

Command	Function	Common Options	Example/Notes
<b>General Commands</b>			
ctrl+a	go to beginning of line		
ctrl+e	go to end of line		
ctrl+u	delete text before cursor		
ctrl+k	delete text following cursor		
ctrl+w	delete to beginning of word		
alt+f	go to next word		
alt+b	go to previous word		On a Mac go to Terminal > Preferences > Keyboard and select use Option as Meta key
ctrl+c	stop current command		On a Mac go to Terminal > Preferences > Keyboard and select use Option as Meta key
ctrl+d	log out of current session, similar to exit		
clear	clear shell		
whatis	short description of a command		whatis bc
man	display manual for command		man bc (type q at the : prompt to exit)
date	display date and time		date
cal	display calander	month year	cal 01 2021
pwd	display current working directory		pwd
cd	change directory	.. (up one level), otherwise home directory	cd ..
ls	list files in current directory	-l (list file permissions) <i>directoryname</i>	ls (for working directory) OR <i>ls directory</i> (for any directory on the computer)
mkdir	make new directory		mkdir <i>directory</i>
history	display command history		history
bc	basic calculator		bc (to end, type ctrl-D)
cat	display file contents		cat > <i>file</i> (place input into <i>file</i> , can be used to combine files, eg cat *.fasta >newfile)
more (or less)	display file screenful at time	/pattern (search), q (quit)	more <i>file</i> (use zmore for compressed files)
head	output the first 10 lines of a file	-n (custom number of lines to display)	head <i>file</i>
tail	output the last 10 lines of a file	-n (custom number of lines to display), -f (monitor file in real time)	tail <i>file</i>
wc	word, line, character count	-l (linecount only), -c (character count only)	wc <i>file</i> (line count in column1, word count in column 2, character count in column 3)
diff	compare file contents		diff <i>file1 file2</i>
uniq	return or omit unique lines	-d (see list of duplicate lines)	uniq <i>file</i>
touch	touch a file; create it if it doesn't exist		touch <i>file</i>
open	open a file	-a <i>program</i>	open -a "Microsoft Word" <i>file</i>
cp	copy file	-R (recursive), -X (ignore attributes)	cp -RX <i>dir1 dir2</i>
mv	move or rename file		mv <i>file1 file2</i>
rm	remove (delete) file	-r (remove directory), -f (force remove file)	rm <i>file</i>
paste	paste lines of separate files on the same line of one file		paste <i>file1 file2 file3</i>
echo	write arguments to standard output	-n (don't add new line after input)	echo Hello, world!
	pipe - output from one command to another		for example, see ifconfig command below
tee	capture output at an intermediate step while piping		ls   tee list.txt   tail
df	disc usage	-g use 1 gbyte blocks instead of 512 kbyte	df -g
top	display all running processes		top
ifconfig	configure a network interface		ifconfig   grep "inet" (use to get IP address)
find	search a directory tree for files that match a name		find <i>/path -name filename</i>
vi	visual editor, shell-based text editor		<i>vi file</i> (to edit, type I for insert to exit insert mode hit esc; to save type :w, to quit, type :q)
screen	GNU Screen for running jobs on remote machines	ctrl+a+d (detach window) screen -DR (show detached windows) screen -r (resume a detached window)	screen
./	Execute program		./ <i>program.pl</i>
<b>Text Searching and Manipulation</b>			
grep	search for pattern in file	-r (grep directory), -c (return only count) -o (output matches)	grep <i>pattern files</i> OR grep -r <i>directory</i> (use zgrep for compressed files)
sort	display sorted content	-r sort in reverse order	sort <i>inputfile</i> -o <i>output file</i>
sed	stream editor for transforming text, line-based	-i (make changes to input file), -e (for multiple commands in one line)	sed 's/[[:<:]]old_pattern[[:>:]]/new_pattern/g;' <i>inputfile</i> -o <i>outputfile</i>
awk	pattern scanning and processing language	-F (field separator)	awk 'pattern {action}' <i>inputfile1 inputfile2</i> (optional) -o <i>outputfile</i>
