SCO™ TCP/IP

Derived from

LACHMAN™ SYSTEM V STREAMS TCP

User's Reference

The Santa Cruz Operation™

SCO TCP/IP was developed by Lachman Associates.
SCO TCP/IP is derived from LACHMAN™ SYSTEM V STREAMS TCP, a joint development of Lachman Associates and Convergent Technologies.

This document was typeset with an IMAGEN® 8/300 Laser Printer.

SCO Document Number: 11-25-89-1.1.0D
Printed: 12/4/89

Contents

Networking Commands (TC)

intro

Introduction to networking commands

Description

This section describes publicly accessible networking utilities in alphabetical order.

See Also

The (ADMN) and (ADMP) sections for network administration commands.

Diagnostics

Upon termination, each command returns two bytes of status, one supplied by the system giving the cause for termination, and (in the case of normal termination) one supplied by the program. The former byte is 0 for normal termination; the latter is customarily 0 for successful execution, nonzero to indicate troubles such as erroneous parameters, bad or inaccessible data, or other inability to cope with the task at hand. The second byte is called "exit code," "exit status" or "return code," and is described only where special conventions are involved.

finger

User information look-up program

Syntax

finger [options] name ...

Description

By default, *finger* lists the login name, full name, terminal name and write status (as an * before the terminal name if write permission is denied), idle time, log-in time, and office location and phone number (if they are known) for each current UNIX user. (Idle time is minutes if it is a single integer, hours and minutes if a **:** is present, or days and hours if a **d** is present.)

A longer format also exists and is used by *finger* whenever a list of people's names is given. (Account names as well as first and last names of users are accepted.) This format is multi-line, and includes all the information described above as well as the user's home directory and login shell, any plan which the person has placed in the file **.plan** in their home directory, and the project on which they are working from the **.project** file, also in the home directory.

finger may be used to look up users on a remote machine. The format is to specify the user as user@host. If the user name is left off, the standard format listing is provided on the remote machine.

finger options include:

-m
Match arguments only on user name.

-l Force long output format.

-p Suppress printing of the **.plan** files

-s Force short output format.

Files

/etc/utmp	who file (current users)
/etc/wtmp	who file (past logins)
/etc/passwd	for users names, offices, ...
$HOME/.lastlogin	last log-in information
$HOME/.plan	plans
$HOME/.project	projects

See Also

who(C), fingerd(ADMN)

Notes

Only the first line of the **.project** file is printed.
There is no way to pass arguments to the remote machine, as *finger* uses an internet standard port.

ftp

ARPANET file-transfer program

Syntax

ftp [**-v**] [**-d**] [**-i**] [**-n**] [**-g**] [**host**]

Description

ftp is the user interface to the ARPANET standard File Transfer Protocol. The program allows a user to transfer files to and from a remote network site.

The client host with which *ftp* is to communicate may be specified on the command line. If this is done, *ftp* immediately attempts to establish a connection to an FTP server on that host; otherwise, *ftp* will enter its command interpreter and await instructions from the user. When *ftp* is awaiting commands from the user, the prompt **ftp>** is provided to the user. The following commands are recognized by *ftp*:

! [*command* [*args*]]
Invoke an interactive shell on the local machine. If there are arguments, the first is taken as a command to execute directly, with the rest of the arguments as its arguments.

$ *macro-name* [*args*]
Execute the macro *macro-name* that was defined with the **macdef** command. Arguments are passed to the macro unglobbed.

account [*passwd*]
Supply a supplemental password which is required by a remote system for access to resources once a login has been successfully completed. If no argument is included, the user is prompted for an account password in a non-echoing input mode.

append *local-file* [*remote-file*]
Append a local file to a file on the remote machine. If *remote-file* is left unspecified, the local file name is used in naming the remote file after being altered by any **ntrans** or **nmap** setting. File transfer uses the current settings for **type**, **format**, **mode**, and **structure**.

ascii
Set the file transfer **type** to network ASCII. This is the default type.

bell
Arrange for a bell to sound after each file-transfer command is completed.

binary
Set the file transfer **type** to support binary image transfer.

bye
Terminate the FTP session with the remote server and exit *ftp*. An end-of-file will also terminate the session and exit.

case
Toggle remote computer file-name case mapping during **mget** commands. When **case** is on (default is off), remote computer file names with all letters in uppercase are written in the local directory with the letters mapped to lowercase.

cd *remote-directory*
Change the working directory on the remote machine to *remote-directory*.

cdup
Change the remote machine working directory to the parent of the current remote machine working directory.

close
Terminate the FTP session with the remote server, and return to the command interpreter. Any defined macros are erased.

cr
Toggle carriage-return stripping during ascii type file retrieval. Records are denoted by a carriage return/linefeed sequence during ascii type file transfer. When **cr** is on (the default), carriage returns are stripped from this sequence to conform to the UNIX single-linefeed record delimiter. Records on non-UNIX remote systems may contain single linefeeds; when an ascii type transfer is made, these linefeeds may be distinguished from a record delimiter only when **cr** is off.

delete *remote-file*
Delete the file *remote-file* on the remote machine.

debug [*debug-value*]
Toggle debugging mode. If an optional *debug-value* is specified, it is used to set the debugging level. When debugging is on, *ftp* prints each command sent to the remote machine, preceded by the string:

-->

dir [*remote-directory*] [*local-file*]
Print a listing of the directory contents in the directory, *remote-directory*, and, optionally, place the output in *local-file*. If no directory is specified, the current working directory on the remote machine is used. If no local file is specified, or *local-file* is **-**, output comes to the terminal.

disconnect
A synonym for **close**.

form *format*
Set the file transfer **form** to *format*. The default format is file.

get *remote-file* [*local-file*]
Retrieve the *remote-file* and store it on the local machine. If the local file name is not specified, it is given the same name it has on the remote machine, subject to alteration by the current **case**, **ntrans**, and **nmap** settings. The current settings for **type**, **form**, **mode**, and **structure** are used while transferring the file.

glob
Toggle filename expansion for **mdelete**, **mget** and **mput**. If globbing is turned off with **glob**, the file name arguments are taken literally and not expanded. Globbing for **mput** is done as in *sh*(C). For **mdelete** and **mget**, each remote file name is expanded separately on the remote machine and the lists are not merged. Expansion of a directory name is likely to be different from expansion of the name of an ordinary file. The exact result depends on the foreign operating system and *ftp* server, and can be previewed with the command:

mls *remote-files* **-**

Note: **mget** and **mput** are not meant to transfer entire directory subtrees of files. That can be done by transferring a *tar*(C) archive of the subtree (in binary mode).

hash
Toggle hash-sign (#) printing for each data block transferred. The size of a data block is BUFFERSIZE bytes. BUFFERSIZE is defined in the *ftp* source.

