22. Unix Command Reference

Basic Commands

Login Procedure

Logging in

1	System name	Request:	Enter your system name. The system name is the name of your particular Unix system.
2	Login name	login:	Enter you login name, the name given to you for your Unix account.
3	Password	Password:	Enter your Password. It will not show up on the screen.
4	Terminal name	TERM =	Enter you Terminal name. Your system has assigned specific name for the type of terminal you are using.

Logging Out

`exit`	Log out from the system if in the Bourne or Korn shell
`logout`	Log out from the system if in the C-shell
Ctrl-d	Log out from the system in any shell.

apropos

`apropos` *keyword* — Access the on-line manual using a keyword instead of a command name. apropos is a C-shell command.

finger

`finger` — Display more information about users. The finger command with a user's login name or real name as a argument will list specific information about that user.

```
$ finger name
```

options

`-m` Search only for login names
`-s` Display information in short format.
`-l` Display information in long format.

help

`help` *topic*	Find out more detailed information about different features of the Unix system. Enter help with no arguments for a list of topics

learn

`learn`	Use on-line tutorials to learn specific features of the system.

locate

`locate` *keyword*	Access the on-line manual using a keyword instead of a command name. locate is a Bourne and Korn shell command.

man

`man` *command-name*	Access the on-line manual and display information about a Unix command.
`man -k` *topic*	Display commands closest to topic `man -k printer`

passwd

`passwd` Change your password. You will first be prompted for your old password, then your new password. You will then be prompted again for you new password. The passwords never appears on your screen. Choose one at least seven characters long including at least one number.

```
$ passwd
Old password:
New Password:
Retype new password:
$
```

whatis

`whatis` *command*	List a brief description of a command

who

`who` List users currently logged into the system.

options

`-q` Displays only the number of users logged in.
`-H` Display headers for each column in user list.
`-u` Displays complete information about users.

File and Directory Commands

cancel

`cancel` Remove a printing job from the printing queue.

cat

`cat` Display a file. It can take file names for its arguments. It outputs the contents of those files directly to the standard output, which by default is directed to the screen.

```
$ cat filenames
```

cd

`cd` *directory-name* Change to that directory making it the current working directory. cd without a directory name changes back to the home directory.

```
$ cd reports
$ cd
```

`..` Reference the parent directory of the current working directory. The .. represents the path name of that parent directory.

```
$ cd ..
$ cp today ..
```

directory-name/filename A slash is used to separate a directory name and a file name. This is very useful for copying and moving files from the home directory to a lower directory, while renaming them.

```
$ cp today reports/monday
$ mv today reports/tuesday
```

cp

`cp` Copy a file. cp takes two arguments: the original file and the name of the new copy.

```
$ cp today monday
```

options

`-i` Check for overwrite condition.

`cp` *filename filename* Copy a file. `cp` takes two arguments: the original file and the name of the new copy. You can use path names for the files in order to copy across directories.

```
$ cp mydata stories/rumors
```

`cp -r` *dirname dirname* Copy a subdirectory from one directory to another. The copied directory will include all its own subdirectories.

```
$ cp -r stories/today oldstories
```

file

`file` Examines the first few lines of a file to determine a classification.

`-f` *filename* Read the list of file names to be examined from a file.

find

`find` Search directories for files based on a search criteria. Find has several options that specify the type of criteria and an actions to be taken.

options

`-name` *pattern* Search for files with the pattern in the name.

`-group` *name* Search for files belonging to this group name.

`-size` *numc* Search for files with the size num in blocks. If c is added after num, then the size in bytes (characters) is searched for.

`-mtime` *num* Search for files last modified *num* days ago.

`-newer` *pattern* Search for files that were modified after the one matched by pattern.

`-print` Output the result of the search to the standard output. The result is usually a list of file names, including their full path names.

`-type` *filetype* Search for files with the specified file type.

- `b` Block device file.
- `c` Character device file
- `d` Directory file.
- `f` Ordinary (regular) file.
- `p` Named Pipes (fifo).
- `l` Symbolic links.

lp

`lp` Print a file. Sends a file to the line printer. A list of files may be used. lp is an AT&T and SVR4 command.

```
$ lp mydata preface
```

options

`-P`*printer-name* Select a specific printer.

lpr

`lpr` Print files on BSD and other versions of Unix.

```
$ lpr mydata preface
```

options

`-P`*printer-name* Select a specific printer.

lpq

`lpq` List the print queue for printing jobs (BSD).

lprm

`lprm` Remove a printing job from the printing queue (BSD).

ln

`ln` *filename filename* Create added names for files. They are referred to as links. A link can be created in one directory that references a file in another directory.

```
$ ln newfacts stories/rumors
```

lpstat

`lpstat` List the print queue for printing jobs.

ls

`ls` List file and directory names.

```
$ ls
mydata preface reports
```

`ls -l` *filename*	List a file name with its permissions displayed.
`ls -ld` *directory*	List a directory name with its permissions displayed.
`ls -l`	List all files in a directory with its permissions displayed.

`ls -F`	List directory name with a preceding slash.
	`$ ls -F`
	`newsflash /stories/stories`

`ls -R`	List working directory as well as all subdirectories.

more

`more` Displays a file screen by screen. It can take a file names for its arguments. It outputs the contents of those files to the screen, one screen at a time. This filter is in BSD and Sytem V release 4.

`$ more` *filenames*

options

+*num*	Displaying the file beginning at page num

commands

num`f`	Skip forward num number of screens.
num`b`	Skip backward num number of screens.
`d`	Display half a screen.
`h`	List all more commands.
`q`	Quit more utility.

mkdir

`mkdir` Create a directory.

```
$ mkdir reports
```

mv

`mv` Move (rename) a file. mv takes two arguments: the current file name and the new name of the file.

```
$ mv today monday
```

options

`-i` Check for overwrite condition.

`mv` *filename filename* Move (rename) a file. `mv` takes two arguments: the first is the file to be moved. The second argument can be the new file name or the path name of a directory. If it is the name of a directory, then the file is literally moved to that directory, changing the file's path name.

```
$ mv today /home/larisa/stories
```

`mv` *dirname dirname* Move directories. In this case, the first and last arguments are directories.

```
$ mv stories/today oldstories
```

od

`od` Prints out the contents of a file byte by byte in either octal, character, decimal, or hexadecimal. octal is the default.

`-c` Output character form of byte values. Non-printing characters have a corresponding character representation.

`-d` Output decimal form of bytes values.

`-x` Output hexadecimal form of bytes values.

`-o` Output octal form of bytes values.

pg

`pg` Displays a file screen by screen. It can take file names for its arguments. It outputs the contents of those files to the screen, one screen at a time. This filter is in early versions of System V and in SVR4.

`$ pg` *filenames*

options

+*num* Displaying the file beginning at page num.

commands

num spacebar

num enter Display page num. If no number is entered, then the next page is displayed.

`b` Display previous page.

`d` Scroll the screen down by one-half page.

`h` Display a list of all the pg commands

`q` Quit the pg utility.

pwd

`pwd` Display the path name of the working directory.

```
$ pwd
/home/larisa/stories
```

pathnames

directory-name/filename — A slash is used in path names to separate directory names. In the case of path names for files, a slash separated the preceding directory names from the file name.

```
$ cd /home/larisa/stories
$ cat /home/larisa/stories/mydata
```

`..` — The double dot references the parent directory. You can use it as an argument or as part of a path name.

```
$ cd ..
$ mv ../newsflash oldletters
```

`.` — The single dot references the working directory. You can use it as an argument or as part of a path name.

```
$ ls .
$ mv ../rumors .
```

~/pathname — In the BASH, Korn, C-shell, and Z-shell, the tilde is a metacharacter that represents the path name for the home directory. Useful when you need to use an absolute path name for a file or directory.

```
$ cp rumors ~/facts
$ mv scoops ~/newsflash
```

rm

`rm` — Remove (erase) a file. rm can take any number of file names as its arguments.

```
$ rm today monday sunday
```

options

`-i` Check for overwrite condition.

`-r` Remove a directory with its files.

`rm` *filenames* — Remove (erase) a file. `rm` can take any number of file names as its arguments. The `rm` command literally removes links to a file, file names. If a file has more than one link, you need to remove all of them in order to finally erase a file.

```
$ rm storm newsflash weekend
```

rmdir

`rmdir` — Erase a directory.

```
$ rmdir letters
```

BSD Standard System Files

/	The root directory, beginning of the file system structure.
/usr	This directory contains files and subdirectories used by the system, including user home directories. However, the placement of home directories may vary from system to system. On some systems, home directories many be placed in subdirectories of **usr** or in different system directories. **/usr** may also contain a sub directory called bin that contains commands and utilities.
/bin	This directory contains the most widely used commands as well as those needed to boot the system.
/usr/bin	Like /bin, this directory also contains commands and utilities, but they are usually those used less frequently.
/dev	This directory contains interface files for devices like terminals and printers.
/etc	This directory contains commands and files used for system administration
/tmp	This directory holds any temporary files the commands may need to generate such as work buffers for editors.

SVR4 Standard System Files

/	The root directory, beginning of the file system structure.
/home	This directory usually contains user home directories. However, many systems may use a different directory for their home directories.
/usr	Holds those files and commands used by the system. This directory breaks down into several subdirectories.
/usr/bin	Holds all the standard commands and utility programs, including the different shells such as /bin/sh for the Bourne shell.
/usr/lib	Holds libraries and daemons used for system administration operations.
/usr/include	Holds .h header files for the C compiler.
/usr/ucb	Holds BSD compatibility files.
/usr/games	Holds games that you can run on your system.
/usr/share	Holds files that can be shared by different systems.
/usr/share/man	Holds the on-line manual man files.
/usr/sadmin/skel	Holds the default files such as profile that are automatically installed in a user's home directory when it is created.

`/sbin`	Holds system administration commands including those used to startup the system.
`/var`	Holds files that vary such as mailbox files. There are several subdirectories used for specified tasks.
`/var/spool`	Holds spooled files such as those generated for printing jobs and network transfers.
`/var/mail`	Holds user's mail files.
`/opt`	Location of add on application packages.
`/boot`	Location of files used to boot your system.
`/dev`	Holds file interfaces for devices such as the terminal and printer.
`/dev/term`	Holds terminal device files.
`/dev/dsk`	Holds disk device files.
`/etc`	Holds system configuration files and any other system files.
`/etc/cron.d`	Holds files to control `cron` operations.
`/etc/init.d`	Holds files to control `cron` operations.
`/etc/lp`	Holds printer configuration files
`/etc/mail`	Holds mail configuration files
`/etc/rc.d`	Holds system initialization files used when changing states.
`/tmp`	Holds temporary files.
`/mnt`	Usually used to temporarily mount file systems such as those on floppy disks or CD ROMs.

Permissions and Archives

Absolute Permissions

Octal and Binary Digits

Octal	Binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Quick Calculation

Permission	Number	Binary
r	4	100
w	2	010
x	1	001

chmond

`chmod` Change the permission of a file or directory. The + and - signs add or remove permissions.

`+` Add a permission.

`-` Remove a permission.

File and Directory Permissions

`r` Read permission for a file or directory. A file can be displayed or printed. A directory can have the list of its files displayed.

`w` Write permission for a file or directory. A file can be edited or erased. A directory can be removed.

`x` Execute permission for a file or directory. If the file is a shell script, it can be executed as a program. A directory can be changed to and entered.

User Categories for Permissions

`u` Permissions set for the user that created and owns the file or directory.

`g` Permissions set for group access to a file or directory.

`o` Permissions set for access to a file or directory by all other users on the system.

`a` Permissions set for access by the user, group, and all other users.

`s` Set User ID and Group ID permission, program owned by owners and group.

`t` Set sticky bit permission, program remains in memory.

chgrp

`chgrp` *groupname filename* Change the group for a file or files.

chown

`chown` *username filename* Change the owner of a file or files.

ls

`ls -l` *filename*	List a file name with its permissions displayed.
`ls -ld` *directory*	List a directory name with its permissions displayed.
`ls -l`	List all files in a directory with its permissions displayed.

tar

`tar` *options files* Back up files to tape, device, or archive file.

tar options

`c`	Create a new archive.
`r`	Append files to an archive.
`u`	Update an archive with new and changed files. Add only those files that have been modified since they were archived or files that are not already present in the archive.
`w`	Wait for a confirmation from the user before archiving each file. Allows you to selectively update an archive.
`x`	Extracts files from an archive.
`m`	When extracting a file from an archive, do not give it a new time stamp.
`f` *archive-name*	Save the tape archive to the file archive-name instead of to the default tape device. The archive-name can be either a file or another device such as a tape or disk. The default device is held in /etc/default/tar file.
`v`	Display each file name as it is archived.

cpio

`cpio`

The cpio filter copies files to an archive and extracts files from an archive. The cpio filter has two modes of operation: one using the -o option to copy files to an archive, and the other using the -i option to extract files from an archive. When copying files to an archive you need to first generate the list of file name using a command such as ls or find.

```
$ generated-filenames  | cpio -o > archive-file
$ cpio -i filenames < archive-file
```

options

o	out	Creates archived output that can be redirected to a file, making it an archive file. The operation effectively creates a backup for designated files. With this option, cpio takes as its input a list of file names. These files will have their contents read and copied into the standard output in an archived format. File names can be generated with the ls or find commands and piped to cpio. To archive directories you must use the find command.
		
		`$ ls | cpio -o > progarch`
		`$ find -name * | cpio -o > progarch`

i	in	Extracts files from an archive. The archive is read from the standard input which usually has been redirected from an archive file. The files to be extracted have their names listed as argument to the cpio filter. If no file names are listed then all files are extracted. Extracted files are copied out of the archive. A copy of each is written to the current directory unless it was archived with a path name. In that case, the file is saved in the directory referenced by its path name.
		
