CHAPTER 1

NP O/S Command Reference

This chapter contains the manual pages for those NP O/S commands and system utilities that you are most likely to need while operating your network of LightStream 2020 multiservice ATM switches (LS2020 switches).

The CLI **shell** command accesses the bash shell from the CLI so that you can execute NP O/S commands. You can also execute these commands from the bash shell when you log in as root or fldsup. (If you log in as oper or npadmin, the CLI automatically starts.) The following is a list of all the NP O/S commands that are described in this chapter:

- cat - concatenate files
- cbufpr - print contents of a circular buffer
- ckswinstall - check software installation
- comment - issue a trap with specified text
- cp - copy files
- cs - produce file checksum
- date - display or set current date and time
- df - show free disk space
- fsck - file system check and repair
- ftp - file transfer program
- hostname - print or set name of current host system
- ls - show directory contents and file information
- mkdir - create directories
- more - file perusal filter for CRT viewing
- mv - move or rename files
- passwd - change user password
- ps - display status of current processes
- reboot - reboot the system
- rsh - remote shell
- rm, rmdir - remove (unlink) files or directories
- swchgver - software change version
- swdelete - software delete

- swinfo - software information
- swinstall - software installation
- swremoteinstall - remote software installation
- tail - print last few lines of a file
- tar - combine files into an archive backup
- telnet - user interface to the Telnet protocol
- touch - change the modify date of files

Note The **help** command displays help information about commands that are built in to the bash shell (see the "GNU Bash Shell Reference" chapter). Of particular interest are the commands **cd**, **kill**, and **pwd**.

cat - concatenate files

Synopsis

```
cat [-benstuv] [filename ...]
```

Description

The **cat** command concatenates the contents of one or more files onto the standard output. If no *filename* is given, or if the filename "**-**" is present in the list, **cat** copies standard input to standard output. If standard output is not redirected to a file or through a pipe, the effect of **cat** is to print the files on the current TTY.

The output of **cat** is line buffered if standard output is a TTY device. Otherwise, output is buffered in large blocks.

Switches

The following optional command-line switches are available

-b Omits line numbers from blank lines; used with the **-n** option.

-e Appends a $ character to each line; used with the **-v** option to indicate where the end-of-line character is.

-n Appends the line number to the beginning of each output line.

-s Replaces sequences of blank lines by a single blank line.

-t Converts tab characters to the string ^I; used with the **-v** option.

-u Forces unbuffered (character-by-character) output.

-v Indicates positions of non-printing characters. Non-printable characters whose ASCII value is less than 32 (40 octal) are output as ^ followed by the value plus 64; thus ASCII 26 (control Z) is printed as ^Z. The delete character, ASCII 127 (177 octal), is printed as ^?. Characters whose value is greater than 127 are printed as M- followed by the printable version of the low 7 bits of the value. Note that printable control characters such as newline or tab are not translated.

Note Commands like **cat a b > a** or **cat a b > b** do not work as expected and in fact destroy information because the shell opens and resets the output files before **cat** has a chance to read from the input files.

---

cbufpr - print contents of a circular buffer

Synopsis

```
cbufpr [-all] [-f] [-level {snmp|oper|info|trace|debug}] [-tail [-n]] file
cbufpr {[-h] | -v}
```

Description

The **cbufpr** command displays contents of a circular buffer file on a Network Processor (NP). The **show file** command in the CLI executes this command. Files in appropriate circular buffer format include the following:

```
/usr/tmp/apps.log
/usr/tmp/mma/mma.log
/usr/tmp/mma/mma.traplog
/usr/tmp/configure/configure.netdb.log
```

Options

The *file* argument is the name of a log file to be printed. It may be omitted with the **-h** or **-v** switch.

-all	Accept all formats, including plain file.
-f	Continue reading from tail of file as new entries are added, rather than exiting. Type **^C** to kill the process.
-h	Display the help message. This is the default, with no argument. Any other argument given with the **-h** argument is ignored.
-level *traplevel*	Only display traps above the indicated trap level, where *traplevel* is one of **snmp**, **oper**, **info**, **trace**, and **debug**. Useful only with a file containing trap output in LightStream 2020 trap format.
-tail [-*n*]	Read the last 20 lines of the file, or (approximately) the last *n* lines if the -*n* switch is specified in the same command.
-v	Display information about the version of **cbufpr** currently running. Any other argument given with the **-v** argument is ignored (except preceding **-h**).

ckswinstall - check software installation

Synopsis

```
ckswinstall [-v ver#] [-h]
```

Description

The command **ckswinstall** verifies that the software installation for a particular version was successful.

When verifying a software installation using the **ckswinstall** utility on a redundant-NP system, the software installation on each of the two NPs must be checked explicitly.

Step 1 Run **ckswinstall** on the current primary NP.

Step 2 Run **ckswinstall** on the current backup NP.

```
LSnode:2# rsh other-np ckswinstall ...
```

Options

-h Print a help message.

-v *ver#* Version to be verified (for example, 2.1.0). If **ckswinstall** is executed without a version number, it verifies the current running version.

Bugs

The **ckswinstall** utility reports false permission mismatches on directories under /usr/app. Ignore these messages.

When using **ckswinstall** to verify an update distribution (for instance, 2.0.6), **ckswinstall** only verifies the software in the update. On a system running an update, most of the software in use is from the original release, so **ckswinstall** should also be run on that original release (in the example, 2.0.3). When running **ckswinstall** on the underlying release, false errors are reported against software that was upgraded as part of the update. Ignore these messages.

comment - issue a trap with specified text

Synopsis

```
comment {[-h] | [-v]} comment [-u name] [-level traplevel] [-cons={yes|no}]
[-f filename]
```

Description

The **comment** command accepts a text message as input, either interactively (terminated by **^D** at the beginning of a line) or from a file specified with the **-f** *filename* switch. Optional arguments indicate how to identify the message and where to display it.

The command does the following:

- Generates an Info trap, which is copied to the trap log. The trap level may be changed with the **-level** *traplevel* switch.
- Echoes the message on the terminal at which the command was entered.
- Echoes the message on the system console. If the command was entered at the system console, then both copies of the message appear on the console screen. Console output may be suppressed with the **-cons=no** switch.

The comment text is identified by the label [comment.*PID*]:, where *PID* is the process ID of the **comment** command. The trap is identified by the label Comment_*n*001, where the trap number *n*001 reflects the trap level as follows: 1001 is Oper, 2001 is Info, 3001 is Trace, and 4001 is Debug. In both cases, the invoker is identified by user name appended to the beginning of the message. The user name may be changed with the **-u** *name* switch.