help [*command*]
Print an informative message about the meaning of *command*. If no argument is given, *ftp* prints a list of the known commands.

lcd [*directory*]
Change the working directory on the local machine. If no *directory* is specified, the user's home directory is used.

ls [*remote-directory*] [*local-file*]
Print an abbreviated listing of the contents of a directory on the remote machine. If *remote-directory* is left unspecified, the

current working directory is used. If no local file is specified, or if *local-file* is -, the output is sent to the terminal.

macdef *macro-name*
Define a macro. Subsequent lines are stored as the macro *macro-name*; a null line (consisting of consecutive newline characters in a file or carriage returns from the terminal) terminates macro input mode. There are limits of 16 macros and 4096 total characters in all defined macros. Macros remain defined until a **close** command is executed. The macro processor interprets $ and \ as special characters. A $ followed by a number (or numbers) is replaced by the corresponding argument on the macro-invocation command line. A $ followed by an i signal to the macro processor that the executing macro is to be looped. On the first pass, $i is replaced by the first argument on the macro-invocation command line; on the second pass it is replaced by the second argument; and so on. A \ followed by any character is replaced by that character. Use the \ to prevent special treatment of the $.

mdelete [*remote-files*]
Delete the *remote-files* on the remote machine.

mdir *remote-files local-file*
Like **dir**, except multiple remote files may be specified. If interactive prompting is on, *ftp* will prompt the user to verify that the last argument is indeed the target local file for receiving **mdir** output.

mget *remote-files*
Expand the *remote-files* on the remote machine and do a **get** for each file name thus produced. See **glob** for details on the filename expansion. Resulting file names will then be processed according to **case**, **ntrans**, and **nmap** settings. Files are transferred into the local working directory, which can be changed with the command:

lcd *directory*

new local directories can be created with the command:

! mkdir *directory*

mkdir *directory-name*
Make a directory on the remote machine.

mls *remote-files local-file*
Like **ls**, except multiple remote files may be specified. If interactive prompting is on, *ftp* will prompt the user to verify that the last argument is indeed the target local file for receiving **mls** output.

mode [*mode-name*]
Set the file-transfer **mode** to *mode-name*. The default mode is stream mode.

mput *local-files*
Expand wild cards in the list of local files given as arguments and do a **put** for each file in the resulting list. See **glob** for details of filename expansion. Resulting file names will then be processed according to **ntrans** and **nmap** settings.

nmap [*inpattern outpattern*]
Set or unset the filename-mapping mechanism. If no arguments are specified, the filename-mapping mechanism is unset. If arguments are specified, remote filenames are mapped during those **mput** commands and **put** commands issued without a specified remote target filename. If arguments are specified, local filenames are mapped during those **mget** commands and **get** commands issued without a specified local target filename. The **nmap** command is useful when connecting to a non-UNIX remote computer with different file-naming conventions or practices. The mapping follows the pattern set by *inpattern* and *outpattern*. *Inpattern* is a template for incoming filenames (which may already have been processed according to the **ntrans** and **case** settings). Variable templating is accomplished by including the sequences $1, $2, ..., $9 in *inpattern*. Use \ to prevent this special treatment of the $ character. All other characters are treated literally, and are used to determine the **nmap** *inpattern* variable values. For example, given *inpattern* $1.$2 and the remote file name *mydata.data*, $1 would have the value *mydata* and $2 would have the value *data*. The *outpattern* determines the resulting mapped filename. The sequences $1, $2, ..., $9 are replaced by any value resulting from the *inpattern* template. The sequence $0 is replaced by the original filename. Additionally, the sequence [*seq1*,*seq2*] is replaced by *seq1* unless *seq1* is a null string; in that case, it is replaced by *seq2*. For example, the command **nmap $1.$2.$3 [$1,$2].[$2,file]** would yield the output filename *myfile.data* for input filenames *myfile.data* and *myfile.data.old*, *myfile.file* for the input filename *myfile*, and *myfile.myfile* for the input filename *myfile.myfile*. Spaces may be included in *outpattern*, as in the example: nmap $1 |sed "s/ *$//" > $1 . Use the \ character to prevent special treatment of the $, [,] and , characters.

ntrans [*inchars* [*outchars*]]
Set or unset the filename-character translation mechanism. If no arguments are specified, the filename-character translation mechanism is unset. If arguments are specified, characters in remote filenames are translated during those **mput** commands and **put** commands issued without a specified remote target filename. If arguments are specified, characters in local filenames are translated during those **mget** commands and **get** commands issued without a specified local target filename. This command is useful when connecting to a non-UNIX remote computer with different file-naming conventions or practices. Characters in a filename matching a character in *inchars* are replaced with the corresponding character in *outchars*. If the character's position in *inchars* is longer than the length of *outchars*, the character is deleted from

the file name. For example, the command **ntrans * .** would modify the filenames of files copied with *ftp*. This command translates the character "*" to the character "." in filenames. Thus, if you used the *ftp* command **get test*exe**, the file test*exe would be copied as test.exe.

open *host* [*port*]
Establish a connection to the specified *host* FTP server. An optional port number may be supplied, in which case, *ftp* will attempt to contact an FTP server at that port. If the *auto-login* option is on (default), *ftp* will also attempt to log the user automatically in to the FTP server. (See below.)

prompt
Toggle interactive prompting. Interactive prompting occurs during multiple file transfers to allow the user to retrieve or store files selectively. If prompting is turned off (default is on), any **mget** or **mput** will transfer all files, and any **mdelete** will delete all files.

proxy *ftp-command*
Execute an *ftp* command on a secondary control connection. This command allows simultaneous connection to two remote *ftp* servers for transferring files between the two servers. The first **proxy** command should be an **open**, to establish the secondary control connection. Enter the command **proxy ?** to see other *ftp* commands executable on the secondary connection. The following commands behave differently when prefaced by **proxy**: **open** will not define new macros during the auto-login process; **close** will not erase existing macro definitions; **get** and **mget** transfer files from the host on the primary control connection to the host on the secondary control connection; and **put**, **mput**, and **append** transfer files from the host on the secondary control connection to the host on the primary control connection. Third-party file transfers depend upon support of the *ftp* protocol PASV command by the server on the secondary control connection.

put *local-file* [*remote-file*]
Store a local file on the remote machine. If *remote-file* is left unspecified, the local file name is used in naming the remote file after processing according to any **ntrans** or **nmap** settings. File transfer uses the current settings for **type**, **format**, **mode**, and **structure**.

pwd
Print the name of the current working directory on the remote machine.

quit
A synonym for **bye**.

quote *arg1 arg2 ...*
The arguments specified are sent verbatim to the remote FTP server.

recv *remote-file* [*local-file*]
A synonym for **get**.

remotehelp [*command-name*]
Request help from the remote FTP server. If a *command-name* is specified, it is supplied to the server as well.

