

GNU inetutils

GNU networking utilities
for version 1.6, 27 December 2008

Alain Magloire et al.

This manual documents version 1.6 of the GNU networking utilities.

Copyright © 1994, 1995, 1996, 2000, 2001, 2007, 2008 Free Software Foundation, Inc.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover Texts, and with no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

1 Introduction

This manual is a work in progress: many sections make no attempt to explain basic concepts in a way suitable for novices. Thus, if you are interested, please get involved in improving this manual. The entire GNU community will benefit.

Please report bugs to bug-inetutils@gnu.org. Remember to include the version number, machine architecture, input files, and any other information needed to reproduce the bug: your input, what you expected, what you got, and why it is wrong. Diffs are welcome, but please include a description of the problem as well, since this is sometimes difficult to infer.

The individual utilities were originally derived from the 4.4BSDLite2 distribution.

Many features were integrated from NetBSD, OpenBSD, FreeBSD and GNU/Linux, the merges were done by a group of dedicated hackers (in no particular order): Jeff Bailey, Marcus Brinkmann, Michael Vogt, Bernhard Rosenkraenzer, Kaveh R. Ghazi, NIIBE Yutaka, Nathan Neulinger, Jeff Smith, Dan Stromberg, David O'Shea, Frederic Goudal, Gerald Combs, Joachim Gabler, Marco D'Itri, Sergey Poznyakoff.

2 Common options

Certain options are available in all these programs. Rather than writing identical descriptions for each of the programs, they are described here. (In fact, every GNU program accepts (or should accept) these options.)

Many of these programs take arbitrary strings as arguments. In those cases, ‘`--help`’ and ‘`--version`’ are taken as these options only if there is one and exactly one command line argument.

‘`--help`’ Print a usage message listing all available options, then exit successfully.

‘`--version`’
 Print the version number, then exit successfully.

3 traceroute: Trace the route to a host.

Traceroute traces the route packets take to a host.

```
traceroute [option]... [host]

'-M method'
'--type=method'
    Use method ('icmp' or 'udp') for traceroute operations.

'-p port'
'--port=port'
    Use destination port (default: 33434).

'-q num'
'--tries=num'
    Send num probe packets per hop (default: 3).

'--resolve-hostnames'
    Resolve hostnames.
```

4 ftp: FTP client

`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.

```
ftp [option...] [host [port]]
```

'-d'

'--debug' Enables debugging.

'-g'

'--no-glob'
Disables file name globbing.

'-i'

'--no-prompt'
Turns off interactive prompting during multiple file transfers.

'-n'

'--no-login'
Restrains `ftp` from attempting *auto-login* upon initial connection. If *auto-login* is enabled, `ftp` will check the `.netrc` (see below) file 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 login name (default is the user identity on the local machine), and, if necessary, prompt for a password and an account with which to login.

'-t'

'--trace' Enable packet tracing.

'-p[*prompt*]

'--prompt[=*prompt*]'
Print a command-line prompt, even if not on a tty. If *prompt* is supplied, its value is used instead of the default `'ftp>'`. Notice, that the argument is optional. When specified, if short option form is used ('-p') the argument should follow the option letter immediately, *without any intervening white space characters*. If long option is used, the argument must be specified after an equals sign.

'-v'

'--verbose'
Be verbose.

4.1 Commands

The client host with which `ftp` is to communicate may be specified on the command line. If this is done, `ftp` will immediately attempt to establish a connection to an FTP server on that host; otherwise, it 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 to be 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 required by a remote system for access to resources once a login has been successfully completed. If no argument is included, the user will be 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 that a bell be sounded after each file transfer command is completed.
- binary** Set the file transfer type to support binary image transfer.
- bye**
- quit** 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 upper case are written in the local directory with the letters mapped to lower case.
- 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.
- chmod** *mode file-name*
Change the permission modes of the file *file-name* on the remote system to *mode*.
- close**
- disconnect**
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 with 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, placing the output in *local-file*. If interactive prompting is on, `ftp` will prompt the user to verify that the last argument is indeed the target local file for receiving dir output. 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.

`form format`

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

`get remote-file [local-file]`

`recv 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 `csh(1)`. 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 by doing `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(1)` archive of the subtree (in binary mode).

`hash [size]`

Toggle hash-sign ('#') printing for each data block transferred. The size of a data block can optionally be specified. If not given, it defaults to 1024 bytes.

`help [command]`

? [command]

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

`idle [seconds]`

Set the inactivity timer on the remote server to *seconds* seconds. If *seconds* is omitted, the current inactivity timer is printed.

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 a listing of the contents of a directory on the remote machine. The listing includes any system-dependent information that the server chooses to include; for example, most UNIX systems will produce output from the command `ls -l`. (See also `nlist`.) If *remote-directory* is left unspecified, the current working directory is used. If interactive prompting is on, `ftp` will prompt the user to verify that the last argument is indeed the target local file for receiving ls output. 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 (consecutive newline characters in a file or carriage returns from the terminal) terminates macro input mode. There is a limit 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' signals that 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 [\[glob\]](#), page 6, 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 `lcd directory`; new local directories can be created with `! mkdir directory`.

mkdir *directory-name*

Make a directory on the remote machine.

mls *remote-files local-file*

Like `nlist`, except multiple remote files may be specified, and the *local-file* must 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`’.

`modtime` *file-name*

Show the last modification time of the file on the remote machine.

`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 [\[glob\]](#), page 6, for details of filename expansion. Resulting file names will then be processed according to `ntrans` and `nmap` settings.

`newer` *file-name*

Get the file only if the modification time of the remote file is more recent than the file on the current system. If the file does not exist on the current system, the remote file is considered newer. Otherwise, this command is identical to `get`.

`nlist` [*remote-directory*] [*local-file*]

Print a list of the files in a directory on the remote machine. If *remote-directory* is left unspecified, the current working directory is used. If interactive prompting is on, `ftp` will prompt the user to verify that the last argument is indeed the target local file for receiving `nlist` output. If no local file is specified, or if *local-file* is ‘-’, the output is sent to the terminal.

`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 `mput` commands and `put` commands issued without a specified remote target filename. If arguments are specified, local filenames are mapped during `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. The mapping follows the pattern set by *inpattern* and *outpattern*. *Inpattern* is a template for incoming filenames (which may have already 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* if *seq1* is not a null string; otherwise 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'. 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 `mput` commands and `put` commands issued without a specified remote target filename. If arguments are specified, characters in local filenames are translated during `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.

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 `autologin` option is on (default), `ftp` will also attempt to automatically log the user in to the FTP server (see below).

passive Toggle passive mode. If passive mode is turned on (default is off), the `ftp` client will send a `PASV` command for all data connections instead of the usual `PORT` command. The `PASV` command requests that the remote server open a port for the data connection and return the address of that port. The remote server listens on that port and the client connects to it. When using the more traditional `PORT` command, the client listens on a port and sends that address to the remote server, who connects back to it. Passive mode is useful when using `ftp` through a gateway router or host that controls the directionality of traffic. (Note that though `ftp` servers are required to support the `PASV` command by RFC 1123, some do not.)

prompt Toggle interactive prompting. Interactive prompting occurs during multiple file transfers to allow the user to selectively retrieve or store files. 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]`

`send local-file [remote-file]`

Store a local file on the remote machine. If *remote-file* is left unspecified, the local file name is used after processing according to any `ntrans` or `nmap` settings in naming the remote file. 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.

`quote arg...`

The arguments specified are sent, verbatim, to the remote FTP server.

`reget remote-file [local-file]`

`Reget` acts like `get`, except that if *local-file* exists and is smaller than *remote-file*, *local-file* is presumed to be a partially transferred copy of *remote-file* and the transfer is continued from the apparent point of failure. This command is useful when transferring very large files over networks that are prone to dropping connections.

`remotehelp [command-name]`

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

`remotestatus [file-name]`

With no arguments, show status of remote machine. If *filename* is specified, show status of *file-name* on remote machine.

`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.

`restart marker`

Restart the immediately following `get` or `put` at the indicated marker. On UNIX systems, *marker* is usually a byte offset into the file.

`rmdir directory-name`

Delete a directory on the remote machine.

`runique` Toggle storing 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 (see [\[ftp-shell\]](#), page 5). The default value is off.

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 which do ignore PORT commands but, incorrectly, indicate they've been accepted.

site arg...