		`$ cpio -i < progarch`
		`$ cpio -i main.c < progarch`

d	directory	Used with the i option, it directs cpio to extract directories, creating new ones if needed. `$ cpio -id < progarch`
u	unconditional	Used with the i option it directs cpio to copy older files over their corresponding newer ones that already exist in your directory. By default, cpio will not overwrite files newer than those extracted from the archive. `$ cpio -iu < progarch`

Shell Operations

Command Line and Metacharacters

Command execution

Enter	Execute a command line.
;	Separate commands on the same command line.
command\ *opts args*	Continue entering a command on the next line by entering a backslash before Enter
`` `command` ``	Execute a command.
backspace	Erase the previous character
Ctrl-h	Erase the previous character
Ctrl-u	Erase the command line and start over.
Ctrl-c	Interrupt and stop a command execution.

Metacharacters for file name generation.

*	Match any set of characters.
?	Match any single characters.
[]	Match a class of possible characters.
\	Quote the following character. Used to quote metacharacters. Allow you to use the character literally.

Redirection and Pipes

Redirection

command > *filename*	Redirect the standard output to a file or device, creating the file if it does not exist and overwriting the file if it does exist.
command < *filename*	Redirect the standard input from a file or device to a program.
command >> *filename*	Redirect the standard output to a file or device, appending the output to the end of the file.
command >! *filename*	In the C-shell and Korn shell, force the overwriting of a file if it already exists. This overrides the noclobber option.
command 2> *filename*	Redirect the standard error to a file or device in the Bourne shell.
command 2>> *filename*	Redirect and append the standard error to a file or device in the Bourne shell.
command 2>&1	Redirect the standard error to the standard output in the Bourne shell.
command >& *filename*	Redirect the standard error to a file or device in the C-shell.

Pipes

command | *command*	Pipe the standard output of one command as input for another command.
command |& *command*	Pipe the standard error as input to another command in the C-shell.

Jobs and Background

&

Background Jobs and At Jobs

&	Execute a command in the background

fg

fg %*jobnum*	Bring a command from the background to the foreground or resume an interrupted program.

bg

`bg`	Place a command in the foreground into the background
Ctrl-z	Interrupt and stop the currently running program. The program remains stopped and waiting in the background for you resume it.

notify

`notify %`*jobnum*	Notify you when a job ends.

kill

`kill %`*jobnum*	Cancel a job running in the background.
`kill` *proccessnum*	Cancel a job running in the background.

jobs

`jobs`	List all background jobs. The jobs command is not available in the Bourne shell, unless it is using the BASH shell.

ps

`ps`	List all currently running processes including background jobs.

at

`at` *time* *date*

Execute commands at a specified time and date. The time can be entered with hours and minutes and qualified as am or pm.

hour:minutes am pm

The date is specified as a day of the month or day of the week. The month can be represented by a three letter abbreviation: Jan, Feb, etc.

month *day*

Days of the week are specified by their names.

monday tuesday wednesday

Keywords can be used to specify the date and time.

```
am
pm
now
noon
midnight
today
tomorrow
```

You can increment from a date or time by a time segment using the + operator. A number after the + operator specifies how many time segments.

date +*num* *time-segment*

Time segments

hours minutes days weeks months years

The next keyword increments by a time segment from the current date or time:

`next` *time-segment*

`next` *week*

options

`-l` *jobnum* List current at jobs.

`-r` *jobnum* Cancel a job.

`-m` *jobnum* Be notified by mail when job finishes.

Korn Shell

Korn Shell VI Command Line Editing Commands

Cursor Movement

`l` Move one character to the right.

`h` Move one character to the left.

`w` Move one word to the right.

`b` Move one word to the left.

`W` Move one space-delimited word to the right.

`B` Move one space-delimited word to the left.

`e` Move to end of word to the right.

`E` Move to end of space-delimited word to the left.

`0` Move to beginning of the line.

`$` Move to end of the line.

`^` Move to first character on the line.

Input Commands

`i` Insert before the cursor. Input until you hit escape.

`a` Append after the cursor. Input until you hit escape.

`I` Insert at beginning of the line Input until you hit escape.

`A` Append at the end of the line. Input until you hit escape.

Delete Commands

`x` Delete the character the cursor is on.

`dw` Delete the word the cursor is on.

`dd` Delete the entire line, effectively deleting the command

`D` Delete the rest of the line.

Replace and Change Commands

`r` Replace the character the cursor is on.

`R` Overwrite characters. Input until you hit escape.

`cw` Change the word the cursor is on. Input until you hit escape.

`C` Change the rest of the line. Input until you hit escape.

Korn Shell History Commands

`history`	List recent history events. Same as `fc -l` command.	
`fc`	Korn shell history command. Without options it edits and executes an event reference.	
`fc` *event-reference*	Edits an event with the standard editor and then executes it.	
	options	
	`-l`	List recent history events. Same as history command.
	`-e` *editor event-reference*	Invoke a specified editor to edit a specific event.
	`r` *event-reference*	Re-execute the specified history event, a previous command.

Korn Shell Vi History Editing Commands

escape	Enter the Command Line Vi Editor.
`j` and `+`	Move down to next command in history list.
`k` and `-`	Move up to previous command in history list.
`G`	Move to beginning command in history list.
num`G`	Move to command num in history list.
/pattern	Search back in history list for command containing pattern.
`?`*pattern*	Search forward in history list for command containing pattern.
`n`	Repeat previous search.
num`v`	Invoke vi editor to edit command num in the history list.

Korn Shell Features

Korn shell features are turned on and off with the `set` command. `-o` sets a feature on and `+o` turns it off.

```
$ set  -+o feature
$ set -o noclobber    set noclobber on
$ set +o noclobber    set noclobber off
```

`ignoreeof`	Disable Ctrl-d log out.
`noclobber`	Do not overwrite files through redirection.
`noglob`	Disable metacharacters used for file name expansion: `*`, `?`, `~`, and `[]`

C-Shell

C-shell History Commands

Event References

`!`*event num*	Reference an event with event number.
`!`*characters*	Reference an event with beginning characters.
`!?`*pattern*`?`	Reference an event with a pattern in the event
`!-`*event num*	Reference an event with an offset from the first event.
`!`*num*`-`*num*	Reference a range of events.

Event Word References

!*event num*:*word num*	Reference a particular word in an event.
!*event num*:^	Reference first argument (second word) in an event.
!*event num*:$	Reference last argument in an event.
!*event num*:^-$	Reference all arguments in an event.
!*event num*:*	Reference all arguments in an event.

Event Editing Substitutions

!*event num*:s/*pattern*/*newtext*/	Edit an event with a pattern substitution. Reference a particular word in an event.
!*event num*:sg/*pattern*/*newtext*/	Perform a global substitution on all instances of a pattern in the event.
!*event num*:s/*pattern*/*newtext*/p	Suppress execution of the edited event.

C-Shell Features

The C-shell features are turned on and off with the `set` and `unset` commands.

```
$ set feature-variable
$ set noclobber       set noclobber on
$ unset noclobber     set noclobber off
```

`echo`	Display each command before executing it.
`ignoreeof`	Disable Ctrl-d log out.
`noclobber`	Do not overwrite files through redirection.
`noglob`	Disable metacharacters used for file name expansion: `*`, `?`, `~`, and `[]`.
`filec`	Enable file name completion.
`notify`	Notify user immediately when background job is completed.
`verbose`	Display command after a history command reference.

Bash Shell

BASH Shell Command Line Editing Commands

Ctrl-f or right-arrow	Move one character to the right.
Ctrl-b or left-arrow	Move one character to the left.
Esc f	Move one word to the right.
Esc b	Move one word to the left.
Ctrl-a	Move to the beginning of the line.
Ctrl-e	Move to the end of the line.

To Insert text move cursor and then type.

delete	Delete the character the cursor is on.
Ctrl-d	Delete the character after the cursor.
backspace or Ctrl-h	Delete the character before the cursor.
Ctrl-k	Delete the remainder of the line.

BASH Shell History Commands

`history`	List recent history events.
Ctrl-n or down-arrow	Move down to next command in history list.
Ctrl-p or up-arrow	Move up to previous command in history list.
`Esc <`	Move to beginning command in history list.
`Esc >`	Move to end command in history list.
`Esc tab`	History event matching and completion.
`!`*event-num*	Reference an event with event number.
`!`*num*`-`*num*	Reference a range of events.
`!-`*offset*	Reference an event with offset from first event.
`!`*characters*	Reference an event with beginning characters.
`!?`*pattern*`?`	Search back in history list for command containing pattern.
`fc` *event-reference*	Edit an event with the standard editor and then execute it. **options** `-l` List recent history events. Same as history command. `-e` *editor event-reference* Invoke a specified editor to edit a specific event.

BASH Shell Features

BASH shell features are turned on and off with the `set` command. -o sets a feature on and +o turns it off.

```
$ set - +o feature
$ set -o noclobber    set noclobber on
$ set +o noclobber    set noclobber off
```

`ignoreeof` Disable Ctrl-d log out.

`noclobber` Do not overwrite files through redirection.

`noglob` Disable metacharacters used for file name expansion: `*`, `?`, `~`, and `[]`

9. Z-shell

Z-shell History Commands

`!`	Start a history substitution, except when followed by a blank, newline, =, or (.
`!!`	Refer to the previous command. By itself, repeats the previous command.
`!`*num*	Refer to command-line num.

!-*num*	Refer to the current command line minus num.
!*str*	Refer to the most recent command starting with str.
!?*str*[?]	Refer to the most recent command containing str.
!#	Refer to the current command line typed so far.
!{...}	Insulate a history reference from adjacent characters (if necessary).

Word Designators

0	The first input word (command).
num	The num'th argument.
^	The first argument, that is, 1.
$	The last argument.
%	The word matched by (the most recent) ?*str* search.
str-*str*	A range of words; -*str* abbreviates 0-str.
*	All the arguments, or a null value if there is just one word in the event.
*str**	Abbreviates *str*-$.
str-	Like *str** but omitting word $.

Shell Special Variables

Bourne	**Korn**	**C-shell**	
System Determined Special Variables			
HOME	HOME	home	Path name for user's home directory.
LOGNAME	LOGNAME	LOGNAME	Login name.
Redefinable Special Variables			
SHELL	SHELL	shell	Path name of program of type of login shell.
PATH	PATH	path	List of path names for directories searched for executable commands.
PS1	PS1	prompt	Primary shell prompt.
PS2	PS2		Secondary shell prompt.
MAIL	MAIL	mail	Name of mail file checked by mail utility for received messages.
MAILCHECK	MAILCHECK		Interval for checking for received mail. MAILCHECK
	PWD	cwd	Current working directory.

	HISTFILE		File that holds history events.
	HISTSIZE		Number of history events recorded.
	VISUAL		Editor for the command line (default is Vi).
	FCEDIT		fc editor (default is Vi).
	COLUMNS		Number of columns displayed (line length; default is 80).
ENV	ENV	ENV	Name of the Korn shell configuration file.
		argv	Command line arguments.
		#argv	Number of command line arguments.

User Defined Special Variables

TERM	TERM	TERM	Terminal name.TERM
CDPATH	CDPATH	cdpath	Path names for directories searched by cd command for subdirectories.
EXINIT	EXINIT	EXINIT	Initialization commands for Ex/Vi editor.
MAILPATH	MAILPATH		List of mail files to be checked by mail for received messages.
		history	Turns on the history utility and sets the number of events displayed.
		savehist	Name of file that holds history events.
IFS	IFS		Internal field separator.
LINENO	LINENO		Line number of command executed.
PPID	PPID		Process ID of parent process.PPID
	AUTO_LIST	autolist	List choices for ambiguous file completion
		autologout	Automatically logout if system is idle for a specifed period of time.
	CORRECT	correct	Enable spell correction of commands

dirstack	Maintain stack of directories.
fignore	Holds list of suffixes to be ignored for file completion.
shlvl	Holds the depth of the current subshell.
tsch	Holds version number of TCSH shell you are using.
tperiod	How often periodic alias is executed.
version	Information about the TCSH shell.
watch	Users or terminals whose login you want to detect.
who	Format for displaying reports from watch.

Shell Configuration Files

Bourne Shell

.profile	Login and configuration file.

Korn Shell

.profile	Uses same login and configuration as file as Bourne shell.
.kshrc	Shell configuration file for Korn shell.

C-shell

.login	Login configuration file.
.cshrc	Shell configuration file.
.logout	Logout file.

BASH Shell

.bash_profile Login configuration file (uses .profile if this file does not exist).

.bashrc Shell configuration file.

.bash_logout Logout file.

TCSH

.login Login configuration file (same as C-shell).

.tcshrc Shell configuration file (uses .cshrc if this file does not exist).

.logout Logout file.

Z-shell

.zshenv Shell login file (first to be read).

.zprofile Shell configuration file (uses .profile if this file does not exist).

.zlogin Shell login file.

.zschrc Shell configuration file.

.zlogout Logout file.

File Filters

cat

`cat`

The cat filter can be used to display a file. It can take filenames for its arguments. It outputs the contents of those files directly to the standard output, which, by default is directed to the screen.

`$ cat` *filenames*

cmp

`cmp`

The cmp filter compares two files character by character checking for differences. It stops at the first difference it finds and outputs the character position and line number.

```
$ cmp file1 file2
```

options

`l` With this option, cmp outputs all the character positions of differing characters, as well as their respective octal values.

```
$ cmp lunch dinner
15 160 164
17 164 155
```

comm

`comm`

The comm filter compares two files line by line and outputs both files according to lines that are similar and different for each.

```
$ comm file1 file2
```

options

`1` With this option, comm suppress output for lines unique to the first file. The 1 refers to the first column.

