- Initial current directory
 - FTP, 2-8, 2-56
 - TELNET, 3-8, 3-18
- Initial settings
 - input requests, 3-13
 - operating modes, 3-14
- Input mode, 3-3, 3-8, 3-10, 3-12, B-2, B-5
- Input requests, 3-3, 3-10, 3-16, B-5
 - default key sequences, 3-12
 - defining key sequences, 3-12, B-5
 - displaying current settings, 3-16, B-4
 - escape, 3-10
 - issuing from input mode, 3-10
 - issuing with the send subcommand, 3-10, B-5
 - no-operation, 3-11
 - responses of Stratus remote hosts, 3-19
 - synchronize, 3-11
 - turning off key sequences, 3-13, B-6
- Interactive-prompting mode, 2-3, 2-21, 2-22, 2-35, 2-39, 2-44, A-3, A-8, A-11, A-14, A-16
- Internal operating-system commands, 2-53, 2-54, A-5
- Internet addresses, 2-5, 2-6, 3-5, 3-6
- Interrupt-process input request, 3-13
- Issuing
 - FTP
 - nonstandard commands, 2-52, A-9, A-15
 - operating-system commands, 2-53, 2-54, A-5
 - protocol commands, 2-52, A-9, A-15
 - subcommands, 2-2
 - TELNET
 - input requests, 3-8, 3-10, 3-12, 3-19, B-5
 - subcommands, 3-3, 3-8, 3-10

K

- Key sequences for input requests, 3-12, 3-16, B-5

Keywords for .netrc file, 2-9

- account, 2-9
- login, 2-9
- macdef, 2-9
- machine, 2-9
- password, 2-9

L

- lappend subcommand, 2-35, A-9
- lcd subcommand, 2-41, 2-42, A-9
- Limitations for macros, 2-47
- Listing directory contents, 2-43
 - for multiple directories, A-11, A-12
 - with details, A-7
 - without details, A-10, A-13
- Local host, 1-2
 - FTP, 2-2
 - TELNET, 3-2
- localchars operating mode, 3-10, 3-15, B-5
- Logging in
 - FTP, 2-7, 2-8
 - Stratus remote host, 2-55, 3-17
 - TELNET, 3-1, 3-8, 3-17, B-2
- Logging out from a TELNET remote host, 3-4
- login
 - command, 3-17
 - keyword, 2-9
- ls subcommand, 2-41, 2-43, A-7, A-10, A-13

M

- macdef
 - keyword, 2-9
 - subcommand, 2-45, A-5, A-10
- machine keyword, 2-9
- Macros, 2-45
 - current, 2-47
 - defining, 2-45, A-10
 - with arguments, 2-48
 - with multiple executions, 2-49
 - executing, 2-45, A-5
 - in an automatic login file, 2-9, 2-47
 - limits, 2-47
 - naming, 2-46, 2-47
 - reserved characters, 2-46, A-10
 - star names and abbreviations, 2-46
- Mapping source file names to destination file names, 2-27, A-13

- mdelete subcommand, 2-22, 2-23, 2-32, 2-39, A-3, A-11
- mmdir subcommand, 2-41, 2-43, 2-44, A-11, A-12
- mget subcommand, 2-23, 2-25, 2-32, 2-36, 2-37, A-3, A-11
- mkdir subcommand, 2-41, 2-44, A-12
- mls subcommand, 2-41, 2-43, 2-44, A-11, A-12
- mode subcommand, 2-21, A-12
- Modes
 - FTP, 2-14
 - TELNET, 3-3, 3-8
- Modification-time information, 2-40, A-12
- modtime subcommand, 2-32, 2-40, A-12
- Moving up one directory level, 2-42, A-7
- mput subcommand, 2-12, 2-25, 2-32, 2-34, A-12, A-18
- Multiple file deletions, 2-22, 2-39, A-11
- Multiple file transfers, 2-22, 2-32, 2-36, A-11, A-12