The arguments specified are sent, verbatim, to the remote FTP server as a SITE command.

size file-name

Return size of *file-name* on remote machine.

status Show the current status of `ftp`.

struct [struct-name]

Set the file transfer structure to *struct-name*. By default 'stream' structure is used.

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

system Show the type of operating system running on the remote machine.

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.

umask [newmask]

Set the default umask on the remote server to *newmask*. If *newmask* is omitted, the current umask is printed.

user user-name [password] [account]

Identify yourself to the remote FTP server. If the password is not specified and the server requires it, `ftp` will prompt the user for it (after disabling local echo). If an account field is not specified, and the FTP server requires it, the user will be prompted for it. If an account field is specified, an account command will be relayed to the remote server after the login 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.

Command arguments which have embedded spaces may be quoted with quote ‘”’ marks.

4.2 Aborting A File Transfer

To abort a file transfer, use the terminal interrupt key (usually **C-C**). Sending transfers will be immediately halted. Receiving transfers will be halted by sending a 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 completed 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.

4.3 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. Ftp then forks a shell, using popen(3) with the argument supplied, and reads (writes) from the stdout (stdin). If the shell command includes spaces, the argument must be quoted; e.g. “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 csh(1); c.f. the glob command (see [glob], page 6). If the ftp command expects a single local file (e.g. 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 a ntrans or nmap setting. The resulting filename may then be altered by the remote server if sunique is on.

4.4 File Transfer Parameters

The FTP specification specifies many parameters which 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). Ftp supports the ‘ascii’ and ‘image’ types of file transfer, plus local byte size 8 for tenex mode transfers.

Ftp supports only the default values for the remaining file transfer parameters: mode, form, and struct.

4.5 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 machine 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 or a default token is encountered.

`'default'` This is the same as machine name except that default matches any name. There can be only one default token, and it must be after all machine tokens. This is normally used as:

```
default login anonymous password user@site
```

thereby giving the user automatic anonymous ftp login to machines not specified in `.netrc`. This can be overridden by using the `-n` flag to disable auto-login.

`'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 if the remote server requires a password as part of the login process. Note that if this token is present in the `.netrc` file for any user other than anonymous, `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 if the remote server requires an additional account password, or the auto-login process will initiate an `ACCT` command if it does not.

`'macdef name'`

Define a macro. This token functions like the `ftp macdef` command functions. 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.

Ftp utilizes the following environment variables.

`HOME` For default location of a `.netrc` file, if one exists.

`SHELL` For default shell.

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 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.

5 ftpd: FTP daemon

`ftpd` is the Internet File Transfer Protocol server process. The server uses the TCP protocol and listens at the port specified in the ‘`ftp`’ service specification; see [Section “Internet network services list” in *services\(5\) man page*](#).

```
ftpd [option]...
```

```
‘-A’
```

```
‘--anonymous-only’
```

Only anonymous login is allowed.

```
‘-a auth’
```

```
‘--auth=auth’
```

Specify what authentication mechanism to use for incoming connections. Possible values are: ‘`kerberos`’, ‘`kerberos5`’, ‘`opie`’ and ‘`default`’.

Anonymous logins will continue to work when this option is used.

```
‘-D’
```

```
‘--daemon’
```

`ftpd` enters daemon-mode. That allows `ftpd` to be run without `inetd`.

```
‘-d’
```

```
‘--debug’
```

Debugging information is written to the `syslog` using facility ‘`LOG_FTP`’.

```
‘-l’
```

```
‘--logging’
```

Each successful and failed ftp session is logged using `syslog` with a facility of ‘`LOG_FTP`’. If this option is specified twice, the retrieve (`get`), store (`put`), append, delete, make directory, remove directory and rename operations and their filename arguments are also logged.

```
‘-p pidfile’
```

```
‘--pidfile=pidfile’
```

Change default location of *pidfile*

```
‘-q’
```

```
‘--no-version’
```

Quiet mode. No information about the version of the `ftpd` is given to the client.

```
‘-T’
```

```
‘--max-timeout’
```

A client may also request a different timeout period; the maximum period allowed may be set to timeout seconds with the ‘`-T`’ option. The default limit is 2 hours.

```
‘-t timeout’
```

```
‘--timeout=timeout’
```

The inactivity timeout period is set to timeout seconds (the default is 15 minutes).

`'-u umask'`

`'--umask=umask'`

Set default umask(base 8).

The file `‘/etc/nologin’` can be used to disable ftp access. If the file exists, `ftpd` displays it and exits. If the file `‘/etc/ftpwelcome’` exists, `ftpd` prints it before issuing the `‘ready’` message. If the file `/etc/motd` exists, `ftpd` prints it after a successful login.

The FTP server currently supports the following FTP requests. The case of the requests is ignored.

Request	Description
ABOR	abort previous command
ACCT	specify account (ignored)
ALLO	allocate storage (vacuously)
APPE	append to a file
CDUP	change to parent of current working directory
CWD	change working directory
DELE	delete a file
HELP	give help information
LIST	give list files in a directory (“ls -lgA”)
MKD	make a directory
MDTM	show last modification time of file
MODE	specify data transfer mode
NLST	give name list of files in directory
NOOP	do nothing
PASS	specify password
PASV	prepare for server-to-server transfer
PORT	specify data connection port
PWD	print the current working directory
QUIT	terminate session
REST	restart incomplete transfer
RETR	retrieve a file
RMD	remove a directory
RNFR	specify rename-from file name
RNTO	specify rename-to file name
SITE	non-standard commands (see next section)
SIZE	return size of file
STAT	return status of server
STOR	store a file
STOU	store a file with a unique name
STRU	specify data transfer structure
SYST	show operating system type of server system
TYPE	specify data transfer type
USER	specify user name
XCUP	change to parent of current working directory (deprecated)
XCWD	change working directory (deprecated)
XMKD	make a directory (deprecated)

<code>XPWD</code>	print the current working directory (deprecated)
<code>XRMD</code>	remove a directory (deprecated)

The following non-standard or UNIX specific commands are supported by the `SITE` request.

Request	Description
<code>UMASK</code>	change umask, e.g. <code>SITE UMASK 002</code>
<code>IDLE</code>	set idle-timer, e.g. <code>SITE IDLE 60</code>
<code>CHMOD</code>	change mode of a file, e.g. <code>SITE CHMOD0 0CHMOD1 1CHMOD2</code>
<code>HELP</code>	give help information.

The remaining FTP requests specified in Internet RFC 959 are recognized, but not implemented. `MDTM` and `SIZE` are not specified in RFC 959, but will appear in the next updated FTP RFC.

The `ftp` server will abort an active file transfer only when the `ABOR` command is preceded by a Telnet ‘Interrupt Process’ (IP) signal and a Telnet ‘Synch’ signal in the command Telnet stream, as described in Internet RFC 959. If a `STAT` command is received during a data transfer, preceded by a Telnet IP and Synch, transfer status will be returned.

`Ftpd` interprets file names according to the globbing conventions used by `csh(1)`. This allows users to utilize the metacharacters ‘*?[]{}~’.

`Ftpd` authenticates users according to three rules.

1. The login name must be in the password data base, ‘`/etc/passwd`’, and not have a null password. In this case a password must be provided by the client before any file operations may be performed.
2. The login name must not appear in the file ‘`/etc/ftpusers`’.
3. The user must have a standard shell returned by `getusershell(3)`.
4. If the user name is ‘`anonymous`’ or ‘`ftp`’, an anonymous `ftp` account must be present in the password file (user ‘`ftp`’). In this case the user is allowed to log in by specifying any password (by convention an email address for the user should be used as the password).

In the last case, `ftpd` takes special measures to restrict the client’s access privileges. The server performs a `chroot(2)` to the home directory of the ‘`ftp`’ user. In order that system security is not breached, it is recommended that the ‘`ftp`’ subtree be constructed with care, following these rules:

‘`~ftp`’ Make the home directory owned by ‘`root`’ and unwritable by anyone.

‘`~ftp/bin`’
 Make this directory owned by ‘`root`’ and unwritable by anyone (mode 555). The program `ls` must be present to support the list command. This program should be mode 111.

‘`~ftp/etc`’
 Make this directory owned by ‘`root`’ and unwritable by anyone (mode 555). The files ‘`passwd`’ and ‘`group`’ must be present for the `ls` command to be able to produce owner names rather than numbers. The password field in ‘`passwd`’ is not used, and should not contain real passwords. The file ‘`motd`’, if present, will be printed after a successful login. These files should be mode 444.