`2` With this option, comm suppress output for lines unique to the second file. The 2 refers to the second column.

`3` With this option, `comm` suppress output for lines common both files. The 3 refers to the third column.

fgrep

`fgrep`

The `fgrep` filter can search files in the file-list for several patterns at the same time. If executes much faster than either `grep` or `egrep`, however, `fgrep` cannot interpret metacharacters. It cannot search for regular expressions.

```
$ fgrep patterns file-list
$ fgrep milk  perishables packaged
```

options

`f` *filename* With this option, `fgrep` reads its pattern list from a file called filename.

```
$ fgrep -f mypats  perishables packaged
```

grep

`grep`

The `grep` filter searches files for a pattern and lists any matched lines.

```
$ grep pattern  filenames
```

options

`i` With this option, grep ignores upper and lower case differences.

```
$ grep -i milk perishables packaged
```

`c` With this option, grep only outputs a number, the count of the lines with the pattern.

```
$ grep -c milk perishables
```

`l` With this option, grep only displays the names of the files that contain the matching pattern.

```
$ grep -l milk perishables packaged
```

`n` `grep` outputs the line number along with the line text of those lines with the matching pattern.

```
$ grep -n milk perishables packaged
```

`v` grep outputs all those lines that do not contain the matching pattern.

```
$ grep -v milk perishables packaged
```

head

`head`

The head filter displays the first few lines of a file. You can specify the number of lines. The default is 10 lines.

```
$ head filenames
```

options

-num With this option, head will display the number of lines specified by num and starting from the beginning of the file.

pr

`pr`

The `pr` filter outputs a paginated version of the input, adding headers, page numbers, and any other specified format.

```
$ pr options  filenames
```

options

+num This option instructs pr to output only from the page numbered num. Unlike other options, there is no preceding dash.

```
$ pr +6 wreport | lp
```

-num This option instructs pr to format the output into columns. The number of columns is designated by num.

```
$ pr -2 wreport | lp
```

d — This option instructs pr to double space the formatted output.

```
$ pr -d wreport | lp
```

h*name* — This option replaces the default header with name. The default header is the filename.

```
$ pr -h "Weather Report" wreport | lp
```

l*num* — This option sets the number of lines on a page.

```
$ pr -l20 wreport | lp
```

m — With this option, pr outputs several files as the same time, each in their own column. The output shows multiple columns, one for each file.

```
$ pr -m mon tues | lp
```

n — This options instructs pr to number lines. The option can be modified with either a character or a number. The character represents a character to be placed between the line number and the line text. The number represents the minimum number of spaces between the lined number and line text.

```
$ pr -n wreport | lp
$ pr -n: wreport | lp
$ pr -n3 wreport | lp
```

t — This options instructs pr to not to generate a header or trailer for the formatted output.

```
$ pr -t wreport | lp
```

w*num* — This option sets the width of a page. num is the number of character columns. The default is 72.

```
$ pr -w25 wreport | lp
```

spell

`spell`

The spell filter checks the spelling of each word in a file and outputs only the misspelled words

```
$ spell filename
```

options

+*filename* — With this option, you can specify your own user-defined dictionary of words to be searched.

```
$ spell +mydict list1
```

sort

`sort`

The sort filter outputs a sorted version of a file. The sort filter is described in detail in chapter 11.

```
$ sort filename
```

tail

`tail`

The tail filter displays the last lines in a file. You can specify the number of lines. The default is 10 lines. tail has several options for displaying the end of a file.

`$ tail filenames`

options

`-num` With this option, tail will display the number of lines specified by num and starting from the end of the file.

`+num` With this option, tail will display the rest of the text starting form page num .

`c` With this option, tail will display by characters. This option is used with either -num or +num and num will then refer to a number of characters to be displayed.

`l` With this option, tail will display by line. This option is used with either `-num` or `+num` and *num* will then refer to a number of lines to be displayed. This is the default option.

`r` With this option, tail display lines in reverse order. This option is used with either `-num` or `+num` and *num* will then refer to a number of lines to be displayed in reverse. `+1r` displays the entire file in reverse order.

tee

`tee`

The tee filter copies the standard input to a file while sending it on to the standard output. It is usually used in conjunction with another filter and allows you to save output to a file while sending the output on to another filter or utility. As with cat, if the standard output is not further redirected or piped, it will be sent its default destination, the screen. This allows you to see on the screen what is saved in the file.

```
$ filter | tee filename | utility
$ filter | tee filename
```

`a` Append to the file, do not overwrite it.

```
$ filter | tee -a filename | utility
```

wc

`wc`

The wc filter counts the number of lines, words, and characters in a file and outputs just that number.

```
$ wc filename
```

options

`c`	With this option, wc counts the number of characters in a file.
`l`	With this option, wc counts the number of lines in a file.
`w`	With this option, wc counts the numbers of words in a file.

Edit Filters

Ed Commands

Input and Change:

`a`	append	Add text after a line. End input with a period on a line of its own.
`i`	insert	Insert text before a line. End input with a period on a line of its own.
`c`	change	Change the line or set of lines. End input with a period on a line of its own.

Delete, Move, and Copy:

`d`	delete	Delete a line or set of lines.
`m`*Num*	move	Move line or set of lines by deleting them and then inserting them after line *Num*.
`t`*Num*	copy	Copy line or set of lines by copying them and then inserting the copied text after line *Num*.

Line References:

Num	Line number	A number references that line number.
Num`,` *Num*	Set of Lines	Two number separated by a comma references a set of lines.
Num`-`*Num*	Range of Lines	Two number separated by a dash references a range of lines.
`-`*Num*	Offset reference	The minus sign preceding a number offset to a line before the current line.
`+`*Num*	Offset reference	The plus sign preceding a number offset to a line after the current line..

`.`	Current Line	The period symbol references the current line, the line you are currently positioned at.
`$`	Last Line in File	The dollar sign symbol references the last line in the file.
`/`*Pattern*`/`	Pattern Ref.	A line can be located and referenced by a pattern. The slash searches forward.
`?`*Pattern*`?`	Pattern Ref.	A line can be located and referenced by a pattern. The question mark searches backward.
`g/`*Pattern*`/`	Global Pattern	A set of lines can be located and referenced by a repeated pattern reference. All line with pattern in it are referenced.
`v/`*Pattern*`/`	Global Pattern	All line without pattern in it are referenced.

Substitution Command:

`s/`*pattern*`/`*replacement*`/`	Substitution	Locate pattern on a line and substitute pattern with replacement pattern.
`s/`*pattern*`/`*replacement*`/g`	Global Substitution on a line	Substitute all instances of a pattern on a line with the replacement pattern.
Num`-`*Num* `s/`*pattern*`/`*replacement*`/`	Substitution	Perform substitutions on the range of lines specified.
`1,$ s/`*pattern*`/`*replacement*`/g`	Global Substitution on the file	Substitute all instances of a pattern in the file with the replacement pattern.

Joining and Breaking Lines:

Num`,`*Num* `j`	join	Combine lines
`s/`*pattern*`/`*pattern*`\` `/`		Break a line into two lines by inserting a return.

sed

`sed`

The `sed` filter outputs an edited form of its input. `sed` takes as an argument an editing command and a file list. The editing command is executed on input read from files in the file list. `sed` then outputs an edited version of the files. The editing commands are line editing commands similar to those used for the Ed line editor.

```
$ sed editing-command file-list
$ sed '1d' perishables
```

sed options

`n` With this option, sed does not output lines automatically. This option is usually used with the p command to output only selected lines.

```
$ sed -n '/veg/ p' perishables
```

`f` *filename* With this option, sed can read editing commands from the file filename.

```
$ sed -f myed perishables
```

sed line editing commands

You need to quote any carriage returns if you are entering more than one line.

`a`			Append text after a line.
`i`			Insert text before a line.
`c`			Change text.
`d`			Delete lines.
`p`			Print lines - output lines.
`w`			Write lines to a file.
`r`			Read lines from a file.
`q`			Quit the sed editor before all lines are processed.
`n`			Skip processing to next line.
`s/`*`pat`*`/`*`rep`*`/`			Substitute matched pattern with replacement text.
	`g`	`s/pat/rep/g`	Global substitution on a line.
	`p`	`s/pat/rep/p`	Output the modified line.
	`w`	`s/pat/rep/w` *fname*	Write the modified line to a file.

diff

`diff`

The diff filter compares two files and outputs the lines that are different as well as the editing changes needed to make the first file the same as the second file. With the `-e` option diff will output ed line editing commands that you can then use to actually make the first file the same as the second.

f1-linenum `a` *f2-line1*`,` *f2-line2* Append lines from **file2** to after f1-linenum in **file1**.

f1-line1`,` *f1-line2* `d` *f1-linenum* Delete the lines in **file1**.

f1-line1`,` *f1-line2* `c` *f2-line1*`,` *f2-line2* Replace lines in **file1** with lines in **file2**.

```
$ diff file1 file2
$ diff breakfast brunch
```

options

`b` With this option, diff ignores any trailing or duplicate blanks.

```
$ diff -b breakfast brunch
```

`c` With this option, diff outputs a context for differing lines. Three lines above and below are displayed.

```
$ diff -c breakfast brunch
```

`e` With this option, diff outputs a list of Ed editing commands that, when executed, change the first file into an exact copy of the second file. You usually redirect this output to a file and then add the `w` and `q` commands to this file. Then the file can be used as redirected input for the Ed command.

```
$ diff -e breakfast brunch > bchanges
$ echo "w\nq" >> bchanges
$ ed breakfast < bchanges
```

tr

`tr`

The tr filter outputs a version of the input in which characters in the first character list that occur in the input are replace in the output by corresponding characters in the second character list. In effect, you can replace each occurrence of a character with another character.

```
$ tr first-character-list second-character-list
$ tr "abc" "xyz"
```

metacharacters

`[]` You can use brackets to specify a range of characters.

```
$ tr "[a-z]" "[A-Z]"
```

options

`d` With this option, tr deletes any character in the character list.

```
$ tr -d "abc"
```

`c` With this option, tr replaces those characters not in the character list.

```
$ tr -c "abc" "xyz"
```

`s` With this option, tr replaces multiple instances of characters in the character list with only one corresponding replacement character.

```
$ tr -s "abc" "xyz"
```

Regular Expressions

Regular Expression Metacharacters

^	Start of a Line	References the beginning of a line.
$	End of a Line	References the end of a line.
.	Any Character	Matches on any one possible character in a pattern.
*	Repeated Chars	Matches on zero or more repeated characters in a pattern.
[]	Classes	Matches on classes of characters, a set of characters, in the pattern.

Replacement Text Metacharacters

&	Current pattern	Represents the current pattern matched.
\ (*pattern*\ )	Quoted Pattern	Segments matched pattern into fields that can be referenced in the replacement text using the backslash \ and a number. \1 references the first quoted pattern.

Extended Metacharacters: `egrep` and `awk`

pattern | *pattern*	Logical OR for searching for alternative patterns. `'milk|soup'`
(*pattern*)	Parentheses for grouping patterns. `(canned|lowfat) milk`
char+	Searches for 1 or more repetitions of the previous character. `it+i`
char?	Searches for zero or one instance of the previous character. `si?t`

egrep

The `egrep` filter searches files for the occurrence of a pattern. Like `fgrep`, it can read patterns from a file. Like `grep`, it can use regular expressions, interpreting metacharacters. However, unlike `grep`, it can also interpret extended metacharacters such as `?`, `|`, and +.

```
$ egrep pattern file-list
$ egrep milk  perishables packaged
```

Extended metacharacters

pattern | *pattern*	Logical OR for searching for alternative patterns. `$ egrep 'milk|soup'  perishables`

(*pattern*) Parentheses for grouping patterns.
```
$ egrep '(canned|lowfat) milk' perishables
```

char+ Searches for 1 or more repetitions of the previous character.
$ egrep 'it+i' perishables

char? Searches for zero or one repetition of the previous character.
```
$ egrep 'si?t' perishables
```

options

`f` *filename* With this option, `egrep` reads its pattern list from a file called filename.
```
$ egrep -f mypats  perishables
```

Data Filters

cut

`cut`

The `cut` filter copies out specified fields or columns in a file. You must always use either the `-f` option or the `-c` option with `cut`.

```
$ cut -option file-list
$ cut -f2,3 listdataD
```

`-f`*num* The `-f` option specifies what fields you want copied out of a file. Fields are numbers from 1. You can specify more than one field by separating them with a comma, or you can specify a range of fields using a dash between numbers.
-fnum1,num2 Specify fields to be cut out..
```
$ cut -f1,3 listdataD
```
-fnum1-num2 Specify a range of fields beginning with num1 and ending with num2.
```
$ cut -f2-4 listdataD
```

`-c`*num-num* The `-c` option allows you to specify columns of characters to be cut out.
```
$ cut -c20-35 listdataS
```

`-d`*delimiter-list* The `-d` option allows you to specify your own delimiter to look for in a file.
```
$ cut -d: -f2-4  listdataC
```

`-s` Ignores any lines that do not have a delimiter in them. This option can only be used with the `-f` *option*. You use it to pass over lines with no data, such as headings, titles, or empty lines.
```
$ cut -f2-4 -s listdataD
```

join

`join`

The `join` filter joins the lines of different files if the values of a specified field in each file matches.

```
$ join -option file-list
$ join -j1 2 -j2 1 foods counts
```

`-j`*filenum fieldnum*

The `-j` option specifies what fields in each file are to be compared. Each field is numbered from 1. If you are comparing the same field in each file, you need only one `-j` option and the fieldnum. If you are comparing different fields in each file, then you need a `-j` option and *filenum* as well as the fieldnum for each file. The filenum is the position of the file's name in the file-list.