The **comment** command is used mainly to provide a record in the trap log of operator-initiated events. For example, if one is taking down an interface for test purposes, a comment trap can be issued noting why the interface is coming down.

Options

The following optional arguments are available:

-cons={**no** | **yes**}	Suppress (**no**) or enable output to the console. The **-cons=yes** switch is the default.
-f *filename*	Take the trap message text from *filename*, rather than from standard input.
-h	Display a help message.
-level *traplevel*	Determine the trap level for the comment trap, where the *traplevel* argument is one of **oper**, **info**, **trace**, **debug**, and **none**. The default is to generate an info trap. With the **none** argument, no trap is generated.
-u *name*	Specify the user name to be included in the trap message. By default, the author of the trap is specified by login name.
-v	Display program version information. This information is displayed with every invocation of the command, including invocations with the **-h** switch.

cp - copy files

Synopsis

```
cp [-irv] file1 file2
cp [-irv] file1 ... directory
```

Description

The **cp** command creates copies of files. The destination file [*file2*] is created if it does not exist, or overwritten if it does. If more than two files are given, then the last filename must be a directory, and the files are copied into it with their original base names. **cp** does not copy a file to itself.

Options

-i **cp** prompts the user whenever a file is about to be overwritten. Answer **y** to overwrite the existing file; type anything else to skip that file and go on to the next specified file (if any).

-r **cp** recursively copies all subtrees of all *file* arguments which are directories and deposits them into the destination directory. **cp** flags an error if the destination file is not a directory. Hard and symbolic links are not understood by **cp**; a complete copy is made based upon each directory entry in the source subtree. To truly duplicate subtrees, complete with link information, see “tar - combine files into an archive backup” later in this chapter.

-v Print list of files encountered during a recursive **cp**.

Examples

The following example simply copies from file1 into file2:

```
cp file1 file2
```

The following example copies file1, file2, and file3 into the directory dir1:

```
cp file1 file2 file3 dir1
```

The following example copies all files in dir1 recursively into dir2, and also copies file1 into dir2:

```
cp -r dir1 file1 dir2
```

cs - produce file checksum

Synopsis

```
cs [-value] file
```

Description

The **cs** command sums every byte of the named *file*, modulo 256. If no *value* is given, the result is printed to standard output. If a *value* is given, **cs** compares it to the actual checksum and exits with status 1 if the values are different, or with status 0 if the values are the same.

date - display or set current date and time

Synopsis

```
date [-u] [-dn] [-zm] [newdate]
```

Description

The **date** command can be used either to set the current system date and time or to display it. If no arguments are given, then date only displays the current date and time. You must be running as the user root to change the date.

The format for the *newdate* argument describing a new date and time is as follows:

yymmddhhmm[*.ss*]

Here, *yy* is the last two digits of the year, *mm* the number of the month (1-12), *dd* the day number in the month, *hh* the hour (0-23), *mm* the minute, and optionally, *.ss* the second. If the date and seconds parts are omitted, the current values are used.

Options

The following optional arguments are available:

-u Displays the current date in Greenwich Mean Time (GMT). When setting the current time, this flag prevents local transformations of the given time value.

-d*n* Sets the daylight saving method to *n*, which must be one of the following values:

- 0 No daylight savings
- 1 USA
- 2 Australia
- 3 Eastern Europe
- 4 Central Europe
- 5 Western Europe

-z*m* Sets the time zone to *m*, given as minutes from Greenwich (0° longitude). Positive values are for the western hemisphere, for example, for North America, and negative values are for longitudes east of Greenwich.

Examples

The following example shows how to set the date to December 8, 1986 12:30 AM GMT, and the daylight saving method to USA.:

```
date -d1 -u 8612080030
```

The following example shows how to leave the current time the same, but change the time zone to Central Standard Time:

```
date -z360
```

df - show free disk space

Synopsis

```
df [-i] [file ...]
```

Description

The **df** program displays the total space and amounts of used and free space on selected mounted file systems. If *file* names a device on which a file system is mounted, **df** gives information for that file system.

If *file* refers to a regular file or directory, **df** gives information for the filesystem containing that file. If no *file* arguments are given, then **df** reads the file /etc/fstab and produces a report for each device listed there.

Option

-i Give information on free and used inodes as well as free and used space.

fsck - file system check and repair

Synopsis

`fsck` *block-dev*

Description

The **fsck** command checks and repairs the file system on the named block device. It looks for inconsistencies among the following file system components:

Inode contents	All of the information recorded in each active inode is checked for consistency. The block pointers are all checked for validity. Cross-checks are made to ensure that blocks are owned by at most one inode. The block and byte sizes recorded in the inode are verified and corrected if necessary.
Directory structure	Directory contents are checked for correctness and consistency. Reference counts for each inode are re-computed and repaired if necessary.
Free inode list	The free inode list is rebuilt in reverse order of inode number.
Free block list	The free block list is rebuilt in order of block number.

Any inconsistencies found are reported and repaired or otherwise dealt with.

If orphaned files exist, a /lost+found directory is created and the original /lost+found, if it exists, is renamed to the next name in the set /lost+found[A-z]. The orphaned file is put into /lost+found and its file name is set to its inode number.

Example

The following example shows the use of the **fsck** command to check and repair the file system on /dev/sd0b:

```
LSnode:2# fsck /dev/sd0b
(all sizes and block numbers in decimal)
(file system creation time is Mon Apr 11 04:57:52 1994)
checking used files
recovering orphaned files
making free block list
making free inode list
32192 free blocks 3361 free inodes
LSnode:2#
```

ftp - file transfer program

Synopsis

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

Description

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

The client host with which **ftp** is to communicate may be specified on the command line. If this is done, **ftp** immediately attempts to establish a connection to an FTP server on that host; otherwise, **ftp** enters its command interpreter and awaits instructions from the user.

Options

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

-v (Verbose) forces **ftp** to show all responses from the remote server, as well as report on data transfer statistics.

-n Restrains **ftp** from attempting "auto-login" upon initial connection. If "auto-login" is enabled, **ftp** checks the **.netrc** file in the user's home directory for an entry describing an account on the remote machine. If no entry exists, **ftp** uses the login name on the local machine as the user identity on the remote machine, and prompts for a password and, optionally, an account with which to log in.

-i Turns off interactive prompting during multiple file transfers.

-d Enables debugging.

-g Disables file name globbing.