‘~ftp/pub’

Make this directory mode 777 and owned by ‘ftp’. Guests can then place files which are to be accessible via the anonymous account in this directory.

5.1 Configuration files

‘/etc/ftpusers’

List of unwelcome/restricted users.

‘/etc/ftpwelcome’

Welcome notice.

‘/etc/motd’

Welcome notice after login.

‘/etc/nologin’

Displayed and access refused.

6 inetd

Inetd program should be run at boot time by `/etc/rc` (see `rc(8)`). It then listens for connections on certain internet sockets. When a connection is found on one of its sockets, it decides what service the socket corresponds to, and invokes a program to service the request. The server program is invoked with the service socket as its standard input, output and error descriptors. After the program is finished, inetd continues to listen on the socket (except in some cases which will be described below). Essentially, inetd allows running one daemon to invoke several others, reducing load on the system.

There are two types of services that inetd can start: standard and TCPMUX. A standard service has a well-known port assigned to it; it may be a service that implements an official Internet standard or is a BSD-specific service. As described in RFC 1078, TCPMUX services are nonstandard services that do not have a well-known port assigned to them. They are invoked from inetd when a program connects to the “tcpmux” well-known port and specifies the service name. This feature is useful for adding locally-developed servers.

6.1 Invocation

Normally, `inetd` is invoked without any arguments. It does, however, support several command line options. These are:

- `‘-d’`
- `‘--debug’` Turns on debugging. With this option, `inetd` stays in foreground and prints additional debugging information of `stderr`.
- `‘-R RATE’`
- `‘--rate=RATE’`
Specifies the maximum number of times a service can be invoked in one minute; the default is 1000.
- `‘--environment’`
Pass local and remote socket information in environment variables. See [Section 6.5 \[Inetd Environment\]](#), page 22.
- `‘--resolve’`
Resolve IP addresses when setting environment variables. See [Section 6.5 \[Inetd Environment\]](#), page 22.
- `‘-?’`
- `‘--help’` Display a short option summary.
- `‘--usage’` Display a short usage message
- `‘-V’`
- `‘--version’`
Print program version

6.2 Configuration file

Upon execution, inetd reads its configuration information from a configuration pathnames on the command line, by default, `‘/etc/inetd.conf’` and `‘/etc/initd.d’`. If the configuration pathname is a directory, all the files in the directory are read like a configuration file.

All of the configuration files are read and merged. There must be an entry for each field in the configuration file, with entries for each field separated by a tab or a space. Comments are denoted by a “#” at the beginning of a line. There must be an entry for each field. The fields of the configuration file are summarized in the table below (optional parts are enclosed in square brackets):

[service node:]service name

The service-name entry is the name of a valid service in the file ‘/etc/services’. For “internal” services (see [Section 6.3 \[Built-in services\], page 21](#)), the service name must be the official name of the service (that is, the first entry in ‘/etc/services’), or a numeric representation thereof. For TCPMUX services, the value of the ‘service name’ field consists of the string ‘tcpmux’ followed by a slash and the locally-chosen service name (see [Section 6.4 \[TCPMUX\], page 22](#)).

Optional ‘service node’ prefix is allowed for internet services. When present, it supplies the local addresses `inetd` should use when listening for that service. ‘Service node’ consists of a comma-separated list of addresses. Both symbolic host names and numeric IP addresses are allowed. Symbolic hostnames are looked up in DNS service. If a hostname has multiple address mappings, `inetd` creates a socket to listen on each address.

To avoid repeating an address that occurs frequently, a line with a host address specifier and colon, but no further fields is allowed, e.g.:

```
127.0.0.1,192.168.0.5:
```

The address specifier from such a line is remembered and used for all further lines lacking an explicit host specifier. Such a default address remains in effect until another such line or end of the configuration is encountered, whichever occurs first.

A special hostname ‘*’ stands for `INADDR_ANY`. When used in a normal configuration line, it causes the default address specifier to be ignored for that line. When used in a default address specification, e.g.:

```
*:
```

it causes any previous default address specifier to be forgotten.

socket type

The socket type should be one of ‘stream’, ‘dgram’, ‘raw’, ‘rdm’, or ‘seqpacket’, depending on whether the socket is a stream, datagram, raw, reliably delivered message, or sequenced packet socket. TCPMUX services must use ‘stream’.

protocol

The protocol must be a valid protocol as given in ‘/etc/protocols’. Examples might be ‘tcp’ or ‘udp’. TCPMUX services must use ‘tcp’. If IPv6 support is enabled the sockets will accept both IPv4 and IPv6 connections if that is supported by the OS. If `inetd` should only accept IPv4 or IPv6 connections, add ‘4’ or ‘6’ to the protocol name. For example ‘tcp4’ will only accept IPv4 tcp connections and ‘udp6’ will only accept IPv6 udp connections.

wait/nowait[.max]

The ‘wait/nowait’ entry specifies whether the server that is invoked by `inetd` will take over the socket associated with the service access point, and thus

whether `inetd` should wait for the server to exit before listening for new service requests. Datagram servers must use `'wait'`, as they are always invoked with the original datagram socket bound to the specified service address. These servers must read at least one datagram from the socket before exiting. If a datagram server connects to its peer, freeing the socket so `inetd` can receive further messages on the socket, it is said to be a “multi-threaded” server; it should read one datagram from the socket and create a new socket connected to the peer. It should fork, and the parent should then exit to allow `inetd` to check for new service requests to spawn new servers. Datagram servers which process all incoming datagrams on a socket and eventually time out are said to be “single-threaded”. `Comsat` (see [\(undefined\) \[biff\], page \(undefined\)](#)) and `talkd` (see [\(undefined\) \[talkd\], page \(undefined\)](#)) are both examples of the latter type of datagram server. `Tftpd` (see [\(undefined\) \[tftpd\], page \(undefined\)](#)) is an example of a multi-threaded datagram server.

Servers using stream sockets generally are multi-threaded and use the `'nowait'` entry. Connection requests for these services are accepted by `inetd`, and the server is given only the newly-accepted socket connected to a client of the service. Most stream-based services and all TCPMUX services operate in this manner. For such services, the number of running instances of the server can be limited by specifying optional `'max'` suffix (a decimal number), e.g.: `'nowait.15'`.

Stream-based servers that use `'wait'` are started with the listening service socket, and must accept at least one connection request before exiting. Such a server would normally accept and process incoming connection requests until a timeout. services must use `'nowait'`.

`user` The user entry should contain the user name of the user as whom the server should run. This allows for servers to be given less permission than root.

`server program`

The `server-program` entry should contain the pathname of the program which is to be executed by `inetd` when a request is found on its socket. If `inetd` provides this service internally, this entry should be `'internal'`.

It is common usage to specify `'/usr/sbin/tcpd'` in this field.

`server program arguments`

The server program arguments should be just as arguments normally are, starting with `argv[0]`, which is the name of the program. If the service is provided internally, this entry must contain the word `'internal'`, or be empty.

6.3 Built-in services

The `inetd` program provides several “trivial” services internally by use of routines within itself. All these services can operate both in `'stream'` and in `'dgram'` mode. They are:

`echo` Send back to the originating source any data received from it. This is a debugging and measurement tool, defined in [RFC 862 \(STD0020\)](#).

`discard` Silently throw away any data received. See [RFC 863 \(STD0021\)](#).

- chargen** This is a character generator service, defined in [RFC 864 \(STD0022\)](#). It can be operated as both stream or dgram service. When operating in ‘stream’ mode, once a connection is established a stream of data is sent out the connection (and any data received is thrown away). This continues until the calling user terminates the connection. When operating in ‘dgram’ mode, `inetd` listens for UDP datagrams, and for each received datagram, answers with a datagram containing a random number (between 0 and 512) of characters. Any data in the received datagram are ignored.
- daytime** Send back the current date and time in a human readable form. Any input is discarded.
See [RFC 867 \(STD0025\)](#)
- time** Send back the current date and time as a 32-bit integer number, representing the number of seconds since midnight, January 1, 1900.
See [RFC 868 \(STD0026\)](#)

6.4 TCPMUX

The TCPMUX protocol is defined in [RFC 1078](#) as follows:

A TCP client connects to a foreign host on TCP port 1. It sends the service name followed by a carriage-return line-feed <CRLF>. The service name is never case sensitive. The server replies with a single character indicating positive (+) or negative (-) acknowledgment, immediately followed by an optional message of explanation, terminated with a <CRLF>. If the reply was positive, the selected protocol begins; otherwise the connection is closed.” The program is passed the TCP connection as file descriptors 0 and 1.