N

- Naming a macro, 2-46, 2-47
- Negotiating the terminal type, 3-9
- netdata operating mode, 3-15
- .netrc file, 2-9
 - example, 2-10
 - keywords, 2-9
- Network interface, 1-1, 1-2
 - definition, 1-2
 - FTP, 2-2
 - TELNET, 3-2, 3-4
- nlist subcommand, 2-41, 2-43, 2-44, A-7, A-10, A-13
- nmap subcommand, 2-27, A-13
- non-print file-transfer format, 2-21, A-8
- Nonstandard FTP commands, 2-52, A-9, A-15
- Nontext files, 2-18
- No-operation input request, 3-11
- nop input request, 3-11
- ntrans subcommand, 2-25, A-14

O

- Online help
 - FTP, 2-50, A-5, A-9
 - TELNET, 3-15, B-4

open subcommand

FTP, 2-5, 2-6, 2-7, 2-11, 2-12, A-3, A-14
TELNET, 3-3, 3-5, 3-6, B-5

Opening a connection

FTP, 2-5, A-2, A-14
TELNET, 3-5, B-2, B-5

Opening a proxy connection, 2-10, A-14**Operating modes, 3-3, 3-13, 3-14, 3-16, B-6**

autoflush, 3-15
autosynch, 3-15
crmod, 3-15
current settings, 3-16, B-4
initial settings, 3-14
localchars, 3-15
netdata, 3-15
options, 3-9, 3-15
toggle arguments, 3-15

Operating-system commands, 2-53

.. subcommand, 2-54, A-5
! subcommand, 2-54, A-5
issuing from a subprocess, 2-54, A-5
issuing from FTP, 2-54, A-5

options operating mode, 3-9, 3-15**OS TCP/IP**

previously released product, 1-3

Overwriting existing files, 2-21, A-17, A-18**P****Passive mode**

FTP, A-14

passive subcommand, 2-30**Password encryption, 2-7, 3-8****password keyword, 2-9****Passwords**

FTP, 2-7, 2-8, 2-56, A-19
TELNET, 3-8, 3-17

PASV FTP protocol command, 2-12, A-15**Path names, 2-31, A-4****Person names, 2-56, 3-17****PORT FTP protocol command, 2-29, 2-30, A-17****port_number argument, 3-4, 3-5, B-3****Ports**

FTP, 2-6, 2-30, A-14
TELNET, 3-5, 3-6, B-3, B-5

Primary connections, 2-11**Printing the working directory, 2-41, A-15****prompt subcommand, 2-22, A-3, A-14****Prompts**

ftp>, 2-3, 2-7
login, 2-3, 2-7, 3-8, 3-17
telnet>, 3-3, 3-10
verification, 2-3, 2-21, 2-35

Protocol commands

FTP, 2-3, 2-52, A-3, A-7, A-16
TELNET, 3-6, 3-10, 3-15, 3-19

Proxy connections, 2-10, A-14

definition of, 2-11
opening, 2-10
subcommands, 2-11, 2-12
support requirements, 2-12

proxy subcommand, 2-5, 2-10, A-14**put subcommand, 2-12, 2-23, 2-25, 2-32, 2-33, A-15, A-17, A-18****pwd subcommand, 2-41, 2-56, A-15****Q****quit subcommand**

FTP, 2-4, A-6, A-15
TELNET, 3-3, 3-4, 3-5, B-5

quote subcommand, 2-52, A-9, A-15**R****Receive-unique mode, 2-24, 2-36, 2-38, A-8, A-11, A-16, A-17****Receiving files from the remote host, 2-36**

displaying files, 2-36
ensuring unique file names, 2-36
specifying file names, 2-36
verifying, 2-37

recv subcommand, 2-12, 2-32, 2-36, A-8, A-15**Registering on the remote host, 2-7, 3-8****Relative files, 2-18****Remote file transfer, 1-1, 2-1, A-2****Remote hosts, 1-2**

account passwords, 2-7
addressing, 2-5, 3-5, A-2, B-2
assigning unique file names, 2-24, A-17, A-18
automatic login, 2-10
changing the current directory, 2-42, A-7
closing a connection, 2-4, 2-12, 3-4, A-7, A-8, B-4, B-5
creating a directory, 2-44, A-12
defining terminal types, 3-9