Commands

When **ftp** is awaiting commands from the user, the prompt **ftp**> is provided for the user. The following commands are recognized by **ftp**:

!	Invoke a shell on the local machine.
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. 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	Terminate the FTP session with the remote server and exit **ftp**.
cd *remote-directory*	Change the working directory on the remote machine to *remote-directory.*

close	Terminate the FTP session with the remote server, and return to the command interpreter.
connect host [*port*]	A synonym for open.
delete *remote-file*	Delete the file *remote-file* on the remote machine.
debug [*debug-value*]	Toggle debugging mode. If an optional *debug-value* is specified, it is used to set the debugging level. When debugging is on, **ftp** prints each command sent to the remote machine, preceded by the string ->.
dir [*remote-directory*] [*local-file*]	Print a listing of the directory contents in the directory *remote-directory* and, optionally, place the output in *local-file*. If no directory is specified, the current working directory on the remote machine is used. If no local file is specified, output comes to the terminal.
form *format*	Set the file transfer form to *format*. The default format is ”file.”
get *remote-file* [*local-file*]	Retrieve *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. The current settings for type, form, mode, and structure are used while transferring the file.
hash	Toggle hash-sign (**#**) printing for each data block transferred. The size of a data block is 1024 bytes.
glob	Toggle file name globbing. With file name globbing enabled, each local file or pathname is processed for shell metacharacters. These characters include “*****”, “**?**”, “**[**”, “**]**”, “ ” (that is, space), “**{**”, “**}**”. Remote files specified in multiple-item commands, for example, **mput**, are globbed by the remote server. With globbing disabled, all files and pathnames are treated literally.
help [*command*]	Print an informative message about the meaning of *command*. If no argument is given, **ftp** prints a list of the known commands.
lcd [*directory*]	Change the working *directory* on the local machine. If no directory is specified, the user’s home directory is used.
ls [*remote-directory*] [*local-file*]	Print an abbreviated listing of the contents of a directory on the remote machine. If *remote-directory* is left unspecified, the current working directory is used. If no *local file* is specified, the output is sent to the terminal.
mdelete *remote-files*	Delete the specified files on the remote machine. If globbing is enabled, the specification of remote files is first expanded using **ls**.
mdir *remote-files local-file*	Obtain a directory listing of multiple files on the remote machine and place the result in local-file.
mget *remote-files*	Retrieve the specified files from the remote machine and place them in the current local directory. If globbing is enabled, the specification of remote files is first expanded using **ls**.
mkdir *directory-name*	Make a directory on the remote machine.
mls *remote-files local-file*	Obtain an abbreviated listing of multiple files on the remote machine and place the result in local-file.

mode [*mode-name*]	Set the file transfer mode to *mode-name*. The default mode is "stream" mode.
mput *local-files*	Transfer multiple *local files* from the current local directory to the current working directory on the remote machine.
open *host* [*port*]	Establish a connection to the FTP server *host*. An optional *port* number may be supplied, in which case, **ftp** attempts to contact an FTP server at that port. If "auto-login" is enabled, **ftp** also attempts to automatically log the user in to the FTP server (see below).
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), any mget or mput transfers all files.
put *local-file* [*remote-file*]	Store a *local file* on the remote machine. If *remote-file* is left unspecified, the local file name is used in naming the remote file. File transfer uses the current settings for type, format, mode, and structure.
pwd	Print the name of the current working directory on the remote machine.
quit	A synonym for bye.
quote *arg1 arg2 ...*	The arguments specified are sent, verbatim, to the remote FTP server. A single FTP reply code is expected in return.
recv *remote-file* [*local-file*]	A synonym for **get.**
remotehelp [*command-name*]	Request help from the remote FTP server. If a *command-name* is specified it is supplied to the server as well.
rename [*from*] [*to*]	Rename the file *from* on the remote machine, to the file *to*.
rmdir *directory-name*	Delete a directory on the remote machine.
send *local-file* [*remote-file*]	A synonym for put.
sendport	Toggle the use of PORT commands. By default, **ftp** attempts to use a PORT command when establishing a connection for each data transfer. If the PORT command fails, **ftp** uses the default data port. When the use of PORT commands is disabled, no attempt is made to use PORT commands for each data transfer. This is useful for certain FTP implementations which do ignore PORT commands but, incorrectly, indicate that they have been accepted.
status	Show the current status of **ftp**.
struct [*struct-name*]	Set the file transfer structure to *struct-name*. By default, "stream" structure is used.
tenex	Set the file transfer type to that needed to talk to TENEX machines.
trace	Toggle packet tracing.
type [*type-name*]	Set the file transfer type to *type-name*. If no type is specified, the current type is printed. The default type is network ASCII.

user *user-name* [*passwd*] [*acct*]	Identify yourself to the remote FTP server. If the password is not specified and the server requires it, **ftp** prompts the user for it (after disabling local echo). If an account field *acct* is not specified, and the FTP server requires it, the user is prompted for it. Unless **ftp** is invoked with ”auto-login” disabled, this process is done automatically on initial connection to the FTP server.
verbose	Toggle verbose mode. In verbose mode, all responses from the FTP server are displayed to the user. 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*]	A synonym for **help**.

Command arguments which have embedded spaces may be quoted with double quote (”) marks.

File Naming Conventions

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

1. If the file name “-” is specified, then 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;for example, “-- 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 shell; refer to the **glob** command.

File Transfer Conventions

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-10s and PDP-20s mostly). The **ftp** command supports the ASCII and image types of file transfer.

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

Note Aborting a file transfer does not work right; if one attempts this the local **ftp** will likely have to be killed by hand.

Acknowledgements

This section was developed by the University of California, Berkeley.

hostname - print or set name of current host system

Synopsis

```
hostname [name]
```

Description

The **hostname** command prints the name of the current host. The optional *name* argument is accepted only with superuser privilege, to set the host name.

The host name is normally set when the operating system is bootstrapped. If the hostname command gives no output, execute it as superuser with the host name as argument.

ls - show directory contents and file information

Synopsis

```
ls [-1ACFLRacdfgilnqrstu -T types] file ...
```

Description

The **ls** command writes to standard output information requested by options for each file given. If a file is a directory, then the entire contents of that directory are written. Output is sorted alphabetically by default, but can optionally be sorted by access or modification time. If no file is given, the current directory is assumed.

The list of files is sorted alphabetically; non-directory arguments are always processed before directory arguments.

Options

-1 One line of output is generated for each file. This option is assumed when standard output is not to a terminal.

-A When listing directory contents, all files are included, including those whose names begin with "**.**" (except for the directory names "**.**" and "**..**").

-C **ls** lists as many files per output line as possible. This option is assumed when standard output is a terminal.

-F **ls** appends markers to file names in the listing to indicate file type, as follows:

/	directory
&	named pipe (FIFO)
=	socket (not implemented in LynxOS)
@	symbolic link
*****	executable regular file
|	pipc special file

-L For each argument or directory element which is a symbolic link, information about the file or directory referenced by the link (instead of the link itself) is used for printing, sorting, and/or marking the output.

-R Each subdirectory encountered while scanning a directory is itself examined.

-T *types* Only those files are shown whose type is one of those specified by *types*. The types list should be a string of one or more of **-**, **b**, **c**, **d**, **f**, **l**, **p**, **r**, **s**, **i**, **I**, and **+**, as described below.

-a When listing directory contents, all files are printed, including files which begin with "**.**".

-c The time value examined for printing (**-l**) and sorting (**-t**) is the file creation time instead of the modification time.

-d Instructs **ls** to give information about directory arguments themselves instead of listing the contents of the directories.

-f Each argument is interpreted as a directory, and contents are listed in the order encountered (no sort is performed). The **-f** option disables **-l**, **-t**, **-s**, and **-r**, and enables **-a**.

-g Group ID of each file is printed (only effective when **-l** is also given).

-i The inode number of each file is printed.

-l Extended information is given for each file, including file mode, number of links, owner, size in bytes, and time (usually modification time, but optionally access or file creation time). If a file is a special file, the major and minor numbers are given in place of file size. If a file is a symbolic link, then the target of the link is given after the file name.

-n Restricts listing to files created on the current system date.

-q Forces non-printable characters in file names to be printed as "?". This option is assumed when standard output is a terminal.

-r Reverses sort order.

-s Lists size of each file, in 512-byte blocks.

-t Sorts by time rather than by name.

-u Uses time of last access rather than modification time for printing (**-l**) and sorting (**-t**).

Mode Information

The file mode given with the **-l** option is printed as a 10-character string. The first character gives general file type:

- Regular file

f Regular file

r Regular file

b Block special file

c Character special file

d Directory file

p Named pipe (FIFO)

l Symbolic link

s Socket

+ Contiguous file

i Non-persistent ipc special file

I Persistent ipc special file

The next nine characters tell whether or not read, write, and execute permissions are on for the owning user, owning group, and others:

r read permission

w write permission

x execute permission

The permissions are printed in the order read-write-execute; a dash in a position means that permission is off. For directory files, execute permission is interpreted as permission to search the directory. If the “sticky bit” in the mode is turned on (see the system call chmod), the last character of the mode string is printed as t rather than the usual x or -.

mkdir - create directories

Synopsis

`mkdir [-pf]` *directory* `...`

Description

The **mkdir** command creates directory files with the given pathnames. The standard entries "." and ".." are automatically created in the new directory. The **mkdir** program must have write permission into the parent directory of the new directory or directories.

Options

-f Suppresses error messages.

-p The **mkdir** command creates each non-existent directory along each given pathname. Thus the following command creates the subdirectory integers along with **test1**, **data**, and **numlists**, if any of those do not exist:

```
mkdir -p test1/data/numlists/integers
```

See Also

rm, rmdir - remove (unlink) files or directories

more - file perusal filter for CRT viewing

Synopsis

```
more [-cdflsu] [-n] [+linenumber] [+/pattern] [name ...]
```

Description

The **more** program is a filter which allows examination of continuous text, one screenful at a time on a soft-copy terminal. It normally pauses after each screenful, printing --More-- at the bottom of the screen. Type a carriage return to display one more line, and press the spacebar to display another screenful. Other possibilities are enumerated under Options.

Options

The command line options are as follows:

n	An integer which is the size (in lines) of the window which more will use instead of the default.
-c	The **more** program draws each page by beginning at the top of the screen and erasing each line just before it draws on it. This avoids scrolling the screen, making it easier to read while more is writing. This option is ignored if the terminal does not have the ability to clear to the end of a line.
-d	The **more** program prompts the user with the message “Press space to continue, `q' to quit at the end of each screenful,” and responds to subsequent illegal user input by printing “Press `h' for instructions instead of ringing the bell.” This is useful if **more** is being used as a filter in a setting, such as a class, where many users may be unfamiliar with **more**.
-f	This causes **more** to count logical lines, rather than screen lines; that is, lines that are not folded.
-l	Do not treat control-l (form feed) in a special way. If this option is not given, **more** pauses after any line that contains a control-l, as if the end of a screenful had been reached. Also, if a file begins with a form feed, the screen is cleared before the file is printed.
-s	Squeeze multiple blank lines from the output, producing only one blank line.
-u	Normally, **more** handles underlining in a manner appropriate to the particular terminal: if the terminal can perform underlining or has stand-out mode, **more** outputs appropriate escape sequences to enable underlining or stand-out mode for underlined information in the source file. The -u option suppresses this processing.
+linenumber	Start up at linenumber.
+/pattern	Start up two lines before the line containing the regular expression pattern.

The **more** program looks in the file /etc/termcap to determine terminal characteristics, and to determine the default window size. On a terminal capable of displaying 24 lines, the default window size is 22 lines.

The **more** program looks in the environment variable MORE to pre-set any flags desired. For example, if you prefer to view files using the -c mode of operation, the csh command setenv MORE -c or the sh command sequence MORE='-c', export MORE would cause all invocations of more to run with -c set. Normally, one should place the command sequences which set up the MORE environment variable in the .cshrc or .profile file.

If more is reading from a file, rather than a pipe, then a percentage is displayed along with the --More-- prompt. This gives the fraction of the file (in characters, not lines) that has been read so far.

Other sequences which may be typed when more pauses, and their effects, are as follows (i is an optional integer argument, defaulting to 1):

[i]<space>	Display i more lines (or another screenful if no argument is given).
[i]control-d	Display 11 more lines. If i is given, then the scroll size is set to i.
[i]d	Same as control-d.
[i]z	Same as typing <space> except that i, if entered, becomes the new window size.
[i]s	Skip i lines and print a screenful of lines.
[i]f	Skip i screenfuls and print a screenful of lines.
[i]b	Skip back i screenfuls and print a screenful of lines.
[i]control-B	Same as b.
q or Q	Exit from **more**.
=	Display the current line number.
v	Start up the editor vi at the current line.
h	Help command. Gives a description of all the more commands.
[i]/expr	Searches for the i-th occurrence of the regular expression expr. If there are less than i occurrences of expr, and the input is a file (rather than a pipe), then the position in the file remains unchanged. Otherwise, a screenful is displayed, starting two lines before the place where the expression was found. The user's erase and kill characters may be used to edit the regular expression.
[i]n	Search for the i-th occurrence of the last regular expression entered.
`	(single quote) Go to the point from which the last search started. If no search has been performed in the current file, this command goes back to the beginning of the file.
!command	Invoke a shell with command. The characters % and ! in command are replaced with the current file name and the previous shell command respectively. If there is no current file name, % is not expanded. The sequences and are replaced by % and !, respectively.
[i]:n	Skip to the i-th next file given in the command line. (Skips to the last file if i does not make sense.)