If the TCPMUX service name begins with a “+”, `inetd` returns the positive reply for the program. This allows you to invoke programs that use stdin/stdout without putting any special server code in them.

The special service name ‘help’ causes `inetd` to list TCPMUX services in ‘`inetd.conf`’.

To define TCPMUX services, the configuration file must contain a ‘`tcpmux internal`’ definition.

Here are several example service entries for the various types of services:

```
ftp          stream tcp  nowait root  /usr/libexec/ftpd      ftpd -l
ntalk       dgram  udp   wait  root  /usr/libexec/ntalkd    ntalkd
tcpmux      stream tcp  nowait root  internal
tcpmux/+date stream tcp  nowait guest /bin/date              date
tcpmux/phonebook stream tcp  nowait guest /usr/bin/phonebook     phonebook
```

6.5 Inetd Environment

If a connection is made with a streaming protocol (‘stream’) and if ‘`--environment`’ option has been given, `inetd` will set the following environment variables before starting the program:

PROTO Always ‘TCP’.

TCPLOCALIP

Local IP address of the interface which accepted the connection.

TCPLOCALPORT

Port number on which the TCP connection was established.

TCPREMOTEIP

IP address of the remote client.

TCPREMOTEPORT

Port number on the client side of the TCP connection.

Additionally, if given the `--remote` option, `inetd` sets the following environment variables:

TCPLOCALHOST

DNS name of `TCPLOCALIP`.

TCPREMOTEHOST

DNS name of `TCPREMOTEIP`.

6.6 Error Messages

The `inetd` server logs error messages using `syslog(3)`. Important error messages and their explanations are:

`'service/protocol server failing (looping), service terminated.'`

The number of requests for the specified service in the past minute exceeded the limit. The limit exists to prevent a broken program or a malicious user from swamping the system. This message may occur for several reasons:

1. there are lots of hosts requesting the service within a short time period,
2. a “broken” client program is requesting the service too frequently,
3. a malicious user is running a program to invoke the service in a “denial of service” attack,
4. the invoked service program has an error that causes clients to retry quickly.

Use the `-R` option, as described above, to change the rate limit. Once the limit is reached, the service will be reenabled automatically in 10 minutes.

`'service/protocol: No such user 'user', service ignored'`

`'service/protocol: getpwnam: user: No such user'`

No entry for user exists in the `passwd` file. The first message occurs when `inetd` (re)reads the configuration file. The second message occurs when the service is invoked.

`'service/protocol: No such user 'user', service ignored'`

`'service/protocol: getpwnam: user: No such user'`

No entry for user exists in the `passwd` file. The first message occurs when `inetd` (re)reads the configuration file. The second message occurs when the service is invoked.

`'service: can't set uid number'`

`'service: can't set gid number'`

The user or group ID for the entry's user is invalid.

7 ping

Ping uses the ICMP protocol's mandatory ECHO_REQUEST datagram to elicit an ICMP ECHO_RESPONSE from a host or gateway. ECHO_REQUEST datagrams (*pings*) have an IP and ICMP header, followed by a *struct timeval* and then an arbitrary number of *pad* bytes used to fill out the packet.

7.1 Invoking

The options are as follows:

- '--echo' Send ICMP_ECHO requests (default).
- '--address' Send ICMP_ADDRESS packets.
- '--timestamp' Send ICMP_TIMESTAMP packets.
- '--router' Send ICMP_ROUTERDISCOVERY packets.
- '-c n'
- '--count=n' Stop after sending (and receiving) *n* ECHO_RESPONSE packets.
- '-d'
- '-debug' Set the SO_DEBUG option on the socket being used.
- '-i n'
- '--interval=n' Wait *n* seconds between sending each packet. The default is to wait for one second between each packet. This option is incompatible with the '-f' option.
- '-n'
- '--numeric' Numeric output only. No attempt will be made to lookup symbolic names for host addresses.
- '-r'
- '--ignore-routing' Bypass the normal routing tables and send directly to a host on an attached network. If the host is not on a directly-attached network, an error is returned. This option can be used to ping a local host through an interface that has no route through it (e.g., after the interface was dropped by routed(8)).
- '-v'
- '--verbose' Verbose output. ICMP packets other than ECHO_RESPONSE that are received are listed.
- '-f'
- '--flood' Flood ping. Outputs packets as fast as they come back or one hundred times per second, whichever is more. For every ECHO_REQUEST sent a period '.'

is printed, while for every ECHO_REPLY received a backspace is printed. This provides a rapid display of how many packets are being dropped. Only the super-user may use this option. This can be very hard on a network and should be used with caution.

`'-l n'`

`'--preload=n'`

If *n* is specified, ping sends that many packets as fast as possible before falling into its normal mode of behavior.

`'-p pat'`

`'--pattern=pat'`

You may specify up to 16 pad bytes to fill out the packet you send. This is useful for diagnosing data-dependent problems in a network. For example, `'-p ff'` will cause the sent packet to be filled with all ones.

`'-q'`

`'--quiet'` Quiet output. Nothing is displayed except the summary lines at startup time and when finished.

`'-R'`

`'--route'` Record route. Includes the RECORD_ROUTE option in the ECHO_REQUEST packet and displays the route buffer on returned packets. Note that the IP header is only large enough for nine such routes. Many hosts ignore or discard this option.

`'-s N'`

`'--size=N'`

Specifies the number of data bytes to be sent. The default is 56, which translates into 64 ICMP data bytes when combined with the 8 bytes of ICMP header data.

When using ping for fault isolation, it should first be run on the local host, to verify that the local network interface is up and running. Then, hosts and gateways further and further away should be pinged. Round-trip times and packet loss statistics are computed. If duplicate packets are received, they are not included in the packet loss calculation, although the round trip time of these packets is used in calculating the minimum/average/maximum round-trip time numbers. When the specified number of packets have been sent (and received) or if the program is terminated with a `'SIGINT'`, a brief summary is displayed.

This program is intended for use in network testing, measurement and management. Because of the load it can impose on the network, it is unwise to use ping during normal operations or from automated scripts.

7.2 ICMP Packet Details

An IP header without options is 20 bytes. An ICMP ECHO_REQUEST packet contains an additional 8 bytes worth of ICMP header followed by an arbitrary amount of data. When a packet size is given, this indicates the size of this extra piece of data (the default is 56). Thus the amount of data received inside of an IP packet of type ICMP ECHO_REPLY will always be 8 bytes more than the requested data space (the ICMP header).

If the data space is at least eight bytes large, ping uses the first eight bytes of this space to include a timestamp which it uses in the computation of round trip times. If less than eight bytes of pad are specified, no round trip times are given.

7.3 Duplicate And Damaged Packets

Ping will report duplicate and damaged packets. Duplicate packets should never occur, and seem to be caused by inappropriate link-level retransmissions. Duplicates may occur in many situations and are rarely (if ever) a good sign, although the presence of low levels of duplicates may not always be cause for alarm.

Damaged packets are obviously serious cause for alarm and often indicate broken hardware somewhere in the ping packet's path (in the network or in the hosts).

7.4 Trying Different Data Patterns

The (inter)network layer should never treat packets differently depending on the data contained in the data portion. Unfortunately, data-dependent problems have been known to sneak into networks and remain undetected for long periods of time. In many cases the particular pattern that will have problems is something that doesn't have sufficient "transitions", such as all ones or all zeros, or a pattern right at the edge, such as almost all zeros. It isn't necessarily enough to specify a data pattern of all zeros (for example) on the command line because the pattern that is of interest is at the data link level, and the relationship between what you type and what the controllers transmit can be complicated.

This means that if you have a data-dependent problem you will probably have to do a lot of testing to find it. If you are lucky, you may manage to find a file that either can't be sent across your network or that takes much longer to transfer than other similar length files. You can then examine this file for repeated patterns that you can test using the '-p' option of ping.

7.5 TTL Details

The TTL value of an IP packet represents the maximum number of IP routers that the packet can go through before being thrown away. In current practice you can expect each router in the Internet to decrement the TTL field by exactly one.