`-j` *fieldnum* Compare the same field in each file. There is a space between the `-j` option and the *fieldnum*.

```
$ join -j 2 foods counts
```

`-j`*filenum fieldnum* Compare different fields in each file.

```
$ join -j1 2 -j2 1 foodtypes counts
```

`-o`*filenum*`.`*fieldnum*

The `-o` option specifies what fields in each file are to be output. The filenum and fieldnum are separated by a period. You can list several fields, separating each filenum .fieldnum combination with a space.

```
$ join -j1 2 -j2 1 -o 1.3 1.4 2.2 2.4 foodtypes counts
```

`-t`*delimiter*

The `-t` option allows you to specify your own delimiter to look for in a file. This is the same as the `-d` option in the `cut` and `paste` filters.

```
$ join -t: -j1 2 -j2 1 foodtypes counts
```

`-a`*filenum*

The -a option outputs lines whose fields from a specified file do not match, as well as matched fields from both files.. The `-a` option takes a *filenum* to specify the file from which to output unmatched lines.

```
$ join -j1 2 -j2 1 -a1 foodtypes counts
```

sort

`sort`

The `sort` filter sorts the lines it receives as input. You use it to generate a sorted version of a file. You can sort in a variety of ways such as alphabetic sorts, reverse sorts, and numeric sorts. You can sort on a given field or range of fields. The syntax for the `sort` filter is the keyword `sort` followed by any options and then a list of file names. `sort` can also receive its input from the standard input. You can pipe data into `sort` to be sorted.

```
$ sort -option file-list
$ sort -n perishables
```

basic `sort` operations

`-o` *filename* Save the output of `sort` in filename. You can use this option to safely overwrite the original input file, giving you a sorted file.

```
$ sort perishables -o perishables
```

`c` Check only to see of the file is sorted. If the file is not sorted, `sort` displays an error message. Otherwise it displays nothing.

`m` Merge previously sorted files.

`u` Output repeated line only once.

sorting data

`d` Dictionary sort - ignores any characters in the character set that are not alphabetic, numbers, or blanks. Punctuation characters and control characters are ignored.

`f` Ignore case. Lowercase characters are folded into uppercase characters.

```
$ sort -f perishables
```

`i` Ignore non-printing characters.

`M` Sort months. Fields whose values are the names of the month are sorted. The first three characters of the name are examined and changed to uppercase for sorting: JAN, FEB, JUN, NOV. They are order according to the months of the year beginning with January.

`n` Numeric sort - sort according to the numeric value of a field, not its character value. The -b option is automatically applied, ignoring any leading blanks.

```
$ sort -n perishables
```

`-r` Sort in reverse order.

```
$ sort -r perishables
```

sorting fields

`b` Ignore any leading blanks before a field.

```
$ sort -b perishables
```

`+`*num* The number of fields to skip on a line. Sorting begins from the next field. +2 skips the first two fields and begins sorting on the third field.

```
$ sort +2 perishables
```

`-`*num* The number of field where sorting on a line ends. `-3` will stop a sort on a line at the third field. Fields after the third field would not be used in the sort. This option is often used in conjunction with the + option to isolate a field, and restrict the sort to that field. `+2 -3` sorts only on the third field. You can also specify a range of fields: +1 -4.

```
$ sort +2 -3 perishables
```

`-tc` Specify a new field delimiter, c. The default is a space.

```
$ sort -t: +2  booksC
```

```
$ sort -t\tab   +2  booksT
```

uniq

`uniq`

The `uniq` filter eliminates repeated lines from its input. You can also compare lines base on selected fields. Lines whose selected fields have the same values are considered repetitions and can be eliminated from the output

```
$ uniq options  input-file output-file

$ uniq -d itemlist newfile
```

`c` With this option, `uniq` outputs each line preceded by the number of times the line occurs in the input.

```
$ uniq -c itemlist
```

`d` With this option, `uniq` only outputs repeated lines.

```
$ uniq -d itemlist
```

`u` With this option, `uniq` only outputs lines that are not repeated.

```
$ uniq -u itemlist
```

`-num` The number of fields to be skipped for comparison. Only the remaining fields are compared.

```
$ uniq -3 itemlist
```

`+num` The number of characters to be skipped for comparison. Only the remaining characters are compared, including spaces.

```
$ uniq +12 itemlist
```

Awk

The awk filter:

The AWK filter allows you to create your own filter. AWK has a programming language you can use to program the actions you want taken by the AWK filter. AWK has two basic components: the pattern search and the actions to be taken on each line that matches the pattern. actions are enclose within braces, and patterns are enclosed within slashes. Patterns are also usually enclosed within quotes to prevent any evaluation of metacharacters by the shell. The input files for AWK are listed after the actions.

```
$ awk 'pattern' { actions  } file-list
$ awk '/raven/' { print } books
```

Options

`f` *filename* With this option, AWK will read its commands from filename.

```
$ awk -f fname books
```

`F`*c* This option specifies a field delimiter c, for the input file. The default delimiter is tab or space.

```
$ awk -F: books
```

Variables

`NR`	Record number of current record
`NF`	Number of fields in current record
`$0`	The entire current record.
`$1-$`*n*	The fields in the current record, numbered from 1.
`FS`	Input field delimiter Ð default delimiter is space or tab.
`OFS`	Output field delimiter Ð default delimiter is space or tab.
`RS`	Input record delimiter Ð default is a newline character.
`ORS`	Output record delimiter Ð default is a newline character.
`OMFT`	Output format for numbers Ð default delimiter is %g.
`FILENAME`	Name of current input file.

Initial and terminating conditions

`BEGIN` Execute operations before AWK begins processing.

`END` Execute operations after AWK finishes processing.

Text Functions

print	Output the current line.
getline()	Read next input line, returns 0 if at end of file.
length(*str*)	Returns the number of characters in a string. If length has no argument then it calculates the length of the current record in characters.
index(*str*, *str*)	Determines if str1 is contained in str2 and, if so, return the position in str2 where str1 begins. If not, then returns 0.
split(*str*,*arr*,*delim*)	Copies segments of str that are separated by the delimiter delim into elements of the array arr. It returns the number of elements in the array.
substr(*str*,*pos*,*len*)	Returns a substring of str . Characters in the string are numbered from 1. pos is number that references the character where the substring begins. len is a number that indicates how many character from the posth character the substring will use.

Associative Arrays

`array[`*str*`]` An associative array can be indexed with a string.

`for (`*str* in *array* `)` *action* The for command with the in keyword is used to reference associative arrays.

Operators

Relational Operators

>	greater-than
<	less-than
>=	greater-than-or-equal
<=	less-than-or-equal

Equality Operators

==	equal

`!=`	not-equal
`str ~` *regular expr*	matches regular expression Ð right operand is a regular expression
`str !~` *regular expr*	does not match regular expression Ð right operand is a regular expression

Logical Operators

`&&`	logical AND
`||`	logical OR
`!`	logical NOT

Arithmetic Operators and Functions

`*`	multiplication
`/`	division
`+`	addition
`-`	subtraction
`%`	modulo Ð results in the remainder of a division.
int(*num*)	Truncates a floating point number to its integer value.
cos(*num*)	Returns cosine of num.
sin(*num*)	Returns sine of num.
log(*num*)	Returns natural log of num.
exp(*num*)	Returns exponential of num,
sqrt(*num*)	Returns square root of num.
rand()	Returns random number.
srand(*num*)	Uses num as new seed for rand().

Assignment Operators

`=`	assignment
`++`	increment
`--`	decrement
`+=`	addition and assignment: i = i + 1 is the same as i += 1.
`-=`	subtraction and assignment: i = i - 1 is the same as i -= 1.
`*=`	multiplication and assignment: i = i * 1 is the same as i *= 1.
`/=`	division and assignment: i = i / 1 is the same as i /= 1.
`%=`	modulo and assignment: i = i % 1 is the same as i %= 1.

Control Structures

`{` *actions* `}`	A block is formed by opening and closing braces. A block groups actions making them subject to a control structure such as loop or enclosing the actions in the AWK instruction proper.

`if(`*expression*`)`
action
`else`
action
	The if control structure executes an action if its expression is true. If false then the else action is executed

`while(`*expression*`)` *action*	The while control structure executes an action as long as its expression is true.
`for(`*exp1*`;` *exp2*`;` *exp3*`)` *action*	The for control structure executes an action as long as exp2 is true. The first expression, exp1 is executed before the loop begins. The third expression, exp3 , is executed within the loop after the action.

`for(`*variable* `in` *arrayname*`)`
action
The for-in control structure is designed for use with associative arrays. The variable operand is consecutively assigned the strings that index the array.

`next` The next statement stops operations on the current record and skips to the next record.

`exit` The exit statement ends all processing and executes the END command if there is one.

Format Functions: printf and sprintf

`printf(`*format*`,` *argument-list*`)` Formats the value of the arguments in the argument list according to corresponding format specifiers in the format list.

```
printf("Cost is %f \n"", cost)
```

`sprintf(`*str*`,` *format*`,` *argument-list*`)` Formats the value of the arguments in the argument list according to corresponding format specifiers in the format list. Formatted output is place in string str.

Format string

"%d" The percent sign operator performs a format operation on value in argument list. The type of operation is indicated by a following format specifier, in this case, d, which performs a integer format.

"hello\n" Any characters in the format string are take as characters to be output. Non-printing characters such as the newline and tab are represented with their quoted equivalents.

`\n` newline

`\t` tab

`\0`*num* Octal equivalent of character such as \007

Format Specifiers

`%d` decimal, integer

`%f` floating point

`%e` exponential notation

`%g` Use whatever floating point notation is shorter, f or e.

`%o` octal representation of an integer.

`%x` hexadecimal representation.

`%s` character string.

Format Modifiers

`-` left justify

num Any number modifying the format specifier sets the minimum field width of the value being formatted.

num`.`*num* The period is used to format floating point values and specifies the number of places to the right of the decimal to display.

mailx

Sending and Receiving Messages: mailx

`mailx` *login-name* — The mail command followed by a login name sends a message to the user with that login name. You are prompted for a subject and then enter in the message. End the message with a Ctrl-d on a line of its own.

```
$ mailx chris
Subject: greeting
Hello
^D
EOT
$
```

`mailx` — The mail command without a login name invokes the mail shell and allows you to receive mail. A list of message headers is displayed showing a number and other information for each message. The mail shell has several commands:

`p`	Display a message.	`? p1`
`t`	Display a message.	`? t1`
`h`	Redisplay the list of message headers.	`? h`
`q`	quit the mail shell.	`? q`

mailx Command Options

`-f` *mailbox-filename* — Invoke the mailx utility to read messages in a mail box file in your directory rather than your mail box of waiting messages.

`-H` — Displays only the list of message headers.

`-s` *subject* — When sending messages, this option specifies the subject.

`-F` — Save message in a file with the name of the first recipient.

Mailx Tilde Commands for Sending a Message

Tilde Commands for Message Header

`~h` — Prompts the user to enter in addresses, subject, and carbon copy list.

`~s` *subject* — Enter in a new subject.

`~t` *addresses* — Add addresses to the address list.

`~c` *addresses* — Add addresses to the carbon copy list.

`~b` *addresses* — Add addresses to the blind carbon copy list.

Tilde Commands for Message Text

`~v` — Invokes the vi editor. Changes are saved to the message text.

`~p` — Re-display the text of the message.

`~x`	Quit the message and leave the mailx utility.
`~w` *filename*	Save the message in a file.
`~r` *filename*	Read the contents of a file into the message text.
`~e`	Invokes the default text editor.
`~|` *filter*	Pipe the contents of a message to a filter and replace the message with the ouptut of that filter.
`~m` *message-list*	When sending messages or replying to received mail, this command inserts the contents of a received message. The contents are indented. Used when receiving messages.
`~f` *message-list*	When sending messages or replying to received mail, this command inserts the contents of a received message. Unlike ~m, there is no indentation. Used when receiving messages.

General Tilde Commands

`~?`	Display a list of all the tilde commands.
`~~`	Enter a tilde as a character into the text.
`~!` *command*	Execute a shell command while entering a message.

Message Lists

message -number	Reference message with message number.
num1–num2	Reference a range of messages beginning with num1 and ending with num2.
`.`	The current message
`^`	The first message
`$`	The last message
`*`	All the messages waiting in the mailbox
/pattern	All messages with pattern in the subject field.
address	All messages sent from user with address
`:c`	All messages of the type indicated by c. Message types are as follows:
`n`	newly received messages.
`o`	old messages previously received..

`r`	read messages.
`u`	unread messages.
`d`	deleted messages.

Mailx Commands for Displaying Messages

Status Codes

N	Newly received messages
U	Previously unread messages
R	Read messages in the current session.
P	Preserved messages, read in previous session and kept in incomming mailbox.
D	Deleted messages. Messages marked for deletion
0	Old messages.
*	Messages that you have saved to another mailbox file.

Display Messages

`h`	Re-display the message headers.
`z+ z-`	If header list takes up more than one screen you can scroll header list forward and backward with the + and -.
`t` *msge -list*	Displays a message referenced by the message list. If no message list is used, than the current message is displayed.
`p` *msge -list*	Displays a message referenced by the message list. If no message list is used, than the current message is displayed.
`n`	Displays next message.
`+`	Displays next message.
`-`	Displays the previous message
`top` *message -list*	Displays the top few lines of a message referenced by the message list. If no message list is used, than the current message is displayed.
`=`	Displays the number of the current message

Deleting and Restoring Messages

`d` *message -list*	Deletes a message referenced by the indicated message list from your mail box.
`u` *message -list*	Un-deletes a message referenced by the indicated message list that has been previously deleted.
`q`	Quit from the mailx utility and save any read messages in the mbox file.
`x`	Quit from the mailx utility and do NOT erase any messages you deleted. This is equivalent to executing a u command on all deleted messages before quitting.