rename [*from*] [*to*]
Rename the file *from* on the remote machine, to the file *to*.

reset
Clear reply queue. This command re-synchronizes command/reply sequencing with the remote *ftp* server. Resynchronization may be necessary following a violation of the *ftp* protocol by the remote server.

rmdir *directory-name*
Delete a directory on the remote machine.

runique
Toggle storage of files on the local system with unique filenames. If a file already exists with a name equal to the target local filename for a **get** or **mget** command, a .1 is appended to the name. If the resulting name matches another existing file, a .2 is appended to the original name. If this process continues up to .99, an error message is printed, and the transfer does not take place. The generated unique filename will be reported. Note that **runique** will not affect local files generated from a shell command. The default value is off.

send *local-file* [*remote-file*]
A synonym for **put**.

sendport
Toggle the use of PORT commands. By default, *ftp* will attempt to use a PORT command when establishing a connection for each data transfer. The use of PORT commands can prevent delays when performing multiple file transfers. If the PORT command fails, *ftp* will use the default data port. When the use of PORT commands is disabled, no attempt will be made to use PORT commands for each data transfer. This is useful for certain FTP implementations that do ignore PORT commands but, incorrectly, indicate that they have been accepted.

status
Show the current status of *ftp*.

struct [*struct-name*]
Set the file transfer *structure* to *struct-name*. By default, file structure is used.

sunique
Toggle storage of files on a remote machine under unique file names. Remote *ftp* server must support *ftp* protocol STOU command for successful completion. The remote server will report a unique name. Default value is off.

tenex
Set the file transfer type to that needed to talk to TENEX machines.

trace
Toggle packet-tracing.

type [*type-name*]
Set the file transfer **type** to *type-name*. If no type is specified, the current type is printed. The default type is network ASCII.

user *user-name* [*password*] [*account*]
Identify yourself to the remote FTP server. If the password is not specified and the server requires it, *ftp* will prompt the user for it (after disabling local echo). If an account field is not specified, and the FTP server requires it, the user will be prompted for it. When an account field is specified, an account command will be relayed to the remote server after the log-in sequence is completed, if the remote server did not require it for logging in. Unless *ftp* is invoked with auto-login disabled, this process is done automatically on initial connection to the FTP server.

verbose
Toggle verbose mode. In verbose mode, all responses from the FTP server are displayed to the user. In addition, if verbose is on, when a file transfer completes, statistics regarding the efficiency of the transfer are reported. By default, verbose is on.

xmkdir *directory-name*
Make a directory on the remote machine. This sends an XMKD command instead of MKD, and is useful for backwards compatability with 4.2BSD UNIX machines.

xpwd
Print the name of the current working directory on the remote machine. This sends an XPWD command instead of PWD, and is useful for backwards compatability with 4.2BSD UNIX machines.

xrmdir *directory-name*
Delete a directory on the remote machine. This sends an XRMD command instead of RMD, and is useful for backwards compatability with 4.2BSD UNIX machines.

? [*command*]
A synonym for **help**.

Command arguments which have embedded spaces may be quoted with quotation (") marks.

Aborting a File Transfer

To abort a file transfer, use the terminal interrupt key (usually Ctrl-C). The sending of transfers is immediately halted. The receiving of transfers is halted by sending an *ftp* protocol ABOR command to the remote server and discarding any further data received. The speed at which this is accomplished depends upon the remote server's support for ABOR processing. If the remote server does not support the ABOR command, an **ftp>** prompt will not appear until the remote server has finished sending the requested file.

The terminal-interrupt key sequence will be ignored when *ftp* has completed any local processing and is awaiting a reply from the remote server. A long delay in this mode may result from the ABOR processing described above, or from unexpected behavior by the remote server, including violations of the *ftp* protocol. If the delay results from unexpected remote server behavior, the local *ftp* program must be killed by hand.

File Naming Conventions

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

1) If the file name - is specified, the **stdin** (for reading) or **stdout** (for writing) is used.

2) If the first character of the file name is |, the remainder of the argument is interpreted as a shell command. Then *ftp* forks a shell, using *popen*(S) with the argument supplied, and reads from the stdout (or writes to the stdin). If the shell command includes spaces, the argument must be quoted, for instance, "| ls -lt". A particularly useful example of this mechanism is: dir |more.

3) Failing the above checks, if globbing is enabled, local file names are expanded according to the rules used in the *sh*(C); see the **glob** command. If the *ftp* command expects a single local file (such as **put**), only the first filename generated by the globbing operation is used.

4) For **mget** commands and **get** commands with unspecified local file names, the local filename is the remote filename, which may be altered by a **case**, **ntrans**, or **nmap** setting. The resulting filename may then be altered if **runique** is on.

5) For **mput** commands and **put** commands with unspecified remote file names, the remote filename is the local filename, which may be altered by an **ntrans** or **nmap** setting. The resulting filename may then be altered by the remote server if **sunique** is on.

File Transfer Parameters

The FTP specification specifies many parameters that may affect a file transfer. The **type** may be one of *ascii*, *image* (binary), *ebcdic*, and *local byte size* (for PDP-10's and PDP-20's mostly). The *ftp* command supports the ascii and image types of file transfer, plus local byte size 8 for **tenex** mode transfers.

The *ftp* command supports only the default values for the remaining file transfer parameters: **mode , form ,** and **struct .**

Options

Options may be specified at the command line, or to the command interpreter.

The **-v** (verbose on) option forces **ftp** to show all responses from the remote server, as well as report on data transfer statistics. Ordinarily, this is on by default, unless the standard input is not a terminal.

The **-n** option restrains *ftp* from attempting auto-login upon initial connection. If auto-login is enabled, *ftp* will check the file **.netrc** (discussed below) in the user's home directory for an entry describing an account on the remote machine. If no entry exists, *ftp* will prompt for the remote machine log-in name (default being the user identity on the local machine), and, if necessary, prompt for a password and an account with which to log in.

The **-i** means there is no interactive prompt.

The **-d** option enables debugging.

The **-g** option disables file-name globbing.

The .netrc File

The **.netrc** file contains login and initialization information used by the auto-login process. It resides in the user's home directory. The following tokens are recognized; they may be separated by spaces, tabs, or new-lines:

machine *name*
Identify a remote machine name. The auto-login process searches the **.netrc** file for a **machine** token that matches the remote ma-

chine specified on the *ftp* command line or as an **open** command argument. Once a match is made, the subsequent .netrc tokens are processed, stopping when the end of file is reached or another **machine** token is encountered.

login *name*
Identify a user on the remote machine. If this token is present, the auto-login process will initiate a login using the specified name.

password *string*
Supply a password. If this token is present, the auto-login process will supply the specified string when the remote server requires a password as part of the login process. Note that if this token is present in the **.netrc** file, *ftp* will abort the auto-login process if the **.netrc** is readable by anyone besides the user.

account *string*
Supply an additional account password. If this token is present, the auto-login process will supply the specified string when the remote server requires an additional account password, or the auto-login process will initiate an ACCT command when it does not.

macdef *name*
Define a macro. This token functions like the *ftp* **macdef** command. A macro is defined with the specified name; its contents begin with the next .netrc line and continue until a null line (consecutive new-line characters) is encountered. If a macro named *init* is defined, it is automatically executed as the last step in the auto-login process.

Notes

Correct execution of many commands depends upon proper behavior by the remote server.

An error in the treatment of carriage returns in the 4.2BSD UNIX ascii-mode transfer code has been corrected. This correction may result in incorrect transfers of binary files to and from 4.2BSD servers using the ascii type. Avoid this problem by using the binary image type.

hostname

host name resolution description

Description

Hostnames are domains, where a domain is a hierarchical, dot-separated list of subdomains; for example, the machine laiter, in the Lachman subdomain of the COM subdomain of the ARPANET would be represented as

laiter.Lachman.COM

(with no trailing dot).

Hostnames are often used with network client and server programs, which must generally translate the name to an address for use. (This function is generally performed by the library routine *gethostbyname* (SSC).) Hostnames are resolved by the internet name resolver in the following fashion.

If the name consists of a single component, i.e. contains no dot, and if the environment variable "HOSTALIASES" is set to the name of a file, that file is searched for an string matching the input hostname. The file should consist of lines made up of two white-space separated strings, the first of which is the hostname alias, and the second of which is the complete hostname to be substituted for that alias. If a case-sensitive match is found between the hostname to be resolved and the first field of a line in the file, the substituted name is looked up with no further processing.

If the input name ends with a trailing dot, the trailing dot is removed, and the remaining name is looked up with no further processing.

If the input name does not end with a trailing dot, it is looked up in the local domain and its parent domains until either a match is found or fewer than 2 components of the local domain remain. For example, in the domain CHI.Lachman.COM, the name flaime.STG will be checked first as flaime.STG.CHI.Lachman.COM and then as flaime.STG.Lachman.COM. Flaime.STG.COM will not be tried, as the there is only one component remaining from the local domain.

See Also

gethostent(SFF),resolver(ADMN), mailaddr(ADMN),
named(ADMN).
RFC883.

logger

make entries in the system log

Syntax

logger [**-t** tag] [**-p** pri] [**-i**] [**-f** file] [message ...]

Description

Logger provides a program interface to the *syslog*(3) system log module.

A message can be given on the command line, which is logged immediately, or a file is read and each line is logged.

Examples

logger System rebooted

logger -p local0.notice -t OPER -f /tmp/msg

See Also

syslog(SFF), syslogd(ADMN).

netstat

Show network status

Syntax

netstat [**-AainrsS**] [**-f** *address_family*] [**-I** *interface*] [**-p** *protocol_name*] [*interval*] [*namelist*] [*corefile*]

Description

The *netstat* command symbolically displays the contents of various network-related data structures. The options have the following meanings:

-A show the address of any associated protocol control blocks; used for debugging

-a show the state of all sockets; normally sockets used by server processes are not shown

-i show the state of interfaces that have been auto-configured. (Interfaces statically configured into a system, but not located at boot time, are not shown.)

-n show network addresses as numbers. (Normally *netstat* interprets addresses and attempts to display them symbolically.)

-r show the routing tables

-s show per-protocol statistics

-S show serial line configuration

-f limit statistics and control block displays to *address-family*. The only address-family currently supported is **inet**

-I show interface state for *interface* only.

-p limit statistics and control block displays to *protocol_name*, such as **tcp.**

The arguments *namelist* and *corefile* allow substitutes for the defaults **/unix** and **/dev/kmem**.

If an *interval* is specified, *netstat* will continuously display the information regarding packet traffic on the configured network interfaces, pausing *interval* seconds before refreshing the screen.

There are a number of display formats, depending on the information presented. The default display, for active sockets, shows the local and remote addresses, send and receive queue sizes (in bytes), protocol, and, optionally, the internal state of the protocol.

Address formats are of the form host.port or network.port if a socket's address specifies a network but no specific host address. When known, the host and network addresses are displayed symbolically according to the data bases **/etc/hosts** and **/etc/networks,** respectively. If a symbolic name for an address is unknown, or if the **-n** option is specified, the address is printed in the Internet dot format; refer to *rhosts*(SFF) for more information regarding this format. Unspecified, or wildcard, addresses and ports appear as *.

The interface display provides a table of cumulative statistics regarding transferred packets, errors, and collisions. The network address (currently Internet specific) of the interface and the maximum transmission unit (mtu) are also displayed.

The routing table display indicates the available routes and their status. Each route consists of a destination host or network and a gateway to use in forwarding packets. The flags field shows the state of the route (U if up), and whether the route is to a gateway (G). Direct routes are created for each interface attached to the local host. The refcnt field gives the current number of active uses of the route. Connection-oriented protocols normally hold onto a single route for the duration of a connection, while connectionless protocols obtain a route then discard it. The use field provides a count of the number of packets sent using that route. The interface entry indicates the network interface utilized for the route.

When *netstat* is invoked with an *interval* argument, it displays a running count of statistics related to network interfaces. This display consists of a column summarizing information for all interfaces and a column for the interface with the most traffic since the system was last rebooted. The first line of each screen of information contains a summary since the system was last rebooted. Subsequent lines of output show values accumulated over the preceding interval.

The serial line display shows the mapping of serial line units to serial devices. The baud rate and protocols in use are also shown.

See Also

slattach(ADMN), hosts(ADMN), networks(SSC), protocols(SFF), services(SFF).

Bugs

Interface statistics are dependent on the link driver. If it does not

attach itself to the *ifstats* structure in the kernel, the message "No Statistics Available" will be printed for that interface.

nslookup

Query name servers interactively

Syntax

nslookup [*host-to-find* / - [*server address* / *server name*]]

Description

The *nslookup* command queries DARPA Internet domain name servers. Interactive mode allows the user to query the name server for information about various hosts and domains or to print a list of hosts in the domain. Non-interactive mode prints just the name and Internet address of a host or domain.

Arguments

Interactive mode is entered in the following cases:

a) when no arguments are given (the default name server will be used), and

b) when the first argument is a hyphen (-) and the second argument is the host name of a name server.

Non-interactive mode is used when the name of the host to be looked up is given as the first argument. The optional second argument specifies a name server.