The TCP/IP specification states that the TTL field for TCP packets should be set to 60, but many systems use smaller values (4.3 BSD uses 30, 4.2 used 15).

The maximum possible value of this field is 255, and most UNIX systems set the TTL field of ICMP ECHO_REQUEST packets to 255. This is why you will find you can ping some hosts, but not reach them with telnet(1) or ftp(1).

In normal operation ping prints the ttl value from the packet it receives. When a remote system receives a ping packet, it can do one of three things with the TTL field in its response:

- Not change it; this is what Berkeley UNIX systems did before the 4.3BSD-Tahoe release. In this case the TTL value in the received packet will be 255 minus the number of routers in the round-trip path.
- Set it to 255; this is what current Berkeley UNIX systems do. In this case the TTL value in the received packet will be 255 minus the number of routers in the path from the remote system to the pinging host.

- Set it to some other value. Some machines use the same value for ICMP packets that they use for TCP packets, for example either 30 or 60. Others may use completely wild values.

Many Hosts and Gateways ignore the RECORD_ROUTE option.

The maximum IP header length is too small for options like RECORD_ROUTE to be completely useful. There's not much that can be done about this, however.

Flood pinging is not recommended in general, and flood pinging the broadcast address should only be done under very controlled conditions.

8 rcp: Copy files between machines

Rcp copies files between machines. Each file or directory argument is either a remote file name of the form `'rname@rhost:path'`, or a local file name (containing no `'.'` characters, or a `'/'` before any `'.'`s).

```
rcp [option]... old-file new-file
rcp [option]... files... directory
```

`'-K'`

`'--kerberos'`

Turns off all Kerberos authentication.

`'-k'`

`'--realm=realm'`

The option requests rcp to obtain tickets for the remote host in *realm* realm instead of the remote host's realm as determined by `krb_realmofhost(3)`.

`'-p'`

`'--preserve'`

Causes rcp to attempt to preserve (duplicate) in its copies the modification times and modes of the source files, ignoring the umask. By default, the mode and owner of file are preserved if it already existed; otherwise the mode of the source file modified by the umask(2) on the destination host is used.

`'-r'`

`'--recursive'`

If any of the source files are directories, rcp copies each subtree rooted at that name; in this case the destination must be a directory.

`'-x'`

`'--encrypt'`

Turns on DES encryption for all data passed via the rcp session. This may impact response time and CPU utilization, but provides increased security.

Rcp doesn't detect all cases where the target of a copy might be a file in cases where only a directory should be legal.

Rcp can be confused by any output generated by commands in a `'login'`, `'profile'`, or `'cshrc'` file on the remote host.

The destination user and hostname may have to be specified as `'rhost.rname'` when the destination machine is running the 4.2BSD version of rcp.

9 rexecd: server for rexec

Rexecd is the server for the Rexec routine. The server provides remote execution facilities with authentication based on user names and passwords.

```
rexecd [option]...
```

Rexecd listens for service requests at the port indicated in the ‘exec’ service specification; see services(5). When a service request is received the following protocol is initiated:

1. The server reads characters from the socket up to a NUL (‘\0’) byte. The resultant string is interpreted as an ASCII number, base 10.
2. If the number received in step 1 is non-zero, it is interpreted as the port number of a secondary stream to be used for the stderr. A second connection is then created to the specified port on the client’s machine.
3. A NUL terminated user name of at most 16 characters is retrieved on the initial socket.
4. A NUL terminated, unencrypted password of at most 16 characters is retrieved on the initial socket.
5. A NUL terminated command to be passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper bound on the size of the system’s argument list.
6. Rexecd then validates the user as is done at login time and, if the authentication was successful, changes to the user’s home directory, and establishes the user and group protections of the user. If any of these steps fail the connection is aborted with a diagnostic message returned.
7. A NUL byte is returned on the initial socket and the command line is passed to the normal login shell of the user. The shell inherits the network connections established by rexecd.

9.1 Diagnostics

Except for the last one listed below, all diagnostic messages are returned on the initial socket, after which any network connections are closed. An error is indicated by a leading byte with a value of 1 (0 is returned in step 7 above upon successful completion of all the steps prior to the command execution).

```
‘username too long’
```

The name is longer than 16 characters.

```
‘password too long’
```

The password is longer than 16 characters.

```
‘command too long’
```

The command line passed exceeds the size of the argument list (as configured into the system).

```
‘Login incorrect.’
```

No password file entry for the user name existed.

```
‘Password incorrect.’
```

The wrong password was supplied.

'No remote directory.'

The chdir command to the home directory failed.

'Try again.'

A fork by the server failed.

'<shellname>: ...'

The user's login shell could not be started. This message is returned on the connection associated with the stderr, and is not

Note, that indicating 'Login incorrect' as opposed to 'Password incorrect' is a security breach which allows people to probe a system for users with null passwords.

10 rlogin

Rlogin command logs into a specified remote host and connects your local terminal to the remote host. The remote terminal type is the same as that given in the `TERM` local environment variable. The terminal or window size is also the same, if the remote host supports them, and any changes in size are transferred.

When using the `rlogin` command, you can create a link to your path using a host name as the link name. For example:

```
#
# ln -s /usr/bin/rlogin HostName
# Hostname -8
```

Using `HostName` automatically uses the `rlogin` to log in to the remote host named `HostName`.

Rlogin allows access to the remote host without the use of a passwd. For details, See [Section “rcmd” in *The GNU C Library Reference Manual*](#).

10.1 Invoking

The options are as follows :

‘-8’

‘--8-bit’ 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 `C-S/C-Q`.

‘-E’

‘--no-escape’

‘--no-escape’

Stops any character from being recognized as an escape character. When used with the ‘-8’ option, this provides a completely transparent connection.

‘-K’

‘--kerberos’

Turns off all Kerberos authentication.

‘-d’

‘--debug’ Turns on socket debugging (see [Section “set options on sockets” in *setsockopt\(2\) man page*](#)) on the TCP sockets used for communication with the remote host.

‘-e’

‘--escape=char’

Allows user specification of the escape character, which is ‘~’ by default. This specification may be as a literal character, or as an octal value in the form ‘\nnn’.

‘-k’

‘--realm=realm’

The option requests rlogin to obtain tickets for the remote host in *realm* realm instead of the remote host’s realm as determined by `krb_realmofhost(3)`.

'-x'

'--encrypt'

Turns on DES encryption for all data passed via the rlogin session. This may impact response time and CPU utilization, but provides increased security.

A line of the form *escape-char*. disconnects from the remote host. Similarly, the line *escape-charC-Z* will suspend the rlogin session, and *escape-chardelayed-suspend-char* suspends the send portion of the rlogin, but allows output from the remote system. By default, the tilde ('~) character' is the *escape-char*, and normally *C-Y* is the *delayed-suspend-char*.

All echoing takes place at the remote site, so that (except for delays) the rlogin is transparent. Flow control via *C-S/C-Q*, if supported, stop and start the flow of information, flushing of input and output on interrupts are handled properly.

On the server side the *iruserok* and *ruserok* functions are used to authenticate, see the appropriate man page for more information, if supported.

10.2 Kerberos Authentication

If rlogin was compiled with kerberos support, options '-x', '-k', '-K' are available. Each user may have a private authorization list in the file '.klogin' in their home directory. Each line in this file should contain a Kerberos principal name of the form 'principal.instance@realm'. If the originating user is authenticated to one of the principals named in '.klogin', access is granted to the account. The principal 'accountname.@localrealm' is granted access if there is no '.klogin' file. Otherwise a login and password will be prompted for on the remote machine as in login(1). To avoid certain security problems, the '.klogin' file must be owned by the remote user. If Kerberos authentication fails, a warning message is printed and the standard Berkeley rlogin is used instead.

11 rlogind

Rlogind is the server for the **rlogin** program (see [Chapter 10 \[rlogin\], page 31](#)). The server provides a remote login facility with authentication based on privileged port numbers from trusted hosts.