- deleting a directory, 2-44, A-17
- deleting a file, 2-39, A-7
- determining supported FTP protocol
 - commands, 2-52, A-16
- directory operations, 2-40
- disconnected from the network, 2-6
- displaying
 - current system type, 2-14, A-18
 - responses, 2-21, A-19
 - status, 2-14, A-17
- file operations, 2-31
- FTP, 2-1, 2-2
- initial directory, 2-8, 3-8
- listing directory contents, 2-43, A-7, A-11, A-12, A-13
- logging in, 2-7, 2-8, 3-8
- logging out, 3-4
- non-Stratus, 2-60
- opening a connection, 2-2, 2-5, 3-2, 3-5, A-3, A-14, B-2, B-5
- passwords, 2-7, 3-8
- proxy connections, 2-11, A-14
- receiving files from, 2-36, A-8, A-11, A-12, A-15
- registration on, 2-7, 2-56, 3-8, 3-17, A-19
- renaming a file, 2-38, A-16
- sending files to, 2-32, A-5, A-15, A-17
- servers disabled on, 2-7
- setting the terminal type, 3-9, 3-18
- specifying, 2-2, 2-5, 3-4, 3-5, A-2, A-14, B-2, B-5
- Stratus, 2-55, 2-57, 3-17
- TELNET, 3-1, 3-2, 3-5, 3-8
- user names, 2-7, 3-8
- Remote login, 1-1
- remotehelp subcommand, 2-12, 2-50, 2-52, A-16
- Removing key sequences for input
 - requests, 3-13
- rename subcommand, 2-32, 2-38, A-16
- Renaming files, 2-31, 2-38, A-16
- Representation type, 2-17
- Representing the control key, 3-13
- Reserved characters
 - in macro definitions, 2-46, A-10
 - in nmap arguments, 2-27, A-13
- reset subcommand, 2-5
- Resolving host names, 2-6, 3-6
- Restoring sequential save files, 2-19

- rhelph subcommand, 2-50, 2-52, A-16
- rmdir subcommand, 2-41, 2-44, A-17
- rstatus subcommand, 2-13, 2-14, A-17
- runique subcommand, 2-24, A-17