Interactive Commands

Commands may be interrupted at any time by typing a control-C. To exit, type a control-D (EOF). The command line length must be less than 80 characters. **N.B.** an unrecognized command will be interpreted as a host name.

host [server]
Look up information for *host* using the current default server, or using *server* if it is specified.

server *domain*
lserver *domain*
Change the default server to *domain*. The **lserver** command uses the initial server to look up information about *domain* while **server** uses the current default server. If an authoritative answer can't be found, the names of servers that might have the answer are returned.

root
Changes the default server to the server for the root of the domain name space. Currently, the host sri-nic.arpa is used. (This command is a synonym for the **lserver sri-nic.arpa**.) The name of the root server can be changed with the **set root** command.

finger [*name*] [> *filename*]
finger [*name*] [>> *filename*]
Connects with the finger server on the current host. The current host is defined when a previous lookup for a host was successful and returned address information. (See the **set querytype=A** command.) *Name* is optional. > and >> can be used to redirect output in the usual manner.

ls *domain* [> *filename*]
ls *domain* [>> *filename*]
ls -a *domain* [> *filename*]
ls -a *domain* [>> *filename*]
ls -h *domain* [> *filename*]
ls -h *domain* [>> *filename*]
ls -d *domain* [> *filename*]
List the information available for *domain*. The default output contains host names and their Internet addresses. The **-a** option lists aliases of hosts in the domain. The **-h** option lists CPU and operating system information for the domain. The **-d** option lists all contents of a zone transfer. When output is directed to a file, hash marks are printed for every 50 records received from the server.

view *filename*
Sorts and lists the output of the previous **ls** command with *more*(C).

help
? Prints a brief summary of commands.

set *keyword*[=*value*]
This command is used to change state information that affects the lookups. Valid keywords are:

all Prints the current values of the various options to **set**. Information about the current default server and host is also printed.

[no]debug
Turn debugging mode on. A lot more information is printed about the packet sent to the server and the resulting answer.
(Default = nodebug; abbreviation = [no]deb)

[no]d2
Turn exhaustive debugging mode on. Essentially all fields of every packet are printed.
(Default = nod2)

[no]defname
Append the default domain name to every lookup.
(Default = defname; abbreviation = [no]def)

[no]search
With defname, search for each name in parent domains for the current domain.
(Default = search)

domain=*name*
Change the default domain name to *name*. The default domain name is appended to all look-up requests if the **defname** option has been set. The search list is set to parents of the domain with at least two components in their names.
(Default = value in hostname or /etc/resolv.conf; abbreviation = do)

type=*value*

querytype=*value*
Change the type of information returned from a query to one of:

A	the host's Internet address (the default)
CNAME	the canonical name for an alias
HINFO	the host CPU and operating system type
MD	the mail destination
MX	the mail exchanger
MG	the mail group member
MINFO	the mailbox or mail list information

MR the mail rename domain name

NS nameserver for the named zone

Other types specified in the RFC1035 document are valid but are not very useful.
(Abbreviation = q)

[no]recurse
Tell the name server to query other servers if it does not have the information.
(Default = recurse; abbreviation = [no]rec)

retry=*number*
Set the number of retries to *number*. When a reply to a request is not received within a certain amount of time (changed with **set timeout**), the request is resent. The retry value controls how many times a request is resent before giving up.
(Default = 2; abbreviation = ret)

root=*host*
Change the name of the root server to *host*. This affects the **root** command.
(Default = sri-nic.arpa; abbreviation = ro)

timeout=*number*
Change the time-out interval for waiting for a reply to *number* seconds.
(Default = 10 seconds; abbreviation = t)

[no]vc
Always use a virtual circuit when sending requests to the server.
(Default = novc; abbreviation = [no]v)

Diagnostics

If the look-up request was not successful, an error message is printed. Possible errors are:

Time-out
The server did not respond to a request after a certain amount of time (changed with **set timeout**=*value*) and a certain number of retries (changed with **set retry**=*value*).

No information
Depending on the query type set with the **set querytype** command, no information about the host is available, though the host name is valid.

Non-existent domain
The host or domain name does not exist.

Connection refused
Network is unreachable
The connection to the name or finger server could not be made at the current time. This error commonly occurs with **finger** requests.

Server failure
The name server found an internal inconsistency in its database and could not return a valid answer.

Refused
The name server refused to service the request.

The following error should not occur and it indicates a bug in the program:

Format error
The name server found that the request packet was not in the proper format.

Files

/etc/resolv.conf initial domain name and name server addresses.
/usr/lib/nslookup.hlp help file

See Also

resolver(SLIB), resolver(SFF), named(ADMN), RFC974, RFC1034, RFC1035

rcmd

Remote shell command execution

Syntax

rcmd node [**-l** user] [**-n**] [command]

Description

rcmd sends *command* to *node* for execution. It passes the resulting remote command its own standard input and outputs the remote command's standard output and standard error. *Command* can consist of more than one parameter. The second, simplified form of the command is equivalent to the first, but is only available if the system administrator previously ran *mkhosts*(ADMN). Interrupt, quit, and terminate signals received by *rcmd* are also received by the remote command; *rcmd* normally terminates at the same time as the remote command.

If *command* is omitted, *rcmd* simply runs *rlogin*(TC).

By default, the command belongs to the user on the remote node with the same name as the user who ran *rcmd*. This means that the resulting processes belong to the remote user and begin with the remote user's home directory as their working directory. Options permit you to specify another user on *node* as the owner. In any case, the remote system must have declared the local user equivalent to the remote user: an entry in **/etc/hosts.equiv** or in a **.rhosts** file in the current directory (normally the home directory) of the target user will demonstrate equivalence. [See *rcmd*(SLIB).]

rcmd understands the following options:

-l *user* The command is to belong to *user* on *node*.

-n Prevent the remote command from blocking on input by making its standard input be **/dev/null** instead of the standard input of *rcmd*.

If **-n** is not specified, *rcmd* reads the local standard input, regardless of whether the remote machine reads standard input.

Examples

The following command runs **who** on a node called "central," putting the output in a file on the local machine.

```
rcmd central who > /tmp/c.who
```

The next example puts the same output on the remote machine.

```
rcmd central who \> /tmp/c.who
```

Files

$HOME/.rhosts	(on the target machine)
/etc/hosts.equiv	(on the target machine)

See Also

mkhosts(ADMN), rlogin(TC), rshd(ADMN), rhosts(SFF).

Requirements

rshd(ADMN) must be running on the target machine.

Notes

In some installations, this command is called **rsh**, so as to be like other versions of the software.

Unlike **rlogin** and **telnet**, **rcmd** does not actually use a pseudo-tty. The remote program can only read and/or write; therefore, programs such as **more** or **vi** will hang. Hit the <Break> key to continue.

