Navigation (cd shortcuts)

`pwd`	Print current location
`cd dir1`	Change directory to *dir1*
`cd ..`	Go on step backwards
`cd ../..`	Go two steps backwards
`cd -`	Go back to the last location
`cd ~username`	Go to username's home folder
`cd`	Go to the current user home folder

List Content (ls options)

`ls`	List the content of current directory
`ls dir1`	List the content of *dir1*
`ls dir1 dir2`	List the content of *dir1* and *dir2*
`ls -l`	List content with long format
`ls -la`	List content with long format including hidden files/directories
`ls -lha`	List content with long format including hidden files/directories and show size in human format
`ls -lhS`	List content with long format, show size in human format and sort by size
`ls -lhSr`	List content with long format, show size in human format and sort by size in reverse order

Files and Folders

`mkdir dir1`	Create a folder called *dir1*
`mkdir dir1`	Create a hidden folder called dir1
`mkdir dir1 dir2`	Create two directories *dir1 dir2*
`mkdir -p dir1/dir2`	Create dir2 including *dir1*
`touch file1`	Create an empty file called *file1* or update the last modification time
`touch .file1`	Create a hidden empty file called *file1* or update the last modification time
`rm file1`	Delete *file1*
`rm file1 file2`	Delete *file1* and *file2*
`rm -r dir1`	Delete *dir1*
`rm -r dir1 file1`	Delete *dir1* and *file1*
`cp file1 dir1`	Copy *file1* into *dir1*
`cp file1 file2 dir1`	Copy *file1* and *file2* into *dir1*
`cp -r dir1 dir2`	Copy *dir1* into *dir2*
`mv file1 dir1`	Move *file1* into *dir1*. If *dir1* doesn't exist, *file1* will be renamed to *dir1*
`cat file1`	View the content of *file1*
`cat file1 file2`	View the content of *file1* and *file2*
`cat -n file1`	View the content of *file1* with line numbers

Wildcards

`*`	Any character any number of times
`?`	Any character but only one time
`{1,4,7}`	1 and 4 and 7
`{3..10}`	Create range from 3 to 10

nano (Basics shortcut)

`nano file1`	Create/edit *file1*
`Ctrl + O`	Save changes
`Ctrl + X`	Exit *nano*
`Ctrl + W`	Cut a line. To paste use *Ctrl + U*

Streams and Redirects

`Command > file`	Redirect command's *stdout* into *file* (default is *1*). If *file* doesn't exist, it will be created on the fly
`Command >> file`	Append *command's stdout* into *file.* If *file* doesn't exist, it will be created.
`Command 2> file`	Redirect command's *stderr* into *file.* If *file* doesn't exist, it will be created.
`Command < file`	Redirect *file* as stdin (input) to *command* (default is 0).
`Command > file 2>&1`	Redirect *command's stdout* and *stderr* into *file.*

Compression & Archiving

`gzip file1`	Compress *file1* - gz format
`gzip file1 file2`	Compress *file1* and file2 - gz format
`gzip -r dir1`	Compress all the file in dir1 - gz format
`gunzip file1.gz`	Decompress file1.gz
`gunzip file1.gz file2.gz`	Decompress file1.gz and file2.gz
`bzip2 file1`	Compress *file1* - bz2 format
`bzip2 file1 file2`	Compress *file1* and file2 - bz2 format
`bunzip2 file1.bz2`	Decompress file1.bz2
`bunzip2 file1.bz2 file2.bz2`	Decompress file1.bz2 and file2.bz2
`zip file1.zip file1`	Compress *file1* to *file1.zip*
`unzip file1.zip`	Decompress *file1.zip*
`zip -r file.zip dir/`	Compress *dir* recursively to *file.zip*
`unzip -l file.zip`	list the content of *file.zip* without Decompressing the file
`unzip file.zip`	Decompress *file.zip*
`unzip file.zip -d dir1/`	Decompress *file.zip* into *dir1*
`tar cf file.tar file1 file2`	Archive *file1* and *file2* - the output file is a tarball called *file.tar*
`tar cf file.tar dir1/`	Archive *dir1*
`tar xf file.tar`	Extract *file.tar*

The echo command

`echo “string”`	Echo the *stdin* “string” to *stdout*
`echo $var1`	Echo the value of *var1 to stdout*
`echo {1..10}`	Echo the range 1..10 to *stdout*
`echