Telnet

%> **psswrd**

%> **ps aux** BSD based UNIX

%> **ps –el** System V

Lists tasks currently running

%> **ls -la** Extended directory listing

%> **setenv TERM** xterm

%> **telnet**

%> **?**

To enter command mode: table to right

%> **file** *filenam*e Lists type of file (ASCII or Binary)

%> **ls -1c** output in single column

%> **ls -t** output w/ recently modified first

%> **ls -rt** output with reverse time order

%> **ls -lR** (recursive listing)

%> **ls -x** output in row

%> **ls -m** output comma-separated listing

%> **ls -a** [to list files beginning with a dot]

telnet> **?**

Commands may be abbreviated. Commands are:

close	close current connection
display	display operating parameters
mode	try to enter line-by-line or character-at-a-time mode
quit	exit telnet
send	transmit special characters ('send ?' for more)
set	set operating parameters ('set ?' for more)
status	print status information -? will give Escape character
toggle	toggle operating parameters ('toggle ?' for more)
z	suspend telnet
?	print help information

Ftp

ftp> **get** *source-file destination-file*

ftp> **put** *source-file destination-file*

If incorrect login, can reaccess login by:

ftp> **user** *user login*

ftp> **dir** *director-name local file-name* (Only if wanted to put output into local file-name)

ftp> **ls** *directory-name local file-name*

ftp> **ls** (simplified listing)

ftp> **dir** (expanded listing)

ftp> **^S or ^Q** (to suspend get or put apps)

ftp> **^C** (cancels transfer)

(get's screen's worth of data)

ftp> **get** *source-file* **"|more"**

ftp> **get** *source-file* **"|grep** *string* **"**

(only prints files with characters *string* in them)

ftp> **ls * "|more"**

<u>Using wildcards</u>

*	[replaces any number of characters in file/directory name]
?	[replaces any single characters in a file/directory name]
[]	[matches a choice of charcters in a file/directory name]

File	**Mode**

```
ftp> lcd             On local directories
ftp> lcd/x
ftp> lcd..
ftp> cd directory    (On remote directory)
ftp> pwd             (lists current working directory)
ftp> binary
200 Type set to I
ftp> ascii
200 Type set to A
```

To transfer more than one files at a time

```
ftp> mput list of files
ftp> mget list of files
```

More Telnet

```
%> cat filename
%> cat filename_1 filename_n | more
%> cat filename_n > output_filename
%> cat file1 file2 >!output_filename
%> cat > new (save text by ^D )

%> man cat    [term describing cat command]
%> man -k         [search for terms that match the
%> apropos regex [a/ regular expression]

%> more filename
%> /string   [search for text string]
%> v         [shift from more to the vi editor]
%> :n        [shifts buffer to next file]
%> :p        [shifts buffer to previous file]

%> chmod   [to change file permissions]
                   u         g         w
     d            rwx       rwx       rwx
                 4+2+1     4+2+1     4+2+1      777
%> chmod mode filename
```

File	**Mode**
Text file	ASCII, by definition
Spreadsheet	Probably binary
Database file	Probably binary, possibly ASCII
Word processor file	Probably binary, possibly ASCII
Program source code	ASCII
Electronic mail messages	ASCII
UNIX shell archive	ASCII
UNIX tar file	Binary
Backup file	Binary
Compressed file	Binary
Uuencoded or binhexed* file	ASCII
Executable file	Binary, but see below
PostScript (laser printer) file	ASCII
Hypertext (HTML) doc	ASCII
Picture files (GIF, JPEG, MPEG)	Binary

* **uuencode** is a UNIX utility that we will mention in Chapter 7, *Electronic Mail.* The UNIX UUCP utilities use it to encode binary files in an all-ASCII representation, which makes them easier to transfer correctly. **BinHex** is a similar utility used on DOS and Windows systems.

Table 6-2: Common Compression Programs

Compression Program	**Decompression Program**	**File Suffix**	**Typical Filename**
compress	uncompress	*.Z*	*rfc1118.txt.Z*
gzip	gunzip	*.z or .gz*	*textfile.gz*
pack	unpack	*.z*	*textfile.z*
Stuffit	unsit	*.Sit*	*program.Sit*
PackIt	unpit	*.pit*	*report.pit*
PKZIP	unzip41	*.ZIP*	*package.ZIP*
zoo210	zoo210	*.zoo*	*picture.zoo*

```
%> mv old_file or dir new file_dir
%> mv -i   [option to prompt for confirmation before cmd execution]
```

For {} operator

```
%> mv 94h10.seq 94H10.seq ~ mv 94{h,H}10.seq
%> cp 94H10.seq{,.bak}  [to make a copy of 94H10.seq to 94H10.seq.bak]
%> mkdir {chr}{1,2,3,4,5}  [to make chr1, chr2, chr3, chr4, chr5]
%> mkdir {humchr, ricechr}{1,2}   [to make humchr1, humchr2, ricechr1, ricechr2]
```

```
%> rm          [removes files]
%> rmdir       [removes directories only if they are empty]
%> rm -r       [removes directories and all directory content]
```

```
%> find path operators
options: -name pattern  [finds file matching given pattern]
%> find . -name "[a-zA-Z]*.seq"  [find file in path . (pwd) containing a-zA-Z (wildcards in quotations)]
 -perm mode      [find files with the given octal access mode]
 -user name      [find files belonging to a user]
 -group name     [find files belonging to a group]
 -atime n        [find files accessed n days ago]
 -atime+n        [find files accessed over n days ago]
 -atime -n       [find files accessed less than n days ago]
 -mtime n        [find files modified n days ago]
 -newer file     [find files modified more recently than 'file']
 -print          [print out the matched file]
```

To combine files found by the find command and execute another function on the found files:

```
%> find . -name 'hum_chr*' -exec rm -r {} \;      [{} specifies same path and \; specifies an exec command which must end in \;]
%> find . -name 'hum_chr*' -print -exec rm -r {} \;   [to print out the matched directory before deleting it]
```

```
%> wc options filename
options are -l     [counts number of lines]
             -w   [counts number of words]
             -c    [counts number of characters]
```

```
%> uniq -c -n filename       [deletes multiple immediate adjacent lines of identical text from within a file and outputs a unique list]
                             [-c lists the number of ocurrences of all the repeat elements]
%> sort options filename      [sorts by default in ascending order]
list of options:   -r   [do a reverse sort, that is, descending order]
                   -t   [specify a field separator; defaut: tabs or spaces]
                   -f   [ignore case of a field]
                   -M [order by Month]
                   -b  [ignore leading spaces]
%> sort sequence.ids > sequence.sorted     [example of output redirect using sort]
%> sort filename | uniq -c     [piping sort command with the uniq -c to display in ascending order without redundancy & # of occurrences]
```

Basic **awk** (Aho, Weinverger and Kernighan)

```
%> awk '{print $1}' sequence.sorted  [invokes awk to print column 1 of the specified file]
%> awk '$5 > 1000 {print}'               [prints all columns in a file whereever column 5 is > 1000]
%> awk '$5 > 1000 {print "Peptide:" $1, "Score:" $5}'   [adds lables in the print command in double quotations]
```

GREP Utility

```
%> grep keyword filename            [keyword can be arbitrary partial strings]
%> grep -l keyword filename         [ -i feature makes it case insensitive]
%> grep -v keyword filename         [the -v option lists all relevant files not containing the keyword]
%> grep -n keyword filename         [the -n option allows the output to be numbered]
%> grep -c keyword filename         [the -c option counts the number of occurences]
%> grep -l gi*.seq                  [the -l option, lists files that contain the specified pattern]
%> grep -w keyword filename         [the -w option, searches for whole words instead of strings that may be part of words]

%> grep "^P" filename               [the ^ option lists all files that START with the letter P]

%> ls | grep keyword                [grep can be piped with other commands]
%> find . | grep -v keyword         [grep is piped with find command in the current dir & looks for terms not containing keyword]

%> egrep                            [extended grep; to search with extended regexs]
%> egrep "keyword_1 | keyword_2" filename           [egrep has the OR operator (|), need to have quotation marks]
%> egrep -f keywords filename       [egrep can search for multiple keywords using the -f operator]
```

Process Control

```
%> ps                             [lists all the process running in users terminal]
%> ^-c                            [to stop a process]
%> ^-z                            [to suspend a process]
%> kill PID                       [kills a command]
%> ps -all | grep keyword           [ps can be linked with a pipe to another command]
```

Tape archiving and Compressing

```
%> tar -cvf new_tarfile filelist_to_tar     [to transfer several files or dir between locations in binary format]
                          -c               [creat a tar format file]
                          -v               [provide a verbose output]
                          -f               [filename]
                          new_tarfile       [name of archived file]
                          filelist_to_tar   [list of files to archive]

%> tar -uvf tarfile files_to_be_added       [to add a file to a currently existing tarfile]
%> tar -tvf tarfile                        [to get a list of files in the archive using the -t option]
%> tar -xvf tarfile                        [to extract files from tarfile]

%> gzip tarfile.tar                        [to compress tarfile.tar using GNU zip]
%> gunzip tarfile.tar.gz                    [to uncompress tarfile.tar.gz using GNU zip]

%> compress tarfile.tar                    [to compress tarfile.tar using UNIX Compress]
%> uncompress tarfile.tar.Z                 [to uncompress tarfile.tar.Z using UNIX Uncompress]
```