`pr` *message -list* Preserve messages in your waiting mail box even if you have already read them.

Sending and Editing Messages

`r`	Send a reply to all persons who received a message.
`R`	Send reply to the person who sent you a message.
`m` *address*	Send a message to someone while in the mailx utility.
`v` *message -list*	Edit a message with the vi editor.

Saving Messages

`s` *message --list filename*	Save a message referenced by the message list in a file, including the header of the message.
`S` *message-list*	Save a message referenced by the message list in a file named for the sender of the message.
`w` *message-list filename*	Save a message referenced by the message list in a file without the header. Only the text of the message is saved.
`c` *message-list filename*	Copy a message referenced by the message list to a file without marking it as saved.
`folder` *mailbox --filename*	Switch to another mailbox file. `%` Represents name of incoming mailbox file. folder % Switch to incoming mailbox file. `#` Represents name of previously accessed mailbox file. folder # Switch to previous mailbox file. `&` Represents name of mailbox file used to automatically save your read messages, usually called mbox. `folder &` Switch to mbox file.

General Commands

`?`	Display a list of all the mail commands.
`!` *command*	Execute a user shell command from within the mail shell.

`alias` *name address-list*	Create an alias for a list of addresses.
	`alias myclass chris aleina larisa`
	`$ mailx myclass`

Mailx Options

append	Place messages saved in your mailbox at the end, rathe than the beginning (disabled by default).
asksub	Prompt for subject. `set asksub`
askcc	Prompt for carbon copy addresses. `set askcc`
autoprint	When deleting messages, show the next message after the last one deleted (disabled by default).
cmd=*cmd*	Specify default command to use with pip operation should no command be given (disabled by default).
crt=*n*	For messages that are n or more lines, display them using your PAGER program. (disabled by default).
dbug	Debug mode with detailed descriptions of actions take, but there is no actual delivery of messages (disabled by default).
dot	Allow you to end a message by entering a dot on a line by itself, instead of Ctrl-d. (disabled by default).
escape=*c*	Specify c as the escape character in the input mode.
flipr	Switch the R and r commands sothe R sends a reply to senders of several specified messages and r sends a response to all other recipients of a message you received. (disabled by default).
folder=*directory*	Saves any mailbox files created by the s or S command to the directory assigned to it. `set folder=$HOME/mail`
header	Show header sumary when starting up (default)
hold	Keep read messages in your incomming mailbox, instead of mbox. (disabled by default).
ignore	Ignore interrupts when composing messages (disabled by default).
ignoreeof	Disables the use of Ctrl-d to end input when composing messages. Should have dot enabled so you can end messages with just a . on a line by itself, or use ~. to end messages. (disabled by default).

indentprefix=*string*	Specify string (characters) to be placed at the beginning of each line of a copied message included within a response you are composing. (disabled is tab).
keep	Keep mailbox files when they become empty. (disabled by default).
keepsave	When you save a message to a particular mailbox file, also save a copy to your standard mailbox file, usually mbox. (disabled by default).
metoo	Will allow you send a copy of a message to yourself that you are also sending to others. By default your name would automaticaly be deleted from a list of addresses. (disabled by default).

outfolder	Place record file in folder directory. In the following example, outbox will be a file in the directory defined by folder.
	`set record=outbox`
	`set outfolder`

page	When piping several messages through a pipe command, this option will insert a formfeed after each so that each message will start on its own screen. (disabled by default).
prompt=*string*	Redefine mailx prompt `set prompt="&"`
record=*filename*	Automatically save a copy of any message that you create and send. Messages are saved in a file specified when you set the record option. `set record=$HOME/outbox`
save	Save incomplete or interrupted messages in your dead letter file. (disabled by default).
screen=n	Sets the number of lines of the header displayed on your screen (default is 5)
sendwait	Wait for background mailer to finish processing before resuming with mailx
showto	For messages shown in the header summary for which you are the sender, show the recipients name instead of yours (disabled by default).
sign=*string*	Define string to be inserted by the ~a tilde command into a message that you are inputting (empty by default). `set sign="Justin and Dylan"`
Sign=*string*	Define string to be inserted by the ~A tilde command into a message that you are inputting (empty by default).

toplines=*n*	Specifies how many lines the top command will show of the header summary (default is 5).
quit	Do not show identification line (disabled by default)

Mailx Configuration Variables

MBOX=*filename*	Holds the name of the mbox file to which read messages are automatically saved. By default, mbox is placed in your home directory. To put it in the folder directory, place a + sign before the mbox name. `set MBOX=+mbox`
DEAD=*filename*	Specifh the dead letter file when incomplete and interrupted messages are placed.
LISTER=*cmd*	Specify the command to use to list he contents of the folder directory (default is ls).
EDITOR=*cmd*	Specify the editor to use when invoked with the ~e command (default is Ed, a line editor).
VISUAL=*cmd*	Specify the editor to use when invoked with the ~v command (default is Vi).
PAGER=*cmd*	Specify the pager program to use (default is pg or more).
sendmail=*cmd*	Specify the mail transport agent for your mailer (default is usually sendmail or rmail, include full path names).

FTP

FTP Options

`-v`	Verbose, displays all responses from the remote system and reports data transfer statistics.
`-n`	Do not perform "auto-login" upon connecting to a remote system. Otherwise, if auto-login is enabled, Ftp will check the .netrc file in the user's home directory for an entry with the login name on the remote system. If no entry exists, Ftp will prompt for a

	remote login name, and then prompt for a password if needed. The default login name is the user login name on the local system.
`-i`	Turns off interactive prompting during multiple file transfers. Applies to `mget` and `mput` commands, canceling prompts for the transfer of each individual file in an `mget` or `mput` operation.
`-d`	Enables debugging.
`-g`	Disables file name expansion (globbing). Use of *, ?, and [] for filename matching is disabled.
system-address	You can specify the remote system that Ftp is to immediately connect to, skipping an open command for that connection.

FTP Connection and Directory Commands

`!` [*command* [*args*]	Executes a Unix shell command. You can specify arguments for the shell command if needed. With no shell command, the ! places you in an interactive Unix shell where you can issue Unix commands. Enter exit or Ctrl-d to return to Ftp.
`account` [*passwd*]	Provide a supplemental password after login if the remote system requires one. You can enter the password as an argument to account. If not, you will be prompted for it and entry will not be echoed on your screen.
`bye`	End and exit the Ftp program. If you are connected to a remote system at the time, the connection will be terminated.
cd *remote-directory*	Change directory on the remote system to remote-directory, making it your working directory on the remote system.
cdup	Change to the parent directory of the remote system's working directory (like a `cd ..` operation for remote directories).
chmod *mode file-name*	Change the permissions of a remote file. The Ftp chmod command works on remote files.
close	Terminate the FTP session with the remote system, and return to the Ftp command interpreter. Any defined macros are erased.
delete *remote-file*	Delete a file on the remote system.

debug [*debug-level*]	Toggle debugging mode. You can set the debut level. In debug mode, Ftp displays commands sent to the remote system, preceding them with `-->'.
`dir` [*remote-directory*]	[local-file] List the contents of a remote directory (like `ls`), using long form. If you do not specify a directory name, then the current working remote directory is used. You can specify a local filename to which the directory listing will be saved. If not filename is specified, the your local standard output is used, usually displaying the listing on your screen.
`disconnect`	Same as the close command, terminate a connection to a remote system.
`glob`	Toggle Unix filename expansion for `mdelete`, `mget` and `mput`. If globbing is turned off with `glob`, the file name arguments containing expansion character such as *, ?, and [], are not expanded. These character are read literally and taken as part of the file name. Filename expansion is performed by the remote system and may differ accordingly. You can preview the results using the `mls` command, `mls` remote-files -. `glob` does not enable `mget` and `mput` to transfer directory subtrees. You can transfer subtrees using `tar` archives that you later extract.
`hash`	Display hash-signs ("#") during a file transfer. One # is displayed for each data block transferred. The size of a data block is 1024 bytes.
`help` [*command*]	Display a list of Ftp commands. If command is specified, display help information about that command.
`idle` [*seconds*]	Display the inactivity timer setting. With a seconds argument, it sets the inactivity timer on the remote server to that number of seconds.
`lcd` [*directory*]	Change the working directory on your local system. If you do not specify a directory, you change to your local system's home directory (line cd).
`ls` [*remote-directory*] [*local-file*]	List the contents of a remote directory. If you do not specify a directory name, then the current working remote directory is used. You can specify a local filename to which the directory listing will be saved. If not filename is specified, the your local standard output is used, usually displaying the listing on your screen.
`mdelete` [*remote-files*]	Delete several remote-files on the remote machine.
`mdir` *remote-files local-file*	Lets you specify several remote files to list as `dir` does. Instead of specifying a single directory or file you can list several particular

	files. The last file is take to be local file where you want to save the listing results. If interactive prompting is on, Ftp will prompt you to verify that the last argument is that local file.
`modtime` *file-name*	Display the last modification time of file-name on the remote system.
`nlist` [*remote-directory*] [*local-file*]	Print a remote-directory listing. If remote-directory is left unspecified, the current working directory is used. If local-file is specified, the listing is save in the that file on your local system. If no local-file is specified, then the listing is output to your standard output which, by default, is displayed on your screen. If interactive prompting is on, Ftp will prompt you to verify that the last argument is a local file for receiving nlist output.
`open` *system-address* [*port*]	Make an Ftp connection to a remote system or Ftp site. You can specify a port number on which to connect to the remote system. With the autologin option on, Ftp will try to automatically log the user in. auto-login is on by default.
`pwd`	Display current working directory on the remote system.
`quit`	Quite Ftp, closing any open connections. Same as bye.
`quote` *arg1 arg2 ...*	Send the arguments verbatim to the remote system.
`remotehelp` [*command-name*]	Request help from the remote system. You can specify help for a specific command.
`remotestatus` [*file-name*]	With no arguments, show status of remote system. If filename is specified, show status of file-name on remote system.
`rename` [*file-name*] [*new-name*]	Rename a file on the remote system.
`reset`	Clear reply queue. Re-synchronizes command/reply sequencing with the remote system. Used if remote system violates Ftp protocol.
`rmdir` *directory-name*	Delete a directory on the remote system.
`site` *arg1 arg2 ...*	Send verbatim arguments that are commands to be executed on the remote system.
`size` *file-name*	Obtain the size of a file on remote system.
`status`	Display the current status of Ftp.

`system`	Display the type of operating system used on the remote system.
`tenex`	Set the file transfer type to that needed to talk to TENEX machines.
`trace`	Toggle packet tracing.
`umask` [*newmask*]	Set the default umask on the remote server to newmask. With no arguments, the current umask is displayed.
`user` *user-name* [*password*] [*account*]	Identify yourself to the remote system. If the password or account are required by the remote system, and you do not specify them, then Ftp will prompt you to enter them. Unless Ftp is invoked with "auto-login" disabled, this process is done automatically on initial connection to the remote system.
`verbose`	Toggle verbose mode. If on, all responses from the remote system are displayed. When a file transfer completes, statistics regarding the efficiency of the transfer are reported. verbose is on by default.
`?` [*command*]	Display help information about a command. Same as help.

FTP File Transfer Commands

`append` *local-file* [*remote-file*]	Append a local file to a file on the remote system. If you do to specify a remote-file name, then the local-file is for that name.
`ascii`	Set the file transfer type to network ASCII. This is usually the default type (transfer type is changed to binary by many Internet Ftp sites).
`bell`	Sound bell after a file transfer.
`binary`	Set the file transfer type to binary.
`case`	Toggle the remote file name case mapping during `mget` commands. When case is on, remote file names on the remote system that have letter in upper case are written in the local directory with the letters mapped to lower case. Default is off.
`cr`	Toggle carriage return stripping during ascii type file transfer. This is used for files such as DOS files that have use both a carriage return and line-feed character for a newline, instead of just a line-feed character as Unix does. When on, `cr` will strip the carriage-return character from ascii files, making the file conform to the Unix ascii file using just a line-feed for newlines.
`form` *format*	Set the format for the file transfer form. The default format is "file".

`get` *remote-file* [*local-file*]	Transfer a remote-file from the remote system to your local system. You can specify a local file name for it, if not, the same remote name is used. If there is already a file by that name on your local system, then the filename will be altered. The current settings for type, form, mode, and structure are used for transferring the file.
`glob`	Toggle Unix filename expansion for `mdelete`, `mget` and `mput`. If globbing is turned off with `glob`, the file name arguments containing expansion character such as *, ?, and [], are not expanded. These character are read literally and taken as part of the file name. Filename expansion is performed by the remote system and may differ accordingly. You can preview the results using the `mls` command, `mls` remote-files -. `glob` does not enable `mget` and `mput` to transfer directory subtrees. You can transfer subtrees using `tar` archives that you later extract.
`hash`	Display hash-signs ("#") during a file transfer. One # is displayed for each data block transferred. The size of a data block is 1024 bytes.
`mget` *remote-files*	Perform any specified file name expansion in remote-files on the remote system. Then execute a `get` operation for each file name generated. File name expansion is performed as indicated by `glob`. File names are processed according to case, ntrans, and nmap settings.
`mput` *local-files*	Perform any specified file name expansion in local-files on the your local system. Then execute a `put` operation for each file name generated. File name expansion is performed as indicated by `glob`. File names are processed according to ntrans, and nmap settings.
`newer` *file-name* [*local-file*]	Perform a `get` operation to transfer a file from the remote system, only if the modification time of the remote file is more recent that the local-file specified on your local system. If no local-file is specified, then a file of the same name is used. If the file does not exist on the current system, the remote file is considered newer.
prompt	Toggle interactive prompting for multiple file transfers using `mget` or `mput`. Interactive prompting is on by default. If turned off, then you are not prompted for individual files. With `mput` and `mget`, all matching files are transferred, and with `mdelete` all matching files are deleted.
`proxy` *ftp-command*	Allows file transfers between two remote systems. The first proxy operation should be an open command to connect to the second remote system. You can then execute Ftp commands on the second system using the proxy command. Close the connection with a proxy command followed by a close command. Proxy open operation does not define new macros and a proxy close will not erase macros. `get` and `mget` transfer files to the second system first. `put` and `mput` transfer files from the second system to the first. Execute an Ftp command on a secondary control connection.

proxy ? list help information. Depends on support of the Ftp protocol PASV command by the second system.