tr	translate, replace simple text like single characters	-d (delete), -cd (delete everything but)	tr -a-z A-Z < <i>inputfile</i> > <i>outputfile</i> (replace all lowercase letters with uppercase)
rev	reverse characters in every line		rev
tac	reverse order lines		
join	join lines of two files based on a common field	-t (field separator) -1 (field 1, etc)	join -t "\t" -1 N -2 N <i>file1 file2</i>
split	split up a file	-l (linecount), -b (size in bytes)	split -l <i>linecount inputfile outputfile</i>
csplit	split up a file by text match	-f (prefix for each new file)	csplit -f <i>prefix file</i> "/pattern1"/" /pattern2"/" /patternN/"
<b>File Compression</b>			
unzip	uncompress a .zip file		unzip <i>file.zip</i>
tar cf	create a single tar file containing multiple files		tar cf <i>newname.tar file1 file2 fileN</i>
tar xf	extract files from a tar		tar xf <i>file.tar</i>
tar cvzf	create a single tar file and compress it		tar cvzf <i>newname.tar file1 file2 fileN</i>
tar xvzf	extract and uncompress files from a tar		tar xvzf <i>file.tar.gz</i>
gzip	gzip compress a file	-d (decompress a gzip)	gzip <i>file</i> OR gzip -d <i>file.gz</i>
<b>Server Access and File Transfer</b>			
ftp	file transfer protocol (non-secure)		ftp <i>username@server.com</i> ( mget <i>file</i> will transfer file to current directory)
sftp	secure file transfer protocol (encrypted)		sftp <i>username@server.com</i>
ssh	secure shell - use to access remote computers and servers		ssh <i>username@server.com</i>
scp	copy something from one machine to another	-r (for copying directories), use . for path to working directory	Copy something from this machine to some other machine: scp <i>/path/to/local/file username@hostname:/path/to/remote/file</i> Copy something from another machine to this machine: scp <i>username@hostname:/path/to/remote/file /path/to/local/file</i>
<b>UNIX Symbols</b>			
>	redirect standard output		
<	redirect standard input		
>>	redirect and append standard output		
;	separate commands on same line		
()	group commands on same line		
/	separator in a pathname		
~	(tilde) your home directory		
.	present working directory		
..	parent of present working directory		
[]	Wildcard match of any one bracketed character in file name		
<b>Regular Expressions (use in grep, sed, awk, and tr, with some exceptions, particularly with tr)</b>			
.	a wildcard character (doesn't match newline).		
*	match the preceding item zero or more times.		
*.	match any item zero or more times.		
+	match the preceding item one or more times.		
?	match the preceding item once or not at all.		
\b	match at either end of word		
\n	newline		
\t	tab		
\	used to negate something else, such as \t would be \t, not tab		
<b>File Permissions</b>			
chmod	change access permissions	Owner, position 1; Group, position 2; Anyone, position 3 Read = 4, write = 2, execute = 1, none=0 1 = execute only; 2 = write only; 3 = write and execute; 4 = read only 5 = read and execute; 6 = read and write; 7 = read, write and execute	chmod 755 (only owner can write, anyone can read and execute) ls -l (list file permissions)
<b>Shell Scripting</b>			
#!/bin/bash	directs shell to bash interpreter		Start shell script with #!/bin/bash. Execute script with <i>./scriptname</i>
#	note		Information following "#" is not executed, except for special #!/bin/bash
var=value	variable assignment	'command' (use backticks to store output from commands as variable)	<i>variable_name=value</i> (to call variable use <i>Svariable_name</i> )
read	prompt for input from command line	-p (display prompt info)	read -p "Please enter value: " <i>value</i> (entry stored as variable value)
if	conditional statements		if [ <i>condition</i> ]; then <i>commands</i> ; elif [ <i>condition</i> ]; then <i>commands</i> ; else <i>commands</i> ; fi
while loops	repeat a task while some value is defined or true		while <i>condition</i> ; do <i>commands</i> ; done