Rlogind listens for service requests at the port indicated in the ‘**login**’ service specification; see [Section “Internet network services list” in *services\(5\) man page*](#). When a service request is received the following protocol is initiated:

1. The server checks the client’s source port. If the port is not in the range 512-1023, the server aborts the connection.
2. The server checks the client’s source address and requests the corresponding host name (see [gethostbyaddr\(3\)](#), [hosts\(5\)](#) and [named\(8\)](#)). If the hostname cannot be determined, the dot-notation representation of the host address is used. If the hostname is in the same domain as the server (according to the last two components of the domain name), or if the ‘-a’ option is given, the addresses for the hostname are requested, verifying that the name and address correspond. Normal authentication is bypassed if the address verification fails.

Once the source port and address have been checked, **rlogind** proceeds with the authentication process described [Chapter 13 \[rshd\], page 37](#). It then allocates a pseudo terminal (see [pty\(4\)](#)), and manipulates file descriptors so that the slave half of the pseudo terminal becomes the stdin, stdout, and stderr for a login process. The login process is an instance of the **login** program, invoked with the ‘-f’ option if authentication has succeeded. If automatic authentication fails, the user is prompted to log in as if on a standard terminal line.

The parent of the login process manipulates the master side of the pseudo terminal, operating as an intermediary between the login process and the client instance of the **rlogin** program. In normal operation, the packet protocol described in [pty\(4\)](#) is invoked to provide *C-S/C-Q* type facilities and propagate interrupt signals to the remote programs. The login process propagates the client terminal’s baud rate and terminal type, as found in the environment variable, **TERM**. The screen or window size of the terminal is requested from the client, and window size changes from the client are propagated to the pseudo terminal.

Transport-level keepalive messages are enabled unless the ‘-n’ option is client instance of the **rlogin** program. The use of keepalive messages allows sessions to be timed out if the client crashes or becomes unreachable.

See [Section “ruserok” in *The GNU C Library Reference Manual*](#), for details.

11.1 Invoking

The options are as follows:

- ‘-a’
- ‘--verify-hostname’
 Ask hostname for verification.
- ‘-d’
- ‘--daemon’
 Daemon mode.

```

'-l'
'--no-rhosts'
    Ignore '.rhosts' file.

'-L name'
'--local-domain=name'
    Set local domain name.

'-n'
'--no-keepalive'
    Do not set SO_KEEPALIVE.

'-k'
'--kerberos'
    Use kerberos IV authentication.

'-x'
'--encrypt'
    Turns on DES encryption for all data passed via the rlogind session. This may
    impact response time and CPU utilization, but provides increased security.

'-D[level]'
'--debug[=level]'
    Set debug level, not implemented.

'-h'
'--help'    Display usage instructions.

'-V'
'--version'
    Display program version.

'-o'
'--allow-root'
    Allow uid == 0 to login, disabled by default.

'-p port'
'--port=port'
    Listen on given port (valid only in daemon mode).

'-r'
'--reverse-required'
    Required Require reverse resolving of a remote host IP.

```

11.2 Diagnostics

All initial diagnostic messages are indicated by a leading byte with a value of 1, after which any network connections are closed. If there are no errors before login is invoked, a null byte is returned as in indication of success.

```

'Try again.'
    A fork by the server failed.

```

The authentication procedure used here assumes the integrity of each client machine and the connecting medium. This is insecure, but is useful in an “open” environment.

12 rsh

Rsh executes command on host and copies its standard input to the remote command, the standard output of the remote command to its standard output, and the standard error of the remote command to its standard error. Interrupt, quit and terminate signals are propagated to the remote command; rsh normally terminates when the remote command does.

When using the rsh command, you can create a link to your path using a host name as the link name. For example:

```
#
# ln -s /usr/bin/rsh HostName
# Hostname ls
```

HostName will be passed to rsh as the default host.

Rsh allows access to the remote host without the use of a passwd. For details, See [Section “rcmd” in *The GNU C Library Reference Manual*](#).

12.1 Invoking

The options are as follows :

- ‘-K’
- ‘--kerberos’
Turns off all Kerberos authentication.
- ‘-d’
- ‘--debug’ Turns on socket debugging (see [Section “set options on sockets” in *setsockopt\(2\) man page*](#)) on the TCP sockets used for communication with the remote host.
- ‘-k *realm*’
- ‘--realm=*realm*’
The option requests rsh to obtain tickets for the remote host in *realm* realm instead of the remote host’s realm as determined by `krb_realmofhost(3)`.
- ‘-x’
- ‘--encrypt’
Turns on DES encryption for all data passed via the rsh session. This may impact response time and CPU utilization, but provides increased security.
- ‘-l’
- ‘--user’ By default, the remote username is the same as the local username. The ‘-l’ option or the ‘`username@host`’ format allow the remote name to be specified. Kerberos authentication is used, and authorization is determined as in `rlogin` (see [Chapter 10 \[rlogin\], page 31](#)).

If no command is specified, you will be logged in on the remote host using `rlogin`.

Shell metacharacters which are not quoted are interpreted on local machine, while quoted metacharacters are interpreted on the remote machine.

For example, the command

```
# rsh otherhost cat remotefile >> localfile
```

appends the remote file 'remotefile' to the local file 'localfile', while

```
# rsh otherhost cat remotefile ">>" other_remotefile
```

appends 'remotefile' to 'other_remotefile'.

13 rshd

The `rshd` server is the server for the `rcmd(3)` routine and, consequently, for the `rsh` (see [Chapter 12 \[rsh\], page 35](#)) program. The server provides remote execution facilities with authentication based on privileged port numbers from trusted hosts. The `rshd` server listens for service requests at the port indicated in the ‘`cmd`’ service specification; see [Section “Internet network services list” in *services\(5\) man page*](#).. When a service request is received the following protocol is initiated:

1. The server checks the client’s source port. If the port is not in the range 512–1023, the server aborts the connection.
2. The server reads characters from the socket up to a NUL (‘\0’) byte. The resultant string is interpreted as an ASCII number, base 10.
3. If the number received in step 2 is non-zero, it is interpreted as the port number of a secondary stream to be used for the `stderr`. A second connection is then created to the specified port on the client’s machine. The source port of this second connection is also in the range 512–1023.
4. The server checks the client’s source address and requests the corresponding host name (see `gethostbyaddr(3)`, `hosts(5)` and `named(8)`). If the hostname cannot be determined, the dot-notation representation of the host address is used. If the hostname is in the same domain as the server (according to the last two components of the domain name), or if the ‘`-a`’ option is given, the addresses for the hostname are requested, verifying that the name and address correspond. If address verification fails, the connection is aborted with the message, ‘`Host address mismatch.`’
5. A null terminated user name of at most 16 characters is retrieved on the initial socket. This user name is interpreted as the user identity on the client’s machine.
6. A null terminated user name of at most 16 characters is retrieved on the initial socket. This user name is interpreted as a user identity to use on the server’s machine.
7. A null terminated command to be passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper bound on the size of the system’s argument list.
8. `Rshd` then validates the user using `ruserok(3)`, which uses the file ‘`/etc/hosts.equiv`’ and the ‘`.rhosts`’ file found in the user’s home directory. The ‘`-l`’ option prevents `ruserok(3)` from doing any validation based on the user’s ‘`.rhosts`’ file, unless the user is the superuser.
9. If the file ‘`/etc/nologin`’ exists and the user is not the superuser, the connection is closed.
10. A null byte is returned on the initial socket and the command line is passed to the normal login shell of the user. The shell inherits the network connections established by `rshd`.
11. Transport-level keepalive messages are enabled unless the ‘`-n`’ option is present. The use of keepalive messages allows sessions to be timed out if the client crashes or becomes unreachable.
12. The ‘`-L`’ option causes all successful accesses to be logged to `syslogd` (see [Chapter 15 \[syslogd invocation\], page 41](#)) as ‘`auth.info`’ messages.

See [Section “ruserok” in *The GNU C Library Reference Manual*](#), for details.

13.1 Invoking

The options are as follows:

```

'-a'
'--verify-hostname'
    Ask hostname for verification.

'-l'
'--no-rhosts'
    Ignore '.rhosts' file.

'-L name'
'--local-domain=name'
    Set local domain name.

'-n'
'--no-keepalive'
    Do not set SO_KEEPALIVE.

'-k'
'--kerberos'
    Use kerberos IV authentication.

'-x'
'--encrypt'
    Turns on DES encryption for all data passed via the rshd session. This may
    impact response time and CPU utilization, but provides increased security.

'-D[level]'
'--debug[=level]'
    Set debug level, not implemented.

'-h'
'--help'    Display usage instructions.

'-V'
'--version'
    Display program version.