Warnings

As the above examples illustrate, metacharacters to be interpreted by the remote shell must be hidden from the local shell. Thus:

```
rcmd central cd /etc ; cat passwd
```

clearly doesn't do what was intended because the semicolon is interpreted by the local shell, not the remote shell, and the remote shell never even sees the **cat** command. Either of the following commands properly escapes the semicolon:

```
rcmd central cd /etc \; cat passwd
rcmd central 'cd /etc ; cat passwd'
```

Credit

This document was developed at the University of California at Berkeley and is used with permission.

rcp

Remote file copy

Syntax

rcp [**-r**] [**-p**] file1 [file2 ...] target

Description

rcp copies files between two nodes. *rcp* works like the **cp** command (see *cp*(C)), with some extensions.

file1 is copied to *target*. If *target* is a directory, one or more files are copied into that directory; the copies have the same names as the originals.

File and directory names follow a convention which is an extension of the normal UNIX convention. Names take one of three forms:

user **@** *host* **:** *path*
host **:** *path*
path

where

host is the name of the system which contains or will contain the file. If no host is specified (the simple *path* form of the name), the system on which the command is executed is assumed.

user is the name of a user on the specified system. If no user is specified in the name, then the user on the remote system whose name is the same as the user who executed the *rcp* command is used. (That is, this rule applies if the *host*:*path* or *path* form of the name was used.)

Access to the file system is as if by the specified user who has just logged in. Created files belong to the specified user and the specified user's group (taken from the password file). File and directory modifications can only occur if the specified user has permission to make them. If *path* does not begin with a slant (/), it is assumed to be relative to the specified user's home directory.

For you to use a user name on a remote system, the remote system must have declared it equivalent to your user name. See *rhosts*(SFF).

path is a conventional UNIX path name. *Path* can include file-name-generation sequences (*, ?, [...]); it may be necessary to quote these to prevent their expansion on the local system.

The **-r** (recursive) option copies directory hierarchies. If a file specified for copying is a directory and **-r** is specified, the entire hierarchy under it is copied. When **-r** is specified, *target* must be a directory.

When **-r** is not specified, copying directories is an error.

By default, the mode and owner of *file2* are preserved if it already existed; otherwise, the mode of the source file modified by the *umask*(SSC) on the destination host is used.

The **-p** option causes *rcp* to attempt to preserve (duplicate) in its copies the modification times and modes of the source files, ignoring the *umask*.

Note that a third system (not the source or target system of the copy) can execute *rcp*.

Examples

The following examples are executed on system alpha, by user fred. Alpha is networked to beta and gamma.

The first example copies *list* from fred's home directory on alpha to fred's home directory on beta.

```
rcp list beta:list
```

The next example copies a directory hierarchy. The original is rooted at *src* in fred's home directory on beta. The copy is to be rooted in *src* in the working directory.

```
rcp -r beta:src .
```

Finally, fred copies a file from mike's home directory on beta to **/usr/tmp** on gamma; the copy on gamma is to belong to deb. Both mike and deb must have previously declared fred on alpha equivalent to their own user names; see *rhosts*(SFF).

```
rcp mike@beta:file deb@gamma:/usr/tmp
```

Note that *junk* is not placed in deb's home directory because the *path* part of the name begins with a slash.

Files

/etc/hosts.equiv
$HOME/.rhosts

See Also

ftp(TC)

Requirements

Both nodes involved in the copy must be running the *rshd*(ADMN) server.

Diagnostics

Most diagnostics are self-explanatory. "Permission denied" means either that the remote user does not have permission to do what you want or that the remote user is not equivalent to you.

Warnings

If a remote shell invoked by *rcp* has output on startup, *rcp* will get confused. This is never a problem with *sh*(C), because it is not called as a log-in shell.

The **-r** option doesn't work correctly if the copy is purely local, because it relies on underlying support from **cp** , which is only available on BSD-derived systems. Use *cpio*(C), instead.

rlogin

Remote login

Syntax

rlogin rhost [**-ec**] [**-8**] [**-L**] [**-l** username]

Description

Rlogin connects your terminal on the current local host system *lhost* to the remote host system *rhost*.

Each host has a file */etc/hosts.equiv* that contains a list of *rhost*'s with which it shares account names. (The host names must be the standard names as described in *rcmd*(TC).) When you **rlogin** as the same user on an equivalent host, you don't need to give a password. Each user may also have a private equivalence list in a file *rhosts* in his or her login directory. Each line in this file should contain an *rhost* and a *username* separated by a space, giving additional cases where logins without passwords are to be permitted. If the originating user is not equivalent to the remote user, then a login and password will be prompted for on the remote machine as in *login*(TC). To avoid some security problems, the .rhosts file must be owned by either the remote user or root.

The remote terminal type is the same as your local terminal type (as given in your environment TERM variable). The terminal or window size is also copied to the remote system if the server supports the option, and changes in size are reflected as well. All echoing takes place at the remote site, so that (except for delays) the rlogin is transparent. Flow control via ^S and ^Q and flushing of input and output on interrupts are handled properly.

The optional argument **-8** allows an eight-bit input data path at all times; otherwise parity bits are stripped except when the remote side's stop and start characters are other than ^S/^Q.

The argument **-L** allows the rlogin session to be run in without any output post-processing, (*e.g.* **stty -opost**.) A line of the form "~." disconnects from the remote host, where "~" is the escape character. A different escape character may be specified by the **-e** option. There is no space separating this option flag and the argument character.

Notes

The control character for closing **rlogin** connections (~ by default) does not appear until after you have typed in the expected character (.

by default).

When using **rlogin** to a 3.2 system, the login id is always requested, regardless of host equivalence;

See Also

netlogin (ADMN), rcmd(TC), rlogind(ADMN), rhosts(SFF).

Credit

This utility was developed at the University of California at Berkeley and is used with permission.

Files

/usr/hosts/* for *rhost* version of the command

Bugs

More of the environment should be propagated.

When using **rlogin** to a 3.2 system, the -l option is ignored.

ruptime

Show host status of local machines

Syntax

ruptime [**-a**] [**-r**] [**-l**] [**-t**] [**-u**]

Description

The *ruptime* command gives a status line for each machine on the local network; these are formed from packets broadcast by each host on the network, once every 1 - 3 minutes.

Machines for which no status report has been received for 5 minutes are shown as being down.

Users idle an hour or more are not counted unless the **-a** flag is given.

Normally, the listing is sorted by host name. The **-l** , **-t** , and **-u** flags specify sorting by load average, uptime, and number of users, respectively. The **-r** flag reverses the sort order.

Files

/usr/spool/rwho/whod.* data files

See Also

rwho(TC), rwhod(ADMN)

rwho

Who is logged in on local network

Syntax

rwho [**-a**]

Description

The *rwho* command lists users logged in on machines on the local network. The format is similar to that of *who*(C). Without options, only users who have typed in the last hour are listed. For each user listed, *rwho* displays the user name; the host name; and the date and time the user logged in. If the user has not typed in the last minute, *rwho* also displays the user's idle time in hours and minutes.

The *rwho* command understands the following option:

-a List all users on active nodes. (Users idle for more than an hour are listed.)