S

Sample sessions

FTP

- with a non-Stratus remote host, 2-60

- with a Stratus remote host, 2-57

TELNET, 3-20

- save operating-system command, 2-18

- Security, 2-7, 3-8

- Security risks, 3-1

- Selecting a file-transfer type, 2-17

- send subcommand

- FTP, 2-12, 2-25, 2-32, 2-33, A-15, A-17, A-18

- TELNET, 3-3, 3-10, B-5

Sending

- files to the remote host, 2-32, A-12, A-15, A-17

- input requests, 3-10, B-5

- verbatim strings to the FTP server, 2-52, A-9, A-15

- Sendport mode, 2-30, A-17

- sendport subcommand, 2-30, A-17

- Sequential files, 2-18

- Sequential file-transfer type, 2-17, 2-18, 2-19, A-19

- Sequential save files, 2-17, 2-18, 2-19

- Server program, 1-2

- FTP, 2-1, 2-2, 2-5

- TELNET, 3-1

- services file, 2-6, 3-6

Session

- FTP, 2-2, 2-3

- TELNET, 3-2, 3-4

- set subcommand, 3-3, 3-12, 3-13, B-5

Setting

modes

- FTP, 2-14

- TELNET, 3-12, 3-13, B-6

- terminal types, 3-8, 3-9, 3-18

Settings

- input requests, 3-12, 3-16, B-4, B-5

- operating modes, 3-13, 3-14, 3-16, B-4, B-6

Size of a file, 2-40, A-18

`size` subcommand, 2-32, 2-40, A-18

Specifying

port numbers, 2-6, 3-4, 3-5, A-14, B-3, B-5

remote hosts, 2-5, 3-4, 3-5, A-2, A-14, B-2

terminal types, 3-8

Star names, 2-3, 2-22, 2-23, 2-34, 2-37, 2-38,
2-39, 2-46, A-3, A-8, A-16

Status

FTP, 2-13

connections, 2-14, A-17

sessions, 2-13, A-18

TELNET, 3-15

connections, 3-16, B-6

input requests, 3-16, B-4

operating modes, 3-16, B-4

sessions, 3-16, B-4, B-6

`status` subcommand

FTP, 2-13, A-18

TELNET, 3-3, 3-16, B-6

STCP TELNET server, 3-19

Stopping

current TELNET process, 3-10

display of TELNET output, 3-10

`STOR` FTP protocol command, 2-29

Store-unique mode, 2-24, 2-33, A-13, A-15,
A-17, A-18

`STOU` FTP protocol command, 2-25, A-18

Stratus remote hosts

account passwords, 2-56

aliases, 2-56, 3-17

appending a file, 2-34, 2-56

group names, 3-18

home directories, 2-56, 3-18

initial current directory, 2-56, 3-18

issuing input requests, 3-19

logging in, 2-55, 3-17

`login` command, 3-17

passwords, 2-56, 3-17

person names, 2-56, 3-17

responses to TELNET protocol

commands, 3-19

servers, 2-55, 3-17

setting the terminal type, 3-18

temporary directories, 2-56

`TERMINAL_TYPE` TELNET protocol

command, 3-18

terminal-type definition files, 3-18

user registration, 2-56, 3-17

Stratus TCP/IP products, 1-3

Stream files, 2-18

`stream` file-transfer transmission mode, 2-21,
A-12

STREAMS TCP/IP (STCP), 1-1

Stripping return characters from transferred
files, 2-25, A-7

`struct` subcommand, 2-21, A-18

Structure mode, 2-15

Subcommand mode, 3-3, 3-4, 3-7, 3-8, 3-10,
B-2, B-4

Subcommands

FTP, 2-52, A-4

TELNET, 3-1, 3-10, 3-16, B-4

Subprocesses, 2-54

`sunique` subcommand, 2-24, A-18

`synch` input request, 3-11

Synchronization, 3-11

Synchronize input request, 3-11

`system` subcommand, 2-13, 2-14, A-18

System type of the remote host, 2-14, A-18

T

TCP/IP. See Transmission Control
Protocol/Internet Protocol (TCP/IP)