```

13.2 Diagnostics

Except for the last one listed below, all diagnostic messages are returned on the initial socket, after which any network connections are closed. An error is indicated by a leading byte with a value of 1 (0 is returned in step 10 above upon successful completion of all the steps prior to the execution of the login shell).

```

'Locuser too long'
    The name of the user on the client's machine is longer than 16 characters.

'Ruser too long'
    The name of the user on the remote machine is longer than 16 characters.

'Command too long'
    The command line passed exceeds the size of the argument list (as configured
    into the system).

```

'Login incorrect'

No password file entry for the user name existed.

'Remote directory'

The chdir command to the home directory failed.

'Permission denied'

The authentication procedure described above failed.

'Can't make pipe.'

The pipe needed for the stderr, wasn't created.

'Can't fork; try again.'

A fork by the server failed.

'<shellname>: ...'

The user's login shell could not be started. This message is returned on the connection associated with the stderr, and is not preceded by a flag byte.

The authentication procedure used here assumes the integrity of each client machine and the connecting medium. This is insecure, but is useful in an "open" environment.

14 `logger`: Make entries in the system log

`logger` is a program to make entries in the system log files. It provides a shell command interface to the system log module. See Section “Syslog” in *The GNU C Library Reference Manual*, for details.

The message can contain a priority code, which should be a preceding decimal number in angle braces, for example, `<5>`. This priority code should map into the priorities defined in the include file `<sys/syslog.h>`.

```
logger [option]... [message]
```

‘-i’ Log the process ID of the logger process with each line.

‘-s’ Log the message to standard error, as well as the system log. This option might not be supported on all systems.

‘-f *file*’

‘--file=*file*’

Log the content of the specified file.

‘-p *priority*’

‘--priority=*priority*’

Enter the message with the specified priority. The priority may be specified numerically or as a ‘*facility.level*’ pair. For example, ‘-p `local3.info`’ logs the message at the informational level in the ‘`local3`’ facility. The default is ‘`user.notice`’.

The actual list of supported facilities and levels is system specific.

‘-t *tag*’

‘--tag=*tag*’

Mark every line in the log with the specified tag.

The options are followed by the message which should be written to the log. If not specified, and the ‘-f’ flag is not provided, standard input is logged.

The logger utility exits 0 on success, and >0 if an error occurs.

The following examples illustrate the usage of the `logger` command:

```
logger System rebooted
```

```
logger -p local0.notice -t HOSTIDM -f /dev/idmc
```

15 syslogd: system service logging facility

Syslogd is a system service that provides error logging facility. Messages are read from the UNIX domain socket `‘/dev/log’`, from an Internet domain socket specified in `‘/etc/services’`, and from the special device `‘/dev/klog’` (to read kernel messages).

syslogd creates the file `‘/var/run/syslog.pid’`, and stores its process id there. This can be used to kill or reconfigure syslogd.

The message sent to syslogd should consist of a single line. The message can contain a priority code, which should be a preceding decimal number in angle braces, for example, `<5>`. This priority code should map into the priorities defined in the include file `sys/syslog.h`.

```
syslog [options]...
```

```
‘-f file’
```

```
‘--rcfile=file’
```

Override configuration (the default file is `‘/etc/syslog.conf’`).

```
‘--pidfile=file’
```

Override pidfile (the default file is `‘/var/run/syslogd.pid’`).

```
‘-n’
```

```
‘--no-detach’
```

Do not enter daemon mode.

```
‘-d’
```

```
‘--debug’ Print debug information (implies ‘-n’).
```

```
‘-p file’
```

```
‘--socket=file’
```

Override default UNIX domain socket `‘/dev/log’`.

```
‘-a socket’
```

Add UNIX socket to listen. An unlimited number of sockets is allowed.

```
‘-r’
```

```
‘--inet’ Receive remote messages via Internet domain socket.
```

```
‘--no-unixaf’
```

Do not listen on UNIX domain sockets (overrides `‘-a’` and `‘-p’`).

```
‘--no-klog’
```

Do not listen to the kernel log device `‘/dev/klog’`.

```
‘--no-forward’
```

Do not forward any messages (overrides `‘-h’`).

```
‘-h’
```

```
‘--hop’ Forward messages from remote hosts.
```

```
‘-m interval’
```

```
‘--mark=interval’
```

Specify timestamp interval in logs (0 for no timestamps).

`-l hostlist`

Log hosts in *hostlist* by their hostname. Multiple lists are allowed.

`-s domainlist`

List of domains which should be stripped from the FQDN of hosts before logging their name. Multiple lists are allowed.

15.1 Configuration file

Syslogd reads its configuration file when it starts up and whenever it receives a hangup signal. The `syslog.conf` file is the configuration file for the `syslogd` program. It consists of lines with two fields: the *selector* field which specifies the types of messages and priorities to which the line applies, and an *action* field which specifies the action to be taken if a message `syslogd` receives matches the selection criteria. The selector field is separated from the action field by one or more tab or space characters. A rule can be splitted in several lines if all lines except the last are terminated with a backslash `\`.

The Selectors function are encoded as a facility, a period (`.`), and a level, with no intervening white-space. Both the facility and the level are case insensitive.

The facility describes the part of the system generating the message, and is one of the following keywords: `auth`, `authpriv`, `cron`, `daemon`, `kern`, `lpr`, `mail`, `mark`, `news`, `syslog`, `user`, `uucp` and `local0` through `local7`. These keywords (with the exception of `mark`) correspond to the similar `LOG_` values specified to the `openlog` and `syslog` library routines. See [Section “Syslog” in *The GNU C Library Reference Manual*](#), for details.

The level describes the severity of the message, and is a keyword from the following ordered list (higher to lower): `emerg`, `alert`, `crit`, `err`, `warning`, `notice` and `debug`. These keywords correspond to the similar `LOG_` values specified to the `syslog` library routine.

See [Section “syslog and vsyslog” in *The GNU C Library Reference Manual*](#), for a further descriptions of both the facility and level keywords and their significance.

If a received message matches the specified facility and is of the specified level (or a higher level), the action specified in the action field will be taken.

Multiple selectors may be specified for a single action by separating them with semicolon (`;`) characters. It is important to note, however, that each selector can modify the ones preceding it.

Multiple facilities may be specified for a single level by separating them with comma (`,`) characters.

An asterisk (`*`) can be used to specify all facilities or all levels. Two asterisks (`**`) specify all facilities not named previously in the configuration file.

By default, a level applies to all messages with the same or higher level. The equal (`=`) character can be prepended to a level to restrict this line of the configuration file to messages with the very same level.

An exclamation mark (`!`) prepended to a level or the asterisk means that this line of the configuration file does not apply to the specified level (and higher ones). In conjunction with the equal sign, you can exclude single levels as well.

The special facility ‘mark’ receives a message at priority ‘info’ every 20 minutes. This is not enabled by a facility field containing an asterisk.

The special level ‘none’ disables a particular facility.

The action field of each line specifies the action to be taken when the selector field selects a message. There are five forms:

- A pathname (beginning with a leading slash). Selected messages are appended to the file.

You may prepend a minus (‘-’) to the path to omit syncing the file after each message log. This can cause data loss at system crashes, but increases performance for programs which use logging extensively.

- A named pipe (fifo), beginning with a vertical bar (‘|’) followed by a pathname. The pipe must be created with `mkfifo` before `syslogd` reads its configuration file. This feature is especially useful for debugging.
- A hostname (preceded by an at (‘@’) sign). Selected messages are forwarded to `syslogd` on the named host.
- A comma separated list of users. Selected messages are written to those users if they are logged in.
- An asterisk. Selected messages are written to all logged-in users.

Blank lines and lines whose first non-blank character is a hash (‘#’) character are ignored.

A configuration file might appear as follows:

```
# Log all kernel messages, authentication messages of
# level notice or higher and anything of level err or
# higher to the console.
# Don't log private authentication messages!
*.err;kern.*;auth.notice;authpriv.none /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none /var/log/messages

# The authpriv file has restricted access.
authpriv.* /var/log/secure

# Log all the mail messages in one place.
mail.* /var/log/maillog

# Everybody gets emergency messages, plus log them on another
# machine.
*.emerg *
*.emerg @arpa.berkeley.edu

# Root and Eric get alert and higher messages.
*.alert root,eric
```

```
# Save mail and news errors of level err and higher in a  
# special file.  
uucp,news.crit                               /var/log/spoolerr
```

The effects of multiple selectors are sometimes not intuitive. For example 'mail.crit,*err' will select the 'mail' facility messages at the level of 'err' or higher, not at the level of 'crit' or higher.