[i]:p Skip to the i-th previous file given in the command line. If this command is given in the middle of printing out a file, then **more** goes back to the beginning of the file. If i does not make sense, **more** skips back to the first file. If **more** is not reading from a file, the bell is rung and nothing else happens.

:f Display the current file name and line number.

:q or :Q Exit from **more** (same as q or Q).

The commands take effect immediately, that is, it is not necessary to type a carriage return. Up to the time when the command character itself is given, the line kill character cancels the numerical argument being typed, and the erase character redisplays the --More-- (xx%)prompt.

When output is being sent to the terminal, the quit character causes **more** to stop sending output, and it displays the usual --More-- prompt. Any of the above commands may then be entered in the normal manner. Unfortunately, some output is lost when this is done, due to the fact that any characters waiting in the terminal's output queue are flushed when the quit signal occurs.

The terminal is set to noecho mode by this program so that the output can be continuous. This means that what you type does not show on your terminal, except for the / and ! commands.

If the standard output is not a teletype, then **more** acts just like **cat**, except that a header is printed before each file (if there is more than one).

If you do not know what the line kill or quit characters for your terminal are, you can find this information by using the **stty** command.

Acknowledgments

This section was developed by the University of California, Berkeley.

FILES

/etc/termcap	Terminal data base
/usr/lib/more.help	Help file

SEE ALSO

Utility Programs - csh, sh, stty

mv - move or rename files

Synopsis

```
mv [-fi] [-] file1 file2
mv [-fi] [-] file ... directory
```

Description

In the first form, **mv** changes the name of *file1* to *file2*. If pathname *file2* describes a different directory than *file1*, then the link in the original directory is removed and a link is created in the new directory. In the second form, each *file* is moved from its former directory into *directory*, retaining its base name. In either form, if the move crosses a file system boundary, **mv** copies the information from the old file to the new file, then removes the old file.

If the destination file exists, and its permissions disallow writing, **mv** prompts the user and completes the move only on a response of **y.**

Options

- Allows first filename to begin with "**-**".

-f Force writing over existing files. Overrides the **-i** switch.

-i The user is prompted whenever a destination file already exists, regardless of permissions. The move is completed only on response **y.**

Note The **mv** command does not allow a file to be moved onto itself. The side-effects of cross-file system moves are that the user ID of the copying process becomes the owner of the new file, and all linking information tied to other files in the original file system is lost.

---

passwd - change user password

Synopsis

```
passwd [user]
```

Description

The **passwd** command replaces the current password recorded in the **/etc/passwd** file for the current user or for the user named *user*. It prompts the user once for the current password, and then twice for a new one to guard against typographic errors. New passwords must be a minimum of 4 characters long if many different characters are used, 6 if the user decides to use a dull, monocase password.

Only the owner of the account or the superuser may change a password. Passwords are kept in the publicly readable **/etc/passwd** file in an encrypted form.

Files

/etc/passwd	User names and passwords
/etc/ptemp	Temporary copy of **/etc/passwd**

ps - display status of current processes

Synopsis

```
ps [afmnostTx] [r [delay]] [-p pid] [-u uid] [-g pgrp]
```

Description

The **ps** command displays the current status of processes in the system. The statistics available for each process are as follows:

`pid` Process ID number.

`tid` Thread ID number.

`ppid` Process ID of the parent process.

`pgrp` Process group number.

`pri` System priority.

`text` Size of text (executable code) segment, in terms of 1 kilobyte units. If the text segment is being shared with another process displayed by **ps**, the text size is followed by an asterisk.

`stack` Size of the process stack segment.

`data` Size of the process data segment.

`sem` Semaphore ID number on which process is waiting, if any.

`time` User and system time used so far by the process. Given as *s.ss* seconds, *mm*`:`*ss* minutes and seconds, *hh*+*mm*`:`*ss* hours, minutes and seconds, or *hhhhh*+*mm* hours and minutes, as space allows.

`dev` Control device of the process, as recorded in /dev.

`flags` Value of threads' flags.

`user` User name (based on effective user ID) of the process owner.

`S` Current process state, one of

- `C` Currently executing process (usually **ps** itself).
- `R` Ready to run, but not running.
- `S` Suspended by a signal (`SIGSTOP`, `SIGTSTP`, `SIGTTIN`, or `SIGTTOU`).
- `W` Waiting on a semaphore for some resource.
- `E` New process awaiting completion of its fork operation.

`name` Pathname of the executing process.

`dlim` Current data size resource limit, in terms of 1 kilobyte units.

`d%` Percentage of data size limit currently used.

`slim` Current stack size resource limit.

`s%` Percentage of stack size limit currently used.

`smem` Shared memory pages in use, and number of shared memory regions.

By default, **ps** displays all processes with the same effective user ID as the current real user ID.

Options

The following command-line options are available:

a Includes processes owned by others.

f Substitute the `flags` field for the `dev` field

m Show memory usage display instead of normal system use display. This option provides the `dlim`, `d%`, `slim`, `s%`, and `smem` fields but leaves out `sem`, `time`, `dev`, `user`, and `S`.

n Includes the operating system "null" pseudo-process in the output. The accumulated time for the null process is an indication of system idle time.

o Includes the supervisor stack size and process state segment size in the stack and data size values.

-p *pid* Includes information for process *pid*.

-g *pgrp* Includes information for all processes with process group number *pgrp*.

r *delay* Tells **ps** to repeat indefinitely. The display (selected by the other options) is drawn at the top of the CRT. By default, **ps** waits 10 seconds between updates; *delay* may be given to increase or decrease the update frequency.

s Substitute the `sem` field for the `time` field.

t Includes thread IDs.

T Includes stream tasks.

-u *uid* Includes information for all processes owned by user *uid*.

x Includes processes, owned by you, with no control device.

Files

`/dev/mem` System memory device, used to get information about processes.

Diagnostics

Since **ps** cannot sample and display the status of all processes at the speed at which the processes run, its accuracy is limited.

reboot - reboot the system

Synopsis

```
reboot [-ndamfph]
```

Description

The **reboot** command calls the operating system reboot function to restart the entire system. The options to **reboot** allow bits to be set in the flag word that is passed to the operating system:

-n	RB_ASKNAME	Boot monitor will request bootstrap name.
-d	RB_NOSYNC	Do not sync before reboot.
-a	RB_AUTOBOOT	Non-interactive boot from default bootstrap.
-m	RB_MFSCK	Simulate corrupted disk cache upon reboot.
-f	RB_NOFSCK	Check file systems after boot.
-p	RB_PANIC	Simulate OS panic.
-h	RB_HALT	Halt the processor for manual reboot.
-s	RB_NOTREALLY	Set reboot flags, but do not really reboot.
-N	RB_MAKENODES	Instructs init to create nodes in /dev starting rc.

Probably the most common type of reboot is:

```
reboot -a
```

This command reboots to multiuser mode after syncing the disks.

To reboot to single user:

```
reboot
```

Another example of reboot is:

```
reboot -am
```

This command boots the OS into multi-user mode after performing a file systems check. The **-m** option of **reboot** forces the file systems check.

When testing a new operating system configuration, use the following command:

```
reboot -anN
```

This command syncs the disks and returns to the low-level monitor, which then requests the name of the new bootstrap program (the newly configured OS). The **-N** switch is very important if the new OS contains new device drivers and devices.

Note

The **reboot** command can only be executed by user 0 (root).

See Also

Utility Programs - fsck

rsh - remote shell

Synopsis

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

Decription

The **rsh** program connects to the specified host, and there executes the specified command. The **rsh** program 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.

The remote username used is the same as the current local user name, unless a different remote name is given with the **-l** option. This remote name must be equivalent to the originating account; no provision is made for specifying a password with a command.

If *command* is omitted, then instead of executing a single command, **rsh** chains to **rlogin** to log into the remote host.

Shell metacharacters which are not quoted are interpreted on the local machine, while quoted metacharacters are interpreted on the remote machine. Thus, the following command appends *remotefile* to *localfile*:

rsh *otherhost* **cat** *remotefile* >> *localfile*

However, the following command appends *remotefile* to *otherremotefile*:

rsh *otherhost* **cat** *remotefile* `>>' *otherremotefile*

The **-n** option redirects output to /dev/null. This option is useful to avoid unexpected interaction between **rsh** and the shell that invokes the **rsh** command.

Host names are given in the file /etc/hosts. Each host has one standard name (the first name given in the file), which is rather long and unambiguous, and optionally one or more nicknames. The host names for local machines are sometimes made commands in the directory /usr/hosts; if you put this directory in the search path, then the word **rsh** can be omitted.

Note

The **rlogin** executable must be located in the standard search path in order to be found by a command-less invocation of **rsh**.

Stop signals stop the local **rsh** process only; this is arguably wrong, but currently hard to fix for reasons too complicated to explain here.

Files

/etc/hosts
/usr/hosts/*
/etc/hosts.equiv (on remote host).
$HOME/.rhosts (on remote host).

rm, rmdir - remove (unlink) files or directories

Synopsis

`rm [`*options*`] `*file*` ... rmdir `*directory*` ...`

Description

The **rm** command removes directory entries for files. If an entry was the last link to a file, the file is removed as well. Removal of an entry requires write permission in its directory. If a file is not writeable, and **rm** is using the current terminal for its standard input, the file permissions are printed on the screen and a response is read from the terminal. **rm** proceeds to remove the file if the response is **y**.

The **rmdir** command removes directories. Each directory must be empty except for the special entries "." and "..".

Options

- Allow all arguments following to be treated as filenames. This option is useful for filenames which begin with the "**-**" character

-f Force removal of unwriteable files.

-i Ask user's permission every time a file or directory is to be removed. If the user responds with a **y**, the file is removed.

-r Recursively remove a directory subtree. **rm** does not remove a directory unless this option is specified.

swchgver - software change version

Synopsis

swchgver [*version#*] [-noreboot] [-nolinecardreset] [-nordist] [-noflashupdate] [-flashforce] [-force] [-h]

Description

The **swchgver** command changes the symbolic link used in determining which version of the software will be run at start up time.

When no version is given, **swchgver** only works if the symbolic link /usr/app/base-new exists.

By default, **swchgver** updates the Flash, resets the line cards and reboots the network processor.

On a redundant system, **swchgver** should be executed from the primary network processor. If contact is possible, the primary copies the version of the software to the standby network processor using rdist before rebooting it. Then **swchgver** continues executing on the primary network processor by updating the Flash, resetting the line cards and rebooting the primary network processor.

Options

-flashforce	Reload Flash even if it has not changed.
-force	Ignore failures to contact backup network processor on a redundant system. By default, **swchgver** will abort with an error if it cannot contact the backup network processor on a redundant system.
-h	Produces a help message.
-noflashupdate	Do not update Flash even if it has changed.
-nolinecardreset	Do not reset linecards in the chassis.
-noreboot	Do not reboot the network processor.
-nordist	Do not rdist the files to the backup network processor
version#	Set the software version to *version#*. The default is to set the software version to that which was most recently installed. This is the version of software to run the next time the NP is rebooted (for example, 2.1.0).

swdelete - software delete

Synopsis

swdelete [*version*] [-f]

Description

The **swdelete** command deletes the indicated release, first checking to make sure that the release is not currently in use. (For update releases, both the update and the underlying major release are in use).

When no version is given, **swdelete** executes the default of **swinfo**, which is to list all installed releases, identifying the currently running version and (if it is a partial release) the releases on which the currently running version depends. See the following section, "swinfo - software information."

Swdelete is normally run on the current primary network processor. When run on the primary network processor, the command deletes the version on both network processors if it is found on either network processor. When run on a network processor that is not primary, the software on the primary network processor is ignored.

Options

-f	Remove even if currently running this version
version	Version of software to delete (for example, 2.1.0)

See Also

swinfo - software information

swinfo - software information

Synopsis

```
swinfo [-currentonly] [-inuseonly] [-localonly] [-h]
```

Description

The **swinfo** command is used to gather information about the software that is installed on the LightStream 2020. This command also reports on the status of the backup NP should the machine be a redundant system.

The default lists all installed releases, identifying the currently running version and (if it is a partial release) the releases on which the currently running version depends.

On a redundant system, the program lists releases installed on the other network processor. It uses a star (*) to mark releases installed on one network processor, but not on the other.

The **swinfo** program is normally run on the current primary network processor. When it is run on a network processor that is not primary, the software on the primary network processor is ignored

Options

-currentonly	List only the currently running version.
-h	Print a help message.
-inuseonly	List only the currently in-use versions (the currently running version and any versions on which that version depends).
-localonly	On a redundant network processor system, do not list versions installed on the other network processor.

swinstall - software installation

Synopsis

```
swinstall
```

Description

The **swinstall** command installs the LightStream 2020 software from floppy diskette. You must execute this command for each diskette set in each release.