`put` *local-file* [*remote-file*] Transfer local file to the remote system. If file name for remote-file is not unspecified, the local-file name is used. ntrans or nmap settings may apply. Transfers use the current settings for type, format, mode, and structure.

`recv` *remote-file* [*local-file*] Transfer files from the remote system. Same as `get`.

`reget` *remote-file* [*local-file*] Transfer files from the remote system, like `get`. Will also resume transfer of interrupted file transmissions. If the 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. Useful fro transferring large files.

`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.

`runique` Toggle storing of files on the local system with unique filenames. If off (the default) then files transferred with the same name as an existing file will overwrite that file. If off, then a new file name is generated, preserving the exiting file. The same name with a ".1" appended to it is used. 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.

`send` *local-file* [*remote-file*] Transfer files from your local system to the remote system. Same as `put`.

`sendport` Toggle the use of PORT commands. By default, Ftp will attempt to use a PORT command when establishing a connection for each data transfer. The use of PORT commands can prevent delays when performing multiple file transfers. If the PORT command fails, Ftp will use the default data port. When the use of PORT commands is disabled, no attempt will be made to use PORT commands for each data transfer.

`sunique` Toggle storing of files on remote system under unique file names. If off (the default) then files transferred to the remote system with the same name as an existing file will overwrite that file. If off, then a new file name is generated, preserving the exiting file. The same name with a ".1" appended to it is used, and so on.

Macros, map, help, and transmission parameters

`!` [*command* [*args*] Executes a Unix shell command. You can specify arguments for the shell command if needed. With no shell command, the ! places you in an interactive Unix shell where you can issue Unix commands. Enter exit or Ctrl-d to return to Ftp.

`$` *macro-name* [*args*]	Execute the macro macro-name . Macros are defined with the macdef command. Arguments are passed to the macro unglobbed.
`ascii`	Set the file transfer type to network ASCII. This is usually the default type (transfer type is changed to binary by many Internet Ftp sites).
`bell`	Sound bell after a file transfer.
`binary`	Set the file transfer type to binary.
`case`	Toggle the remote file name case mapping during `mget` commands. When case is on, remote file names on the remote system that have letter in upper case are written in the local directory with the letters mapped to lower case. Default is off.
`debug` [*debug-level*]	Toggle debugging mode. You can set the debut level. In debug mode, Ftp displays commands sent to the remote system, preceding them with `-->'.
`form` *format*	Set the format for the file transfer form. The default format is "file".
`help` [*command*]	Display a list of Ftp commands. If command is specified, display help information about that command.
`idle` [*seconds*]	Display the inactivity timer setting. With a seconds argument, it sets the inactivity timer on the remote server to that number of seconds.
`macdef` *macro-name*	Define a macro with the name macro-name. Enter Ftp command for the macro on the following lines. End the macro definition with an empty line. There is a limit of 16 macros and 4096 total characters in all defined macros. Macros remain defined until a close command is executed. Macros can take arguments that are referenced in the macro definition with a `$` and the number of the argument. The `$i` implements a loop on the macro, repeating it once for each argument entered, with `$i` referencing each argument in turn. You can quote `$` in the macro definition by preceding it with `\` to enter the `$` character. `\\` will enter a backslash character.
`nmap` [*inpattern outpattern*]	With no arguments, turns off file mapping. With inpattern and outpattern arguments, `nmap` specifies translations to be performed. Filename matching the inpattern are translated into the outpattern. Elements of the original pattern name are referenced in the inpattern and outpattern templates using `$`*num* references. Useful when connecting to a non-UNIX remote systems with different file naming conventions.
`ntrans` [*inchars* [*outchars*]]	With no arguments, the filename character translation mechanism is unset. With arguments, characters in remote filenames are translated if there is no specified target filename. 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. Useful when connecting to a non-UNIX remote systems with different file naming conventions.
`status`	Display the current status of Ftp.
`struct` [*struct-name*]	Set the file transfer structure to struct-name. The default is "stream".
`system`	Display the type of operating system used on the remote system.
`trace`	Toggle packet tracing.
type [*type-name*]	Set the file transfer type. If no arguments, the current type is displayed. The default type is network ASCII.
`umask` [*newmask*]	Set the default umask on the remote server to newmask. With no arguments, the current umask is displayed.

Archives and Compression

tar

`tar` *options files*	Back up files to tape, device, or archive file.

`tar` options

`c`	Create a new archive.
`r`	Append files to an archive.
`u`	Update an archive with new and changed files. Add only those files that have been modified since they were archived or files that are not already present in the archive.
`w`	Wait for a confirmation from the user before archiving each file. Allows you to selectively update an archive.
`x`	Extracts files from an archive.
`m`	When extracting a file from an archive, do not give it a new time stamp.
`f` *archive-name*	Save the tape archive to the file archive-name instead of to the default tape device. The archive-name can be either a file or another device such as a tape or disk. The default device is held in /etc/default/tar file.
`v`	Display each file name as it is archived.

`z, --gzip, --ungzip`	Filter the archive through `gzip`.
`Z`, --compress, --uncompress	Filter the archive through `compress`.
--use-compress-program *prog*	Filter the archive through prog (must accept -d option)
`d`, --diff, --compare	Find differences between archive and file system

Zip

`-A`	Adjust self-extracting executable archive.
`-b` *path*	Use the specified path for the temporary zip archive.
`-c`	Add one-line comments for each file.
`-d`	Remove entries from a zip archive. .
`-D`	No entries created for directories in zip archives.
`-e`	Encrypt the contents of the zip archive using a password which is entered on the terminal in response to a prompt.
`-f`	Replace (freshen) an existing entry in the zip archive only if it has been modified more recently than the version already in the zip archive. Does not add files new files to the zip archive.
`-F`	Fix the zip archive. Used if part of the archive is missing.
`-g`	Grow (append to) the specified zip archive, instead of creating a new one.
`-h`	Display the zip help information.
`-i` *files*	Include only the specified files
`-j`	Store just the name of a file without the path, and do not store directory names. By default, zip stores the full path (relative to the current path).
`-J`	Strip any prepended data from the archive.
`-k`	Attempt to convert the names and paths to conform to MSDOS.
`-l`	Translate the Unix end-of-line character line-feed into the MSDOS carriage-return and line-feed.
`-ll`	Translate the MSDOS end-of-line, carriage-return, and line-feed into Unix line-feed.

`-L`	Display the zip license.
`-m`	Move the specified files into the zip archive; actually, this deletes the target directories/files after making the specified zip archive.
`-n` *suffixes*	Do not compress files named with the given suffixes.
`-o`	Set the "last modified" time of the zip archive to the latest (oldest) "last modified" time found among the entries in the zip archive.
`-q`	Quiet mode; eliminate informational messages and comment prompts.
`-r`	Travel the directory structure recursively.
`-t` *mmddyy*	Do not operate on files modified prior to the specified date, where mm is the month (0-12), dd is the day of the month (1-31), and yy are the last two digits of the year.
`-T`	Test the integrity of the new zip file.
`-u`	Replace (update) an existing entry in the zip archive only if it has been modified more recently than the version already in the zip archive.
`-v`	Verbose mode or print diagnostic version info.
`-x` *files*	Explicitly exclude the specified files.
`-X`	Do not save extra file attributes (file times on Unix).
`-y`	Store symbolic links as such in the zip archive, instead of compressing and storing the file referred to by the link.
`-z`	Prompt for a multi-line comment for the entire zip archive. The comment is ended by an end of file, ^D.
`-#`	Regulate the speed of compression using the specified digit #, where -0 indicates no compression (store all files), -1 indicates the fastest compression method (less compression) and -9 indicates the slowest compression method (optimal compression, ignores the suffix list). The default compression level is -6.
`-@`	Take the list of input files from standard input.

Remote Access Commands

rwho

`rwho`	This command displays all users logged into systems in your network

ruptime

`ruptime`	This command displays information about each system your network.

ping

`ping`	This command detects whether a system is up and running.

rlogin

`rlogin` *system-name*	The `rlogin` command allow you to remotely login to an account on another system. It has a -l option that allows you to specify the login name of the account.
	`$ rlogin mygame`
	`$ rlogin mygame -l justin`

rcp

`rcp` *sys-name*:*file1* *sys-name*:*file2*	The `rcp` command allows you to copy a file from an account on one system to an account on another system. If no system name is given then the current system is assumed.
	`$ rcp mydata mygame:newdata`
	
	`-r` The `rcp` command with the -r option allows you to copy directories instead of just files.
	`$ rcp -r newdocs mygame:edition`

rsh

`rsh` *sys-name Unix-command*	The `rsh` command allows you to remotely execute a command on another system. `$ rsh mygame ls`

telnet

Telnet Options

`-8`	Request 8-bit operation. Attempts to negotiate the TELNET BINARY option for both input and output.
`-E`	Disable the escape character, setting the escape character to "no character".
`-L`	Set 8-bit data path on output, causing the TELNET BINARY option negotiated on output only.
`-a`	Perform an automatic login using the user name from the Telnet USER variable. By default this is the same as your current login name as specified by your $USER environment variable. The ENVIRON must be supported by the remote system.
-d	Set debug toggle to TRUE.
-r	Emulate `rlogin` operation. The default escape character is a tilde. An escape character followed by a dot disconnects from the remote system. Ctrl-z suspends Telnet, and a ^] escapes to the Telnet command mode. The escape keys can only be entered at the beginning of the line.
`-S` *tos*	Specify the IP type-of-service, tos.
`-e` *escapechar*	Specify a new value for the Telnet escape character used to enter the command mode. If no value is specified, no escape character is used.
`-l` *login-name*	Specify the login name to be used for the remote system. The login name is placed in the Telnet USER variable and requires the remote system to support the TELNET ENVIRON option. This option implies the -a option. You can also use this option with the open command.
`-n` *tracefile*	Saves recorded trace information in tracefile. See tracefile command.
`host`	Specify a remote system (host) to connect to on your network.
`port`	Specifiy a port number to use.

Telnet Commands

open *host* [[-l] *user*] [-*port*]	Open a connection to a remote system (host). Optional arguments are a remote login name as specified with the -l option, or a port number. The host can be a host name or an IP (Internet) address. The default port for the system's Telnet daemon is usually 23. If another port is specified then Telnet protocols are not implemented. You can force Telnet protocols by placing a - before the port number.
close	Close the connection to the remote system, returning to the Telnet command mode.

display *variable* ...	Display Telnet variable and toggle values. You can list the variable or toggle values you want displayed.
environ *arguments*...	Exports your shell environment variables across the Telnet link using the TELNET ENVIRON protocol option. By default the DISPLAY and PRINTER variables are exported. The USER variable is sent if the -a or -l command-line options are used. The remote system may still ask explicitly for variables not marked for export.
Arguments	
define *variable value*	Define a variable with the specified value. The variable is automatically marked for export.
undefine *variable*	Remove definition of variable.
export *variable*	Mark a specified variable for export to the remote system.
unexport *variable*	Specified variable is not marked for export to the remote system.
list	List the current set of environment variables. Those marked with a * will be exported to the remote system.
`?`	Display environment command help information
logout	Close the Telnet connection (like the close command). Requires that the remote system support the Telnet LOGOUT option.
`quit`	Close any open session and exit Telnet.
send *code*	Send Telnet special control character sequences to the remote host. The sequences are referenced with Telnet codes. See Table 4 for the list of send codes.
mode *type*	Sets the mode of operation for the Telnet connection.
Modes	
`character`	Transmit data a character at a time (disables the LINEMODE option).
`line`	Transmit data a line at a time (enables the LINEMODE option). If remote system cannot use LINEMODE option, then "old line by line" mode is used.
`set` *variable value*	Assign a value to a Telnet variable or toggle. To assign a value to a toggle you use the values TRUE to turn it on and False to turn it off. Use display command to list the current values.

`unset` *variable*	Unset the value of a Telnet variable.
`toggle` *toggle-list*	Turn a Telnet toggle on or off, toggling between TRUE and FALSE. You can set their values explicitly with the set command.
`z`	Suspend Telnet.
`!` [*command*]	Execute a single command in a subshell on the local system. If no command is specified then an interactive shell is started up.
`?` [*command*]	Displays a help summary. If a command is specified, Telnet displays the help information for that command.
`slc` *state*	Set or change the state of the special control characters when the TELNET LINEMODE option has been enabled.

States

`check`	Verify the current settings for the current special control characters. The remote system sends all the current metacharacter settings, and, if there are any discrepancies with the local system, the local system switches to the remote value.
export	Use the local defaults for the special control characters.
import	Use the remote defaults for the special control characters.
status	Show the current status of Telnet. Displays the name of the remote system.