16 `talk`: a communication program

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

16.1 Invoking

The command line arguments are as follows:

- person* 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 '`user@host`'.
- ttyname* If you wish to talk to a user who is logged in more than once, the *ttyname* argument may be used to indicate the appropriate terminal name, where *ttyname* is of the form '`ttyXX`'.

When first called, `talk` 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 `C-L` will cause the screen to be reprinted, while your erase, kill, and word kill characters will behave normally. To exit, just type your interrupt character; `talk` then moves the cursor to the bottom of the screen and restores the terminal to its previous state.

Permission to talk may be denied or granted by use of the `mesg(1)` command. At the outset talking is allowed. Certain commands, in particular `nroff(1)` and `pr(1)`, disallow messages in order to prevent messy output.

To exit, just type your interrupt character; `talk` then moves the cursor to the bottom of the screen and restores the terminal to its previous state.

17 talkd: a server for communication between users

Talkd is the server that notifies a user that someone else wants to initiate a conversation. It acts as a repository of invitations, responding to requests by clients wishing to rendezvous to hold a conversation.

```
talkd [option]...  
  
'-a file'  
'--acl=file'  
    Read site-wide ACLs from file.  
  
'-d'  
  
'--debug'  Enable debugging.  
  
'-i seconds'  
'--idle-timeout=seconds'  
    Set idle timeout value.  
  
'-r seconds'  
'--request-ttl=seconds'  
    Set request time-to-live value.  
  
'-t seconds'  
'--timeout=seconds'  
    Set timeout value.
```

In normal operation, a client, the caller, initiates a rendezvous by sending a CTL_MSG to the server of type 'LOOK_UP' (see 'protocols/talkd.h'). This causes the server to search its invitation tables to check if an invitation currently exists for the caller (to speak to the callee specified in the message). If the lookup fails, the caller then sends an 'ANNOUNCE' message causing the server to broadcast an announcement on the callee's login ports requesting contact. When the callee responds, the local server uses the recorded invitation to respond with the appropriate rendezvous address and the caller and callee client programs establish a stream connection through which the conversation takes place.

18 tftp: TFTP client

Tftp is the user interface to the Internet TFTP (Trivial File Transfer Protocol), which allows users to transfer files to and from a remote machine. The remote host may be specified on the command line, in which case tftp uses host as the default host for future transfers (see the connect command below).

```
tftp [option]... host
```

18.1 Commands

Once tftp is running, it issues the prompt and recognizes the following commands:

? *command-name*

Print help information.

ascii Shorthand for mode ascii

binary Shorthand for mode binary

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 commands.

get *file-name*

get *remotename localname*

get *file...*

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 'hosts: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.

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 *file... 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 'hosts:filename' to specify both a host and 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.

quit Exit tftp. An end of file also exits.

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

`status` Show current status.

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

`trace` Toggle packet tracing.

`verbose` Toggle verbose mode.

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 and therefore difficult to document here.

Appendix A GNU Free Documentation License

Version 1.2, November 2002

Copyright © 2000,2001,2002 Free Software Foundation, Inc.
51 Franklin St, Fifth Floor, Boston, MA 02110-1301, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “you”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released

under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none. The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and

that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements.”

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called

an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

A.0.1 ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright (C) *year your name*.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled ‘‘GNU Free Documentation License’’.

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the ‘‘with...Texts.’’ line with this:

with the Invariant Sections being *list their titles*, with the Front-Cover Texts being *list*, and with the Back-Cover Texts being *list*.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Concept Index

This is a general index of all issues discussed in this manual, with the exception of the commands and command-line options.

-	
--8-bit	31
--acl	46
--address	24
--allow-root	34
--anonymous-only	15
--auth	15
--count	24
--d	46
--daemon	15, 33
--debug	4, 15, 19, 24, 31, 35, 41, 46
--echo	24
--encrypt	28, 32, 34, 35, 38
--environment	19
--escape	31
--file	40
--flood	24
--help	2, 34, 38
--hop	41
--idle-timeout	46
--inet	41
--interval	24
--kerberos	28, 31, 34, 35, 38
--local-domain	34, 38
--logging	15
--mark	41
--max-timeout	15
--nignore-routingo-klog	24
--no-detach	41
--no-escape	31
--no-forward	41
--no-glob	4
--no-keepalive	34, 38
--no-klog	41
--no-login	4
--no-prompt	4
--no-rhosts	34, 38
--no-unixaf	41
--no-version	15
--ntimestamp	24
--numeric	24
--pattern	25
--pidfile	15, 41
--port	3, 34
--preload	25
--preserve	28
--priority	40
--prompt	4
--quiet	25
--r	19
--rate	19
--rcfile	41
--realm	28, 31, 35
--recursive	28
--request-ttl	46
--resolve	19
--resolve-hostnames	3
--reverse-required	34
--route	25
--router	24
--size	25
--socket	41
--tag	40
--timeout	15, 46
--trace	4
--tries	3
--type	3
--umask	16
--user	35
--verbose	4, 24
--verify-hostname	33, 38
--version	2, 34, 38
-8	31
-a	15, 33, 38, 46
-A	15
-c	24
-d	4, 15, 19, 24, 31, 33, 35, 41
-D	15, 34, 38
-debug	34, 38
-e	31
-E	31
-f	24, 40, 41
-g	4
-h	34, 38, 41
-i	4, 24, 40, 46
-k	28, 31, 34, 35, 38
-K	28, 31, 35
-l	15, 25, 34, 35, 38, 42
-L	34, 38
-m	41
-M	3
-n	4, 24, 34, 38, 41
-o	34
-p	3, 4, 15, 25, 28, 34, 40, 41
-q	3, 15, 25
-r	24, 28, 34, 41, 46
-R	25
-s	25, 40, 42
-t	4, 15, 40, 46
-T	15
-u	16
-v	4, 24
-V	34, 38

-x 28, 32, 34, 35, 38

B

bug, reporting 1

F

FDL, GNU Free Documentation License 49

ftp 4

ftpd 15

H

help, online 2

I

inetd 19

introduction 1

L

logger 40

N

networking utilities 1

P

ping 24

R

rcp 28

rexecd 29

rlogin 31

rlogind 33

rsh 35

rshd 37

S

syslogd 41

T

talk 45

talkd 46

tftp 47

traceroute 3

V

version number, finding 2

Index

This is an alphabetical list of all commands, command-line options, and environment variables.

(Index is nonexistent)

Table of Contents

1	Introduction	1
2	Common options	2
3	traceroute: Trace the route to a host	3
4	ftp: FTP client	4
4.1	Commands.....	4
4.2	Aborting A File Transfer	12
4.3	File Naming Conventions.....	12
4.4	File Transfer Parameters	12
4.5	The ‘.netrc’ File.....	13
5	ftpd: FTP daemon	15
5.1	Configuration files.....	18
6	inetd	19
6.1	Invocation	19
6.2	Configuration file	19
6.3	Built-in services.....	21
6.4	TCPMUX	22
6.5	Inetd Environment.....	22
6.6	Error Messages.....	23
7	ping	24
7.1	Invoking	24
7.2	ICMP Packet Details	25
7.3	Duplicate And Damaged Packets.....	26
7.4	Trying Different Data Patterns.....	26
7.5	TTL Details.....	26
8	rcp: Copy files between machines	28
9	rexecd: server for rexec	29
9.1	Diagnostics	29
10	rlogin	31
10.1	Invoking	31
10.2	Kerberos Authentication	32

11	rlogind	33
11.1	Invoking	33
11.2	Diagnostics	34
12	rsh	35
12.1	Invoking	35
13	rshd	37
13.1	Invoking	38
13.2	Diagnostics	38
14	logger: Make entries in the system log	40
15	syslogd: system service logging facility	41
15.1	Configuration file	42
16	talk: a communication program	45
16.1	Invoking	45
17	talkd: a server for communication between users	46
18	tftp: TFTP client	47
18.1	Commands	47
Appendix A	GNU Free Documentation License	49
A.0.1	ADDENDUM: How to use this License for your documents	55
	Concept Index	56
	Index	58