The command does the following, with displays at each step:

Step 1 Makes a request for the first diskette to be placed in the floppy drive.

Step 2 Checks to make sure that the disk inserted is the first disk, and responds with an error and an another request if it is not.

Step 3 If the diskette set is an Application or an Update diskette set, the program checks disk space for partitions /sd0b and /sd0c to verify a successful installation. If there is enough space on these partitions, then the installation continues to the next space. Otherwise, an error is given indicating that an old release must be deleted. Refer to the preceding section, "swdelete - software delete," for details on how to accomplish this.

Step 4 Starts the preinstallation stage, which varies for each diskette set.

Step 5 Finishes the preinstallation stage.

Step 6 Starts the installation stage, which extracts the files from **tar** format and then decompresses them using **uncompress**.

Step 7 Finishes the installation stage.

Step 8 Starts the software validation stage, which verifies that all the files were successfully copied from the diskette.

Step 9 Finishes the software validation stage.

Step 10 Determines if there are additional diskettes in the set that need to be installed. If the diskette is not the last in the set, the next diskette is requested. Once the correct disk is successfully placed into the floppy drive and the return key is pressed, the installation continues at Step 6.

Step 11 Starts the postinstallation stage, which varies for each diskette set.

Step 12 Finishes the postinstallation stage and issues a message saying that the installation of the diskette set is complete.

Bugs

The **swinstall** program can fail due to lack of free memory if extra processes are running. Make sure processes which are not needed during the installation, such as CLI, are not running.

See Also

swdelete - software delete
swremoteinstall - remote software installation

swremoteinstall - remote software installation

Synopsis

`swremoteinstall -h` *hostname version*

Description

The **swremoteinstall** command copies a LightStream 2020 release from one machine to another using rdist. Both the hostname and version must be specified.

The command does the following, all of which is displayed to the operator:

Step 1 Displays the diskette sets appearing in the distribution and those that are not.

Step 2 Checks for available disk space on portions /sd0b and /sd0c of the target machine.

- If enough space is available, then the installation continues. Otherwise, an error is given indicating that an old release must be deleted. Refer to the preceding section, "swdelete - software delete," for details on how to accomplish this.
- If you receive the error "Permission denied," you need to modify the .rhosts file on the target machine. Refer to the Upgrade Procedure found in the Release Notes for complete instructions on how to modify the .rhosts file.

Step 3 Installs the description of the release on the target machine.

Step 4 Executes the preinstallation steps for all the diskette sets included in this release.

Step 5 Installs all the files for each diskette set, listing them one at a time.

Step 6 Creates a symbolic link named /usr/app/base-new that points to the release. This is used by **swchgver** to switch over to that version. See "swchgver - software change version" earlier in this chapter.

Step 7 Executes the postinstallations for all the diskette sets included in this release.

Step 8 Reports the completion of the remote installation and advises to run **swchgver** on the target machine. See "swchgver - software change version" earlier in this chapter.

Options

-h *hostname* Host to be updated.

version Version of software to install (for example, 2.1.0)

Files

/usr/app/dist/base-<version>/DIST/<diskette set>/rdist.flist
The list of files and symbolic links to be copied over to the target machine.

/usr/app/dist/base-<version>/DIST/<diskette set>/rdist.devlist
The list of device files to be copied over to the target machine.

See Also

swinstall - software installation
swchgver - software change version
swdelete - software delete

tail - print last few lines of a file

Synopsis

`tail [`*options*`] [`*file*`]`

Description

The **tail** command copies to standard output lines from the end of the named *file*, or from standard input if no *file* is given. By default, the last ten lines are printed.

Options

-f tail	Print the tail of the file, then wait for the file to grow, printing each new line as it appears. Useful for watching log files.
+*n*[**blc**]	Begin output starting *n* blocks (b), lines (l), or characters (c) from the beginning of the file. Default unit is lines.
-*n*[**blc**]	Begin output starting *n* blocks, lines, or characters from the end of the file. Default unit is lines.
-r	Print lines in reverse order. If no distance is specified, tail prints the entire file in reverse.

Diagnostics

Strange things may happen if **tail** is used with character special files.

tar - combine files into an archive backup

Synopsis

```
tar {ctxru} [opbXIBfhmvwk] [blksize] [tape] [-C dir] [-P prefix] files ...
```

Description

The **tar** command is historically intended to combine files into an archive on a tape. The archive need not actually be a tape. **tar** operates according to actions given as a command and zero or more modifiers. Each file is copied to or from the archive; directory files are taken to mean the entire subtree beneath the directory.

When creating an archive, **tar** provides two special mechanisms to allow the archive to reflect a different directory structure than the source of the archived files. A pair of arguments **-C** *dir* causes the **tar** process to switch to directory *dir* before interpreting subsequent filenames. A pair of arguments **-P** *prefix* causes the string *prefix* to be appended to the beginning of each filename as it is recorded in the **tar** output file. Any number of directory-switch and prefixing commands can be given in the file list. Note that once the **tar** process has changed its working directory, subsequent relative pathnames given in **-C** requests are interpreted relative to the then current directory.

Options

The **tar** command string must be the first argument given. It need not (but can) begin with a dash. The basic operation of **tar** is dictated by one of the following letters embedded in the string:

c Create a new output archive.

r Append files to the end of an existing archive.

t Pertinent information about files on an existing archive is printed to the standard output. If no file arguments are given to select files in the archive, all files in the archive are shown.

u Each file is appended to the archive only if it is not there already, or if it has been changed since last put there. Note that the older versions of files are not deleted.

x Each file is extracted from the tape. If a file is actually a directory, the entire subtree is extracted. Shell wildcard operators are recognized within file arguments.

Bugs

The **tar** command can be used with a floppy disk, but error messages are uninformative, and the **tar** process can hang.

If no diskette is present in the drive, or if the diskette in the drive has not been formatted for an IBM PS/2 (models 50, 60, 70, 80, and compatibles), uninformative error messages appear on the console.

If the diskette in the drive has media errors, the **tar** process may hang, preventing access to the floppy drive or to the terminal on which the command was executed. If this happens, the NP must be rebooted.

Modifiers

The following characters may also appear in the command string, and alter the behavior of the basic **tar** commands.

b Instructs **tar** to recognize the argument following the key as the blocking factor. Tar constructs the archive as groups of 512-byte records. The default blocking factor is 20 records; the maximum blocking factor is 256 records. Each block is written with one write request, and therefore constitutes a physical record on a raw tape device. The blocksize is determined automatically when reading an archive.