Telnet send codes

(Send signals with the send command)

abort	TELNET ABORT (Abort Processes) sequence.
ao	TELNET AO (Abort Output) sequence. Causes the remote system to flush all output from the remote system to the user's terminal.
ayt	TELNET AYT (Are You There?) sequence.
brk	TELNET BRK (Break) sequence. Send a break character.
ec	TELNET EC (Erase Character) sequence. Erase the last character entered.
el	TELNET EL (Erase Line) sequence. Erase the current line.
eof	TELNET EOF (End Of File) sequence. Send an end-of-file character, usually a Ctrl-d.
eor	TELNET EOR (End of Record) sequence.

escape	The Telnet escape character.
ip	TELNET IP (Interrupt Process) sequence. Send an interrupt character, usually a Ctrl-c.
susp	TELNET SUSP (Suspend Process) sequence.
synch	TELNET SYNCH sequence. Discards previously typed input that has not yet been read.
?	Prints out help information for the send command.

Telnet variables

(set variable values using the set command and unset the with the unset command.)

ayt	Status character, a TELNET AYT sequence.
echo	Toggles local echoing of entered characters (default is Ctrl-e).
eof	End-of-file character.
erase	Erase character, TELNET EC sequence.
escape	Telnet escape character (default is "^["). Causes entry into Telnet command mode.
flushoutput	Flush character, TELNET AO sequence.
forw1	Forward partial lines to the remote systems, based on eol character.
forw2	Forward partial lines to the remote systems, based on eol2 character.
interrupt	Interrupt character, TELNET IP sequence
kill	Kill character, TELNET EL sequence.
lnext	lnext character.
`quit`	Quit character, Sends a TELNET BRK sequence to the remote system.
reprint	Reprint character.
`rlogin`	`rlogin` mode escape character that enables `rlogin` mode. Same as with the -r Telnet option.
start	Start character, default is your system's kill character.
stop	Stop character.
susp	Suspend character, a TELNET SUSP sequence.

tracefile	Filename for tracefile to which netdata or option tracing will write to. If set to "-", then tracing information is written to standard output (the default).
worderase	Worderase character.
?	Displays the set commands.

Telnet toggle features

(use the toggle command to toggle features on or off. You can list several features with the same toggle command. You can use the set command to turn them on or off by setting their values to TRUE or FALSE)

autoflush	If TRUE, does not display data on the user's system until the remote system acknowledges it has processed ao or quit sequences sent to it. Default is TRUE unless "stty noflsh" is entered.
autologin	Use the user's login name to login.
autosynch	Flush previously typed input. Default is FALSE.
binary	Enable or disable the TELNET BINARY option on both input and output.
inbinary	Enable or disable the TELNET BINARY option on input.
outbinary	Enable or disable the TELNET BINARY option on output.
crlf	If TRUE, then return characters are sent as return and line-feed. If is FALSE, then returns are sent as returns. Default is FALSE.
crmod	Map single return characters received from remote system to return and a line feed. Default is FALSE.
localchars	If TRUE, then Telnet special control characters are recognized locally, and translated into TELNET control sequences.
netdata	Display network data (in hexadecimal format). Default is FALSE.
options	Display of internal Telnet protocol processing. Default is FALSE.
prettydump	With netdata toggle enabled, prettydump outputs netdata output in more readable format.
skiprc	IfTRUE, the .telnetrc file is not read. Default is FALSE.
termdata	Display of terminal data (in hexadecimal format). Default is FALSE.
?	Displays the toggle commands.

Vi and Ex Editor Commands

Cursor Movement Commands

Cursor Movement:

`h`	left	Move cursor left one character.
`l`	right	Move cursor right one character.
`k`	up	Move cursor up one line.
`j`	down	Move cursor down one line.
`w`	forward a word	Move cursor forward one word.
`W`	forward a word	Move cursor forward one space delimited word.
`b`	back a word	Move cursor back one word.
`B`	forward a word	Move cursor back one space delimited word.
`e`	to end of word	Move cursor to the end of the next word.
`E`	to end of word	Move cursor to the end of the next space delimited word.
`0`	start of line	Move cursor to the beginning of line.
`$`	end of line	Move cursor to the end of the line.
enter	start of next line	Move cursor to beginning of next line.
`-`	start of prior line	Move cursor to beginning of previous line.
`(`	previous sentence	Move cursor to beginning of sentence.
`)`	end of sentence	Move cursor to end of sentence, successive command moves to beginning of next sentences.
`{`	start of paragraph	Move cursor to beginning of paragraph.
`}`	next paragraph	Move cursor to end of paragraph.
Ctrl-f	Next screen	Move forward by a screen of text. The next screen of text is displayed.

Ctrl-b	Previous screen	Move backward by a screen of text. The previous screen of text is displayed.
Ctrl-d	Scroll forward	Move forward by one-half screen of text.
Ctrl-u	Scroll backward	Move backward by one-half screen of text.
G	To last line To specific line	Move cursor to last line in the text. When preceded by a number the G command moves the cursor to that line in the text. Entering the numbers 45 followed by the G command will place the cursor on line 45.
H	To top of screen	Move cursor top line displayed on screen.
M	To middle of screen	Move cursor to middle line displayed on screen.
L	To bottom of screen	Move cursor bottom line displayed on screen.
''	Move to previous	Moves the cursor its previous location in the text.
m*mark*	Mark text	Place a mark on a line of text. The mark can be any alphabetic character.
'*mark*	Move to mark	Moves the cursor to the line with the mark.

Search: The two search commands open up a line at the bottom of the screen and allow the user to enter a pattern to be searched for. Press an enter key after typing in the pattern.

/*pattern*	Search pattern	Search forward in the text for a pattern.
?*pattern*	Search pattern	Search backward in the text for a pattern.
n	Repeat search	Repeat the previous search, whether it was forward or backward.
N	Repeat search	Repeat the previous search in opposite direction.
/	Repeat search	Repeat the previous search in forward direction.
?	Repeat search	Repeat the previous search in backward direction.

Metacharacters used in search strings.

.	Any Character	Matches on any one possible character in a pattern.
[]	Classes	Matches on classes of characters, a set of characters, in the pattern.
0	Start of a Line	References the beginning of a line.

`$`	End of a Line	References the end of a line.
`/<`	Start of a word	References the start of a word.
`>/`	End of a word	References the end of a word.

Input, Delete, Copy, and Move Commands

Input: All input command place the user in input. The user then leaves input by pressing the Escape key, ESC.

`a`	append	Enter input after the cursor. Press escape to end input.
`A`	append at end	Enter input at the end of a line. Press escape to end input.
`i`	insert	Enter input before the cursor. Press escape to end input.
`I`	insert at start	Enter input at the beginning of a line. Press escape to end input.
`o`	open below	Enter input below the line the cursor is on. A new empty line is inserted below the one the cursor is currently on. Press escape to end input.
`O`	open above	Enter input above the line the cursor is on. A new empty line is inserted above the one the cursor is currently on. Press escape to end input.

Delete:

`x`	delete character	Delete the character the cursor is on.
`X`	delete before	Delete the character before the character the cursor is on.
`dw`	delete a word	Delete the word the cursor is on.
`db`	deletes to beginning of a word	
`dw`	deletes space delimited word.	
`dB`	deletes to beginning of a space delimited word	
`dd`	delete a line	Delete the line the cursor is on.
`D`	delete rest of line	Delete the rest of the line the cursor is on.
`d0`	deletes to beginning of a line	Delete text from cursor to beginning of line.
`d`	delete text	Delete following text specified.
`d)`	deletes the rest of a sentence.	
`d}`	deletes the rest of a paragraph.	

`dG`	deletes the rest of the file.	
`dm`	d followed by a mark deletes everything to mark.	
`dL`	deletes the rest of the screen.	
`dH`	deletes to the top of the screen.	
`J`	join two lines	Join the line below the cursor to the end of the current line, in effect deleting the newline character of the line the cursor is on.

Change: Except for the replace command, `r`, all change commands place the user into input after deleting text.

`s`	change a character	The character the cursor is on is deleted and the user is placed into the input mode. Press escape to end input.
`cw`	change a word	The word the cursor is on is deleted and the user is placed into the input mode. Press escape to end input.
`cb`	changes to beginning of a word	
`cW`	changes space delimited word.	
`cB`	changes to beginning of a space delimited word	
`cc`	change a line	The line the cursor is one is deleted and the user is placed into input. Press escape to end input.
`C`	change rest of line	The rest of the line the cursor is one is deleted and the user is placed into input. Press escape to end input.
`c0`	changes to beginning of a line	Changes text from cursor to beginning of line.
`c`	changed text	Change following text specified.
`c)`	change the rest of a sentence.	
`c}`	change the rest of a paragraph.	
`cG`	change the rest of the file.	
`cm`	c followed by a mark changes everything to mark.	
`cL`	change the rest of the screen.	
`cH`	change to the top of the screen.	
`r`	replace a character	The `r` command replaces the character the cursor is on. After pressing `r` the user enter the replacement character. The change is made without entering input. The user remains in the vi command mode.
`R`	overwrite	The user is first placed into the input mode. For each character typed in, a corresponding character in the text in the same position will be deleted. This

effect appears as an overwrite mode on the screen. In fact one is in input. Press escape to end input.

Move: Text is moved by first deleting it, moving the cursor to the place where the text is to be inserted, and then pressing the `p` command. When text is deleted it is automatically held in a special buffer. The `p` command inserts the contents of this buffer after the character, word, or line the cursor is on..

`p`	insert after	Deleted or copied text will be inserted after the character or line the cursor is on.
`P`	insert before	Deleted or copied text will be inserted before the character or line the cursor is on.
`dw p`	move a word	Delete the word. Move the cursor to the place the word is to be moved. Press `p` to insert the word after the word the cursor is on.
`dw P`	move a word	Delete the word. Move the cursor to the place the word is to be moved. Press `P` to insert the word before the word the cursor is on.
`dd p`	move a line	Delete the line. Move the cursor to the place the line is to be moved. Press `p` to insert the word after the line the cursor is on.
`d p`	move text	Delete following text specified and the move cursor and press `p` or `P`.
`d) p`	moves the rest of a sentence.	
`d} p`	moves the rest of a paragraph.	
`dG p`	moves the rest of the file.	
`dm p`	`d` followed by a mark moves everything to mark.	
`dL p`	moves the rest of the screen.	
`dH p`	moves to the top of the screen.	

Copy: The copy commands are meant to be used in conjunction with the `p` command. Upon copying text, the user moves to the cursor to the place where the copy is to be inserted. The `p` command then inserts the text after the character or line the cursor is on.

`yw`	copy a word	Copy the word the cursor is on. Move immediately with the cursor to place for the copy and press `p`. The word is inserted after the word the cursor is on.
`yb`	copy to beginning of a word	
`yW`	copy space delimited word.	
`yB`	copy to beginning of a space delimited word	
`yy`	copy a line	Copy the line the cursor is on. Move immediately with the cursor to place for the copy and press `p`. The line is inserted after the line the cursor is on.

`Y`	copy a line	Copy the line the cursor is on. Move immediately with the cursor to place for the copy and press `p`. The line is inserted after the line the cursor is on.
`y`	changed text	Copy following text specified.

`y)` copy the rest of a sentence.
`y}` copy the rest of a paragraph.
`yG` copy the rest of the file.
`ym` `y` followed by a mark copies everything to mark.
`yL` copy the rest of the screen.
`yH` copy to the top of the screen.

Ex Commands

Ex Line Editing Commands and Metacharacters

Input and Change:

`a`	append	Add text after a line. End input with a period on a line of its own.
`i`	insert	Insert text before a line. End input with a period on a line of its own.
`c`	change	Change the line or set of lines. End input with a period on a line of its own.

Delete, Move, and Copy:

`d`	delete	Delete a line or set of lines.
`m`*Num*	move	Move line or set of lines by deleting them and then inserting them after line *Num*.
`co`*Num*	copy	Copy line or set of lines by copying them and then inserting the copied text after line *Num*.

Line References:

Num	Line number	A number references that line number.
Num`,` *Num*	Set of Lines	Two number separated by a comma references a set of lines.
Num`-`*Num*	Range of Lines	Two number separated by a dash references a range of lines.

-*Num*	Offset reference	The minus sign preceding a number offset to a line before the current line..
+*Num*	Offset reference	The plus sign preceding a number offset to a line after the current line..
$	Last Line in File	The dollar sign symbol references the last line in the file.
/*Pattern*/	Pattern Ref.	A line can be located and referenced by a pattern. The slash searches forward.
?*Pattern*?	Pattern Ref.	A line can be located and referenced by a pattern. The question mark searches backward.
g/*Pattern*/	Global Pattern	A set of lines can be located and referenced by a repeated pattern reference. All line with pattern in it are referenced.

Metacharacters

.	Any Character	Matches on any one possible character in a pattern.
*	Repeated Chars	Matches on repeated characters in a pattern.
[]	Classes	Matches on classes of characters, a set of characters, in the pattern.
^	Start of a Line	References the beginning of a line.
$	End of a Line	References the end of a line.
/<	Start of a word	References the start of a word.
>/	End of a word	References the end of a word.

Substitution Command:

s/*pattern*/*replacement*/	Substitution	Locate pattern on a line and substitute pattern with replacement pattern.
s/*pattern*/;/g	Global Substitution on a line	Substitute all instances of a pattern on a line with the replacement pattern.
Num-*Num* s/*pattern*/*replacement*/	Substitution	Perform substitutions on the range of lines specified.

`1,$ s/`*pattern*`/`*replacement*`/g`	Global Substitution on the file	Substitute all instances of a pattern in the file with the replacement pattern.