If information from a host is more than five minutes old, the host is assumed to be down and its users are not listed.

Requirements

Each host to be listed must be running the *rwhod*(ADMN) server, which broadcasts a status packet once every 1 - 3 minutes. The local host must also be running this server to maintain the data files. Since broadcasts do not cross gateways, hosts on other networks will not be listed.

Files

/usr/spool/rwho/whod.* information about other nodes

See Also

ruptime(TC), rwhod(ADMN).

talk

talk to another user

Syntax

talk person [ttyname]

Description

Talk is a visual communication program which copies lines from your terminal to that of another user.

If you wish to talk to someone on your own machine, then *person* is just the person's login name. If you wish to talk to a user on another host, then *person* is of the form :

host!user or
host.user or
host:user or
user@host

though *user@host* is perhaps preferred.

If you want to talk to a user who is logged in more than once, the *ttyname* argument may be used to indicate the appropriate terminal name.

When first called, it sends the message

Message from TalkDaemon@his_machine...
talk: connection requested by your_name@your_machine.
talk: respond with: talk your_name@your_machine

to the user you wish to talk to. At this point, the recipient of the message should reply by typing

talk your_name@your_machine

It doesn't matter from which machine the recipient replies, as long as his login-name is the same. Once communication is established, the two parties may type simultaneously, with their output appearing in separate windows. Typing control L will cause the screen to be reprinted, while your erase and kill characters will work in talk as normal. In addition, control-W is defined as a word-kill character. To exit, just type your interrupt character; *talk* then moves the cursor to the bottom of the screen and restores the terminal.

Permission to talk may be denied or granted by use of the *mesg*(TC) command. At the outset talking is allowed. Certain commands, in particular *nroff*(TC) and *pr*(TC) disallow messages in order to prevent messy output.

Files

/etc/hosts	to find the recipient's machine
/etc/utmp	to find the recipient's tty

See Also

mesg(TC), who(TC), mail(TC), write(TC), talkd(ADMN).

Bugs

The version of *talk*(TC) released with System V STREAMS TCP uses a protocol that is incompatible with the protocol used in the version released with 4.2BSD. The new protocol is compatible with 4.3BSD. The older protocol was not portable across different machine architectures.

Talk may be confused if you attempt to use the host.user format with a fully qualified hostname.

telnet

User interface to the TELNET protocol

Syntax

telnet [host [port]]

Description

The *telnet* command is used to communicate with another host using the **TELNET** protocol. If *telnet* is invoked without arguments, it enters command mode, indicated by its prompt (telnet>). In this mode, it accepts and executes the commands listed below. If it is invoked with arguments, it performs an **open** command with those arguments. (See below.)

Once a connection has been opened, *telnet* enters an input mode. The input mode entered will be either character-at-a-time or line-by-line, depending on what the remote system supports.

In character-at-a-time-mode, most text typed is immediately sent to the remote host for processing.

In line-by-line mode, all text is echoed locally, and (normally) only completed lines are sent to the remote host. The local echo character (initially ^E) may be used to turn the local echo off and on. (This would mostly be used to enter passwords without the passwords being echoed.)

In either mode, if the *localchars* toggle is TRUE (the default in line mode, discussed below), the user's *quit*, *intr*, and *flush* characters are trapped locally, and sent as **TELNET** protocol sequences to the remote side. There are options (**toggle** *autoflush* and **toggle** *autosynch* described below) which cause this action to flush subsequent output to the terminal (until the remote host acknowledges the **TELNET** sequence) and flush previous terminal input (in the case of *quit* and *intr*).

While connected to a remote host, *telnet* command mode may be entered by typing the *telnet* escape character (initially ^]). When in command mode, the normal terminal editing conventions are available.

COMMANDS

The following commands are available. Only enough of each command to uniquely identify it need be typed. (This is also true for arguments to the **mode**, **set**, **toggle**, and **display** commands.)

open *host* [*port*]
Open a connection to the named host. If no port number is specified, *telnet* will attempt to contact a **TELNET** server at the default port. The host specification may be either a host name (such as *hosts*(ADMN)) or an Internet address specified in the dot notation. (See *inet*(SLIB).)

close
Close a **TELNET** session and return to command mode. (This is virtually identical to **quit**.)

quit
Close any open **TELNET** session and exit *telnet*. An end-of-file (in command mode) will also close a session and exit.

z
Suspend *telnet*. On System V systems, this command provides the user with an escape to a shell running on the local machine.

mode *type*
Type is either *line* (for line-by-line mode) or *character* (for character-at-a-time mode). The remote host is asked for permission to go into the requested mode. If the remote host is capable of entering that mode, the requested mode will be entered.

status
Show the current status of *telnet*. This includes the peer to which one is connected, as well as the current mode. In addition, both the local and the remote **TELNET** options in effect are shown.

display [*argument...*]
Displays all, or some, of the **set** and **toggle** values. (See below.)

? [*command*]
Get help. With no arguments, *telnet* prints a help summary. If a command is specified, *telnet* will print the help information for just that command.

send *arguments*
Sends one or more special character sequences to the remote host. The following are the arguments which may be specified. (More than one argument may be specified at a time.)

escape
Sends the current *telnet* escape character (initially ^]).

synch
Sends the **TELNET SYNCH** sequence. This sequence causes the remote system to discard all previously typed (but not yet read) input. This sequence is sent as TCP urgent data. (It may not work if the remote system is a 4.2 BSD system. If it doesn't work, a lowercase **r** may be echoed on the terminal.)

brk
Sends the **TELNET BRK** (Break) sequence, which may have significance to the remote system.

ip
Sends the **TELNET IP** (Interrupt Process) sequence, which should cause the remote system to abort the currently-running process.

ao
Sends the **TELNET AO** (Abort Output) sequence, which should cause the remote system to flush all output **from** the remote system **to** the user's terminal.

ayt
Sends the **TELNET AYT** (Are You There) sequence, to which the remote system may or may not choose to respond.

ec
Sends the **TELNET EC** (Erase Character) sequence, which should cause the remote system to erase the last character entered.

el
Sends the **TELNET EL** (Erase Line) sequence, which should cause the remote system to erase the line currently being entered.

ga
Sends the **TELNET GA** (Go Ahead) sequence, which likely has no significance to the remote system.

nop
Sends the **TELNET NOP** (No OPeration) sequence.