Note: If you use both the **b** and the **f** modifiers, be careful to enter their arguments in the correct order. For example:

tar cbf 20 archive
or
tar cfb archive 20

B This command has no effect and exists for backward compatibility only. Older versions of **tar** truncate the final block of an archive so that its blocking factor is not always the same as the other blocks. The **B** command then forced **tar** to pad the final block with null records up to the blocking factor. The current version of **tar** always pads the final block, regardless of the **B** command.

f Instructs **tar** to recognize the argument following the key as the name of the archive. If omitted, **tar** writes to the file /dev/tar.default. If the file name given is a single dash ('-'), **tar** writes to standard output or reads from standard input. The **tar** command thus provides the method of choice for moving directory subtrees, as follows:

```
cd source; tar cf - | (cd dest; tar xf -)
```

See note under the **b** option.

h Instructs **tar** to follow symbolic links when creating an archive. Normally, symbolic links are recorded as such in the output.

k Instructs **tar** not to repair access times of files read during archive creation. Normally, **tar** ensures that access and modification times of files read during archive creation are not affected by the process.

l Causes **tar** to produce error messages during an extraction when attempts to recreate hard links fail.

m Prevents restoration of modification times. The time stamp on extracted files is equal to the time at the inception of the **tar** process.

o Prevents **tar** from explicitly recording directory files in the archive.

X Instructs **tar** to use POSIX-format **tar** headers, instead of the (simpler and less powerful) UNIX format headers. LynxOS **tar** can read either format.

I Ignore checksum. Use if checksum error is reported and you would like to extract the files anyway.

p Causes present umask information to be ignored when restoring modes of extracted files.

R Suppresses attempts to rename existing files to resolve type conflicts with the archive (see below).

v Sets verbose mode, so that **tar** prints information about each file processed. When checking the contents of an archive (the **t** command), the verbose option gives more information about the files. Normally **tar** operates silently.

w Causes **tar** to request verification of each transaction.

Conflicts During Extraction

When **tar** is extracting files from an archive into a file system, it can encounter conflicts in file type between existing files and files in the archive. In such cases, **tar** attempts to rename the existing file and then retries the extraction. Files are renamed by prefixing a dot and appending a suffix of the following form:

```
,tar-03-Sep-1989-10:42:19
```

In other words, the suffix includes the tag **,tar-** followed by the date and time of the extraction. All files renamed in a single run of **tar** have the same date tag. Because the new file names begin with a dot, they do not appear in directory listings. Take care that old renamed files do not clutter the disk.

Conflicts arise when one of the following is true:

- The file in the archive is a directory, and the existing file is not.
- The file in the archive is a hard or symbolic link, and the existing file is simply present.
- The file in the archive is a regular file, and the existing file is not.
- The file in the archive is a regular file, and the existing file is an executing program.

If the **R** flag is present in the **tar** command string, renames are not attempted, and messages to that effect are printed should conflicts arise.

Examples

To collect all files in a subdirectory called dir into an archive called archive:

```
tar cf archive dir
```

To print the contents of the archive:

```
tar tvf archive
```

To extract the files from the archive and create a duplicate of dir in the subdirectory backup:

```
cd backup tar xf ../archive
```

To copy the contents of one directory to another:

```
cd fromdir tar cf - | (cd destdir; tar xf -)
```

To create an archive of /backup on a tape mounted on drive /dev/rsd1:

```
cd /backup tar cvf /dev/rsd1.
```

To extract an archive from a tape mounted on drive /dev/rsd1 and place it in the root directory:

```
cd / tar xvf /dev/rsd1.
```

telnet - user interface to the Telnet protocol

Synopsis

```
telnet [host [port]]
```

Description

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

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

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

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

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

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

Commands

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

open *host* [*port*]	Open a connection to the named host. If no port number is specified, **telnet** attempts to contact a Telnet server at the default port. The host specification may be either a host name or an Internet address specified as four dot-separated decimal octets.
close	Close a Telnet session and return to command mode.
quit	Close any open Telnet session and exit **telnet**. An end of file (in command mode) also closes a session and exits.
z	Suspend **telnet**. This command enters a sub-shell. The user may return to **telnet** by exiting the subshell.
mode *type*	*Type* is either line (for line-by-line mode) or character (for character-at-a-time mode). The remote host is asked for permission to go into the requested mode. If the remote host is capable of entering that mode, the requested mode is entered.
status	Show the current status of **telnet**. This includes the peer one is connected to, as well as the current mode.

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

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

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

- **escape** Sends the current **telnet** escape character (initially **^]**).
- **synch** Sends the Telnet SYNCH sequence. This sequence causes the remote system to discard all previously typed (but not yet read) input. This sequence is sent as TCP urgent data (and may not work if the remote system is a 4.2 BSD system - if it does not work, a lowercase **r** may be echoed on the terminal).
- **brk** Sends the Telnet BRK (Break) sequence, which may have significance to the remote system.
- **ip** Sends the Telnet IP (Interrupt Process) sequence, which should cause the remote system to abort the currently running process.
- **ao** Sends the Telnet AO (Abort Output) sequence, which should cause the remote system to flush all output from the remote system to the user's terminal.
- **ayt** Sends the Telnet AYT (Are You There) sequence, to which the remote system may choose to respond.
- **ec** Sends the Telnet EC (Erase Character) sequence, which should cause the remote system to erase the last character entered.
- **el** Sends the Telnet EL (Erase Line) sequence, which should cause the remote system to erase the line currently being entered.
- **ga** Sends the Telnet GA (Go Ahead) sequence, which likely has no significance to the remote system.
- **nop** Sends the Telnet NOP (No OPeration) sequence.
- **?** Prints out help information for the **send** command.

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

- **echo** This is the value (initially **^E**) which, when in line-by-line mode, toggles between doing local echoing of entered characters (for normal processing), and suppressing echoing of entered characters (for entering, say, a password).
- **escape** This is the **telnet** escape character (initially **^[**) which causes entry into **telnet** command mode (when connected to a remote system).

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

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

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

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

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

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

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

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

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

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

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

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

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

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

? Displays the legal toggle commands.

Acknowledgments

This section was developed by the University of California, Berkeley. This LynxOS component is available only as a part of the Lynx TCP/IP package.

touch - change the modify date of files

Synopsis

`touch [-cf]` *file* `...`

Description

The **touch** command sets the modification date of a file to the current date. If the file exists, the date is changed by reading a character from it and then writing it back. If the file does not exist, it is created.

Options

-c If the file does not exist, do not create it.

-f Perform a touch even if read and write permissions are denied on the file.