TELNET, 1-1, 3-1

addressing, 3-5, B-2

beginning a session, 3-4, B-2

client program, 3-1

closing a connection, 3-4, 3-7, B-4, B-5

connecting to a Stratus remote host, 3-17

defining key sequences for input
requests, 3-12, B-5

displaying information

current settings, 3-16, B-4

help, 3-15, B-4

status, 3-16, B-6

editing a command line, 3-10

ending a session, 3-4, B-5

establishing communications, 3-5

illustration of a connection, 3-2

input mode, 3-3, 3-8, 3-10, B-2

input requests, 3-3, 3-10, 3-11

issuing input requests, 3-10, B-5

issuing subcommands, 3-10

list of subcommands, 3-3

logging in to the remote host, 3-8

- logging in to the VOS operating system, 3-17
- logging out from the remote host, 3-4
- modes
 - input, 3-3, 3-10, B-2, B-5
 - operating, 3-3, 3-13, B-6
 - subcommand, 3-3, 3-10, B-2
- opening a connection, 3-5, B-2, B-5
- operating modes, 3-3, 3-13, B-6
- overview, 3-2
- prompt, 3-3, 3-4
- registering on the remote host, 3-17
- sample sessions, 3-20
- server program, 3-1
- servers
 - STCP TELNET server, 3-19
- session, 3-2
- setting
 - operating modes, 3-13, B-6
 - the terminal type, 3-8, 3-18
- specifying
 - a command port, 3-4, 3-5, B-3, B-5
 - a remote host, 3-4, 3-5, B-2
 - subcommand mode, 3-3, 3-4, 3-10, B-2
 - telnet command, 3-4, B-2
- telnet command, 3-2, 3-3, 3-4, B-1, B-2
 - arguments
 - host*, 3-3, 3-5, B-2
 - port_number*, 3-5, B-3
 - command-line form, B-2
 - default argument values, 3-4
 - display form, B-2
 - opening a connection, 3-5, B-2
 - specifying
 - port numbers, 3-5, B-3
 - remote hosts, 3-5, B-2
- TELNET subcommands, 3-3, B-4
 - ?, 3-3, 3-15, B-4
 - close, 3-3, 3-4, 3-7, B-4
 - display, 3-3, 3-12, 3-14, 3-16, B-4
 - help, 3-3, 3-15, B-4
 - open, 3-3, 3-5, B-5
 - quit, 3-3, 3-4, B-5
 - send, 3-3, 3-10, B-5
 - set, 3-3, 3-12, B-5
 - status, 3-3, 3-16, B-6
 - toggle, 3-3, 3-13, B-6
- telnet> prompt, 3-3, 3-4, 3-10, B-4
- telnet.pm program, 3-2
- Temporary directories, 2-56
- Tenex file-transfer type, 2-18, A-18, A-19
- tenex subcommand, 2-18, A-18
- Terminal response (beep), 2-21, A-6
- Terminal types
 - defining, 3-9, 3-18
 - negotiation, 3-9
 - setting, 3-8, 3-9, 3-18
- TERMINAL_TYPE TELNET protocol
 - command, 3-9, 3-18
- Terminal-type definition files, 3-18
- Terminating a connection
 - FTP, 2-12, A-8
 - TELNET, 3-4, B-4
- Text files, 2-18
- tftp command, 4-2, C-1
- TFTP. See Trivial File Transfer Protocol (TFTP)
- toggle subcommand, 3-3, 3-9, 3-13, 3-14, B-5, B-6
- Transferring files, 2-1, A-2, A-4
 - abbreviations, 2-22
 - ASCII, A-6, A-19
 - between Stratus hosts, 2-18, 2-20
 - binary, A-6, A-19
 - equivalent to copying, 2-31, A-4
 - extended sequential, 2-20
 - file-transfer types, 2-17, A-19
 - fixed-length records, 2-18, 2-20, A-19
 - image, A-9, A-19
 - indexed, 2-18
 - local host to remote host, 2-32, A-5, A-15, A-17
 - multiple, 2-22, 2-32, 2-34, 2-36, 2-37, A-12
 - nontext, 2-18
 - opening a proxy connection, 2-11, A-14
 - relative, 2-18
 - remote host to local host, 2-35, 2-36, A-8, A-11, A-15
 - sequential, 2-17, 2-18, 2-20
 - sequential save, 2-20
 - star names, 2-22, 2-34, 2-37
 - stream, 2-20
 - subcommands, 2-31
 - Tenex, 2-18, A-18, A-19
 - text, 2-18
 - verifying, 2-35
- Translating file-transfer file names, 2-25, A-14
- Transmission Control Protocol/Internet Protocol (TCP/IP), 1-1

Transmission mode, 2-15, A-12
Trivial File Transfer Protocol (TFTP), 1-1
Turning off key sequences for input requests, 3-13, B-6
Turning operating modes on and off, 3-13, B-6
`type` subcommand, 2-18, A-19

U

Unique file-name assignment, 2-24, A-17, A-18
UNIX operating system, 3-20
Uppercase characters mapped to lowercase characters, 2-25, A-6
User registrations
 FTP, 2-7, 2-56
 TELNET, 3-8, 3-17
`user` subcommand, 2-5, 2-7, 2-8, A-19
Using star names and abbreviations, 2-3

V

Verbose mode, 2-3, 2-21, A-3, A-19, A-20
`verbose` subcommand, 2-21, A-3, A-19
Verifying overwrites and deletions, 2-21, A-17, A-18
Virtual terminal connections, 3-2
VOS command interface, 3-19

W

Well-known ports, 2-6, 3-6
Window terminal software, 3-19
Working directory, 2-40, A-15
Working in input mode, 3-8