?
Prints out help information for the **send** command.

set *argument value*
Set any one of a number of *telnet* variables to a specific value. The special value **off** turns off the function associated with the variable. The values of variables may be interrogated with the **display** command. The variables which may be specified are:

echo
This is the value (initially ^E) which, when in line-by-line mode, toggles between doing local echoing of entered characters (for normal processing), and suppressing echoing of entered characters (for entering, say, a password).

escape
This is the *telnet* escape character (initially ^[) which causes entry into *telnet* command mode (when connected to a remote system).

interrupt
If *telnet* is in *localchars* mode (discussed below) and the *interrupt* character is typed, a **TELNET IP** sequence (**send** *ip*, discussed above) is sent to the remote host. The initial value for the interrupt character is taken to be the terminal's **intr** character.

quit
If *telnet* is in *localchars* mode (discussed below) and the *quit* character is typed, a **TELNET BRK** sequence (**send** *brk*, discussed above) is sent to the remote host. The initial value for the quit character is taken to be the terminal's **quit** character.

flushoutput
If *telnet* is in *localchars* mode (discussed below) and the *flushoutput* character is typed, a **TELNET AO** sequence (**send** *ao*, discussed above) is sent to the remote host. The initial value for the flush character is taken to be the terminal's **flush** character.

erase
If *telnet* is in *localchars* mode (discussed below), **and** if *telnet* is operating in character-at-a-time mode, then when this character is typed, a **TELNET EC** sequence (**send** *ec*, discussed above) is sent to the remote system. The initial value for the erase character is taken to be the terminal's **erase** character.

kill
If *telnet* is in *localchars* mode (discussed below), **and** if *telnet* is operating in character-at-a-time mode, then when this character is typed, a **TELNET EL** sequence (**send** *el*, discussed above) is sent to the remote system. The initial value for the kill character is taken to be the terminal's **kill** character.

eof
If *telnet* is operating in line-by-line mode, entering this character as the first character on a line will cause the character to be sent to the remote system. The initial value of the eof character is taken to be the terminal's **eof** character.

toggle *arguments...*
Toggle (between TRUE and FALSE) various flags that control how *telnet* responds to events. More than one argument may be specified. The state of these flags may be interrogated with the **display** command. Valid arguments are:

localchars
If this is TRUE, then the *flush*, *interrupt*, *quit*, *erase*, and *kill* characters (discussed under **set**, above) are recognized locally, and transformed into (hopefully) appropriate **TELNET** control sequences (respectively, *ao*, *ip*, *brk*, *ec*, and *el*; see **send** above). The initial value for this toggle is TRUE in line-by-line mode, and FALSE in character-at-a-time mode.

autoflush
If *autoflush* and *localchars* are both TRUE, then when the *ao*, *intr*, or *quit* character is recognized (and transformed into **TELNET** sequences; detailed under **set** above), *telnet* refuses to display any data on the user's terminal until the remote system acknowledges (via a **TELNET** *Timing Mark* option) that it has processed those **TELNET** sequences. The initial value for this toggle is TRUE if the terminal user had not done an stty noflsh, otherwise FALSE. (See *stty*(C).)

autosynch
If *autosynch* and *localchars* are both TRUE, then when either the *intr* or *quit* characters (described above) is typed, the **TELNET** sequence sent is followed by the **TELNET SYNCH** sequence. This procedure **should** cause the remote system to begin throwing away all previously typed input until both of the **TELNET** sequences have been read and acted upon. The initial value of this toggle is FALSE.

crmod
Toggle carriage return mode. When this mode is enabled, most carriage return characters received from the remote host will be mapped into a carriage return followed by a line feed. This mode does not affect those characters typed by the user, only those received from the remote host. This mode is not very useful unless the remote host only sends carriage return, but never line feed. The initial value for this toggle is FALSE.

debug
Toggles socket-level debugging (useful only to the super user). The initial value for this toggle is FALSE.

options
Toggles the display of some internal *telnet* protocol processing (having to do with **TELNET** options). The initial value for this toggle is FALSE.

netdata
Toggles the display of all network data (in hexadecimal format). The initial value for this toggle is FALSE.

?
Displays the legal **toggle** commands.

do *option*

dont *option*

will *option*

wont *option*

These commands allow the user to send the appropriate **TELNET** option sequence. If no option is specified, *telnet* will prompt for one.

Notes

There is no adequate way for dealing with flow control.

On some remote systems, echo has to be turned off manually when in line-by-line mode.

There is enough settable state to justify a *.telnetrc* file.

No capability for a *.telnetrc* file is provided.

In line-by-line mode, the terminal's *eof* character is only recognized (and sent to the remote system) when it is the first character on a line.

tftp

User interface to the DARPA TFTP protocol

Syntax

tftp [host [port]]

Description

The *tftp* command is the user interface to the DARPA standard Trivial File Transfer Protocol. The program allows a user to transfer files to and from a remote network site.

The client host with which *tftp* is to communicate may be specified on the command line. If this is done, *tftp* will immediately attempt to establish a connection to a TFTP server on that host. Otherwise, *tftp* will enter its command interpreter and await instructions from the user. When *tftp* is awaiting commands from the user, the prompt:

tftp>

appears. The following commands are recognized by *tftp*:

connect *host-name* [*port*]
Set the *host* (and, optionally, *port*) for transfers. Note that the TFTP protocol, unlike the FTP protocol, does not maintain connections between transfers; thus, the *connect* command does not actually create a connection, but merely remembers what host is to be used for transfers. You do not have to use the *connect* command; the remote host can be specified as part of the *get* or *put* command.

mode *transfer-mode*
Set the mode for transfers; *transfer-mode* may be one of *ascii* or *binary*. The default is *ascii*.

put *file*
put *localfile remotefile*
put *file1 file2 ... fileN remote-directory*
Put a file or set of files to the specified remote file or directory. The destination can be in one of two forms: a filename on the remote host, if the host has already been specified, or a string of the form *host:filename* to specify both a host and a filename at the same time. If the latter form is used, the hostname specified becomes the default for future transfers. If the remote-directory form is used, the remote host is assumed to be a *UNIX* machine. Note that the file or files must previously exist and be publicly writeable on the remote system for **put** to successfully transfer the desired file or files.

get *filename*
get *remotename localname*
get *file1 file2 ... fileN*
Get a file or set of files from the specified *sources*. *Source* can be in one of two forms: a filename on the remote host, if the host has already been specified; or a string of the form *host:filename* to specify both a host and filename at the same time. If the latter form is used, the last hostname specified becomes the default for future transfers.

quit
Exit *tftp*. An end-of-file also exits.

verbose
Toggle verbose mode.

trace
Toggle packet-tracing.

status
Show current status.

rexmt *retransmission-timeout*
Set the per-packet retransmission timeout, in seconds.

timeout *total-transmission-timeout*
Set the total transmission timeout, in seconds.

ascii
Shorthand for *mode ascii*

binary
Shorthand for *mode binary*

? [*command-name ...*]
Print help information.

Files

/etc/hosts

See Also

tftpd(ADMN).

Warnings

Because there is no user-login or validation within the *TFTP* protocol, the remote site will probably have some sort of file-access restrictions in place. The exact methods are specific to each site.