Replacement Text Metacharacters

`&`	Current pattern	Represents the current pattern matched.
`~`	Previous pattern	Represents the previous pattern matched.
`\U`	Upper case	Changes lower case characters in replacement text to upper case.
`\L`	Lower case	Changes upper case characters in replacement text to lower case.
`\(`*pattern*`\)`	Quoted Pattern	Segments matched pattern into fields that can be referenced in the replacement text using the `&` and a number. `&1` references the first quoted pattern.

Ex and Vi Editing Options

Vi-Ex Editor Options

Option	Abrev	Default	Description
Search Options			
`ignorecase`	`ic`	`noic`	Ignore upper and lower case in searches
`magic`		`magic`	Make metacharacters effective.
`wrapscan`	`ws`	`nows`	Wrap search around to beginning of file.
Display Options			
`number`	`nu`	`nonm`	Line numbers displayed
`list`		`nolist`	Tabs and newline displayed with ^I and `$`
`window`		`window=23`	Set number of lines displayed on screen.
`tabstop`	`ts`	ts=8	Set tab spacing for editor display only.
Input Options			
`wrapmargin`	`wm`	`wm=0`	Insert new line at right margin while inputting
`autoindent`	`ai`	`noai`	Automatically indent: Ctrl -d back indents.
`shiftwidth`	`sw`	`sw=8`	Backtab and line shift spacing.
`showmatch`	`sm`	`nosm`	Show opening (,{,[for closing),},]
`beautify`	`bf`	`nobt`	Prevent input of control characters.

Emacs

Emacs Editor Commands

Cursor Movement:

Ctrl-b		Left one character. (backward to the previous character)
Ctrl-f		Right one character. (forward to the next character)
Ctrl-n		Down one line. (the next line)
Ctrl-p		Up one line (the previous line)
Ctrl-v		forward one screen
Ctrl-z		backward one screen.
Ctrl-l		Move to center of screen.
escape f	Alt-f	Forward one word.
escape b	Alt-b	Backward one word.
escape]	Alt-]	Move to next paragraph.
escape [	Alt-[	Move back to previous paragraph.
Ctrl-a		Move to beginning of a line.
Ctrl-e		Move to end of a line.
escape <	Alt-<	Move to beginning of buffer, usually beginning of file.
escape >	Alt->	Move to end of buffer, usually end of file.
escape *Num*		Repeat the following command Num number of times.
escape x	Alt-x	Move to echo area to enter a command.

Deletions:

DEL	Delete the character before the cursor.
Rubout	Delete the character before the cursor.
Ctrl-d	Delete the character after the cursor.

Kills and Yank:

Ctrl-k		Remove the remainder of a line. (kill the rest of the line).
Ctrl-k Ctrl-k		Remove the remainder of a line and the newline character at the end.
escape d	Alt-d	Remove a word after the cursor.
escape DEL	Alt-DEL	Remove the word before the cursor.
escape k	Alt-k	Remove the remainder of a sentence.
Ctrl-w		Remove a region (delete a block).
Ctrl-y		Insert (yank) the contents of a kill buffer into the text.
Ctrl-x u		Undo the previous command.

Regions:

Ctrl-@ or Ctrl-spacebar			Mark a region (block).
Ctrl-x or Ctrl-x			Exchange cursor (point) and mark.
escape H	Alt-h	mark-paragraph	Mark a paragraph as a region
Ctrl-x Ctrl-p			Mark a page as a region
Ctrl-x h		mark-whole-buffer	Mark the entire text in buffer as a region

Text Format:

auto-fill-mode			Set fill mode option.
escape *num* Ctrl-x `f`		`set-fill-column`	Set position of right margin.
escape q	Alt-q	fill-paragraph	Justify a paragraph.
escape g	Alt-q	fill-region	Justify a region.

Search and Replace:

Ctrl-s	Search for a pattern forwards in the text.
Ctrl-r	Search for a pattern backwards in the text (reverse)
`re-search-forward`	Search for a regular expression forwards in the text .
`re-search-backward`	Search for a regular expression backwards in the text
`replace-string`	Perform a global substitution.
`replace-regexp`	Perform a global substitution using regular expression
`escape %` *pattern* ENTER *replacement* ENTER	Query and replace a pattern.

option-key

spacebar	Replace and move to next instance.
del	No replace and move to next instance.
escape	Quit search-replace operation.
.	Replace and exit.
!	Replace all remaining instances.
^	Move back to previous replacement.

`query-replace-regexp`	Search for a regular expression in query and replace operation

Emacs Window Commands

Ctrl-x 2	Split to new window vertically.
Ctrl-x 5	Split to new window horizontally.
Ctrl-x o	Select other window.
Ctrl-x p	Select previous window.
escape Ctrl-v	Scroll the other window.
Ctrl-x 0	Close current window.
Ctrl-x 1	Close all but the current window.
Ctrl-x ^	Extend the current window vertically.
Ctrl-x }	Extend the current window horizontally.

Emacs File Buffer Commands

Ctrl-x Ctrl-f	Open and read a file into a buffer.
Ctrl-x Ctrl-s	Save the contents of a buffer to a file.
Ctrl-x Ctrl-c	Quit editor.

Ctrl-x Ctrl-v	Close the current file and open a new one. (Visiting a new file)
Ctrl-x i	Insert contents of a file to a buffer.
Ctrl-x Ctrl-q	Open a file as read only. You cannot change it.
Ctrl-x d	Enter the `dired` buffer that has a listing of your current directory. Move to different file and directory names. Display other directories. Select and open files.
`n`	move to next file or directory name.
`p`	move to previous file or directory name.
`e`	If the cursor is on a directory, enter that directory
	If the cursor is on a file, open that file.
`s`	Mark a file for saving.
`d`	Mark a file for deletion.
`u`	Unmark a file for deletion.
`x`	Execute marked files.
Ctrl-x b	Change to another buffer. You are prompted for the name of the buffer to change to. To create a new buffer, enter in a new name.
Ctrl-x k	Delete a buffer (kill a buffer).
Ctrl-x Ctrl-b	Display a list of all buffers.
`escape x buffer-menu`	Select different buffers from a list of buffers.
`d` or `k`	Mark a buffer for deletion.
`u`	Unmark a buffer.
`s`	Mark a buffer for saving.
`x`	Execute marked buffers.

Emacs Help

Ctrl-h Ctrl-h	List of possible help options.
Ctrl-h i	Access the Emacs manual.
Ctrl-h t	Run the Emacs tutorial.
Ctrl-h b	Display keys and the commands they represent.

Vi Quick Reference

Cursor Movement:

h	left
l	right
k	up
j	down
w	forward a word
b	backward a word
W	forward space delimited word
B	backward space delimited word
e	forward to end of next word.
E	forward to end of next space delimited word.
0	start of line .
$	end of line.
-	start of prior line
+	start of next line
(	previous sentence
)	end of sentence, then next.
{	start of paragraph
}	next paragraph.
Ctrl F	Next screen
Ctrl B	Previous screen
Ctrl d	scroll forward one half screen
Ctrl u	scroll backward one half screen
H	to top of screen
L	to bottom of screen
M	to middle of screen
G	last line, specific line
''	to previous cursor position
'm	to line wiht mark m.

Delete:

x	delete character cursor is on
X	delete character before cursor
dd	delete a line
D	delete rest of the line
dG	delete rest of the file
dw	delete a word
d)	delete a sentence
d}	delete a paragraph

Move:

p	insert deleted/copied text
dw p	move a word
dd p	move a line
d) p	move a sentence
d} p	move a paragraph
x p	move a character

Input:

a	append
A	append at end of line
i	insert
I	insert at beginning of line
o	open a line below current line
O	open a line above current line

Change:

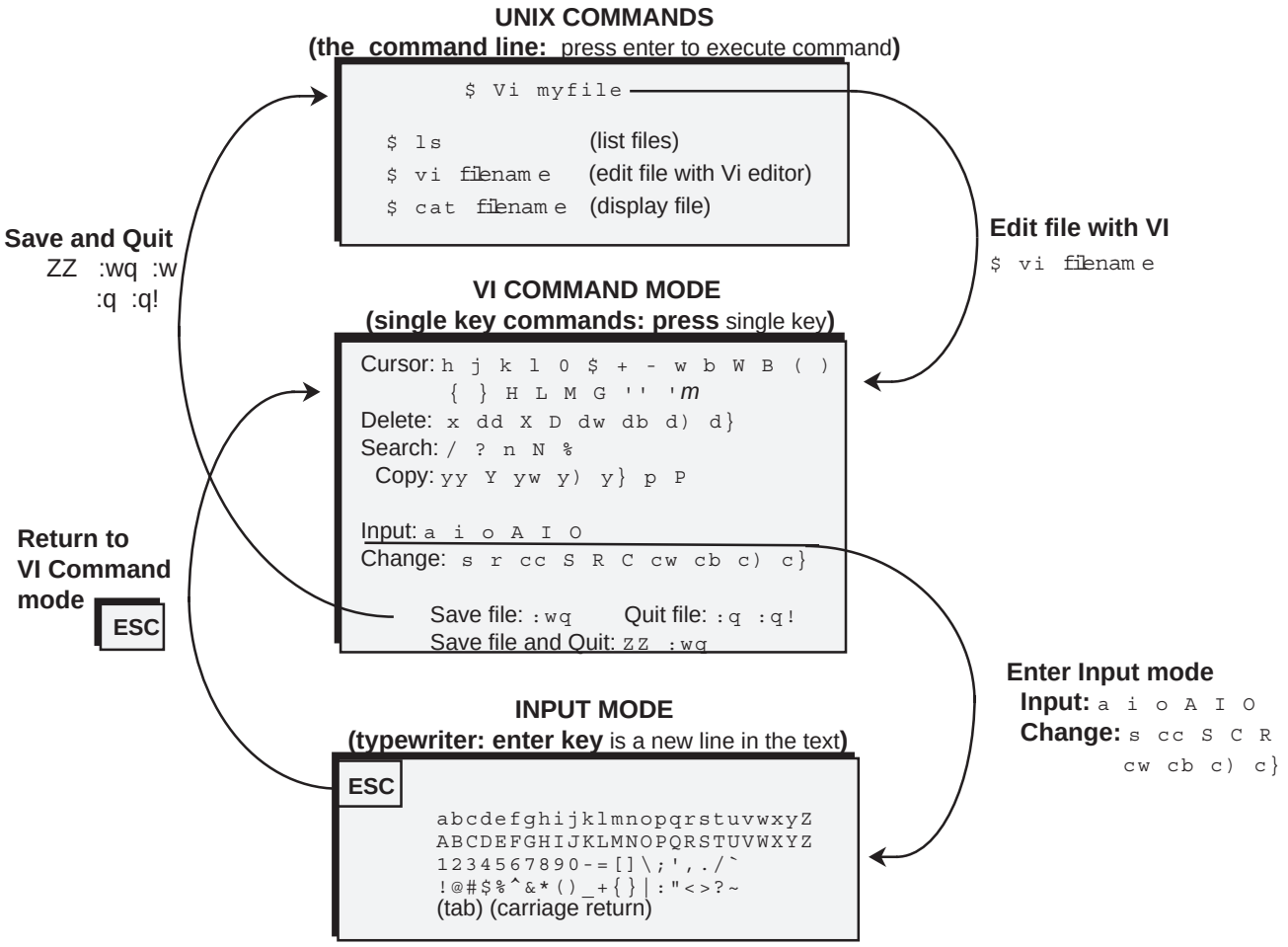
s	change character cursor is on
r	replace character cursor is on
cc	change a line
S	change a line
C	change rest of the line
cG	change rest of the file
cw	change a word
c)	change a sentence
c}	change a paragraph
R	overwrite

Search:

/	Search pattern forward
?	Search pattern backward
n	Repeat search
N	Repeat search in opposite direction

Copy:

yw	copy a word
yW	copy space delimited word
Y	copy a line
yy	copy a line
yG	copy rest of the file
yw	copy a word
y)	copy a sentence
y}	copy a paragraph


UNIX COMMANDS
(the command line: press enter to execute command)
$ Vi myfile
$ ls (list files)
$ vi filename (edit file with Vi editor)
$ cat filename (display file)
Edit file with VI
$ vi filename
Save and Quit
ZZ :wq :w
:q :q!
VI COMMAND MODE
(single key commands: press single key)
Cursor: h j k l 0 $ + - w b W B ( )
{ } H L M G '' 'm
Delete: x dd X D dw db d) d}
Search: / ? n N %
Copy: yy Y yw y) y} p P
Input: a i o A I O
Change: s r cc S R C cw cb c) c}
Save file: :wq Quit file: :q :q!
Save file and Quit: ZZ :wq
Return to
VI Command
mode
ESC
Enter Input mode
Input: a i o A I O
Change: s cc S C R
cw cb c) c}
INPUT MODE
(typewriter: enter key is a new line in the text)
ESC
abcdefghijklmnopqrstuvwxyZ
ABCDEFGHIJKLMNOPQRSTUVWXYZ
1234567890-=[]\;',./`
!@#$%^&*()_+{}|:"<>?~
(tab) (carriage return)

Vi Short Reference

Editor Commands

Cursor Movement:

`h`	left
`l`	right
`k`	up
`j`	down
Ctrl F	Next screen
Ctrl B	Previous screen
`G`	last line specific line

Input:

`a`	append
`i`	insert
`o`	open a line

Search:

`/`	Search pattern
`?`	Search pattern
`n`	Repeat search

Delete:

`X`	delete character
`dd`	delete a line

Change:

`r`	replace a character
`cc`	change a line
`R`	overwrite

Move:

`p`	insert deleted/copied text
`dd p`	move a line

Copy:

`yy p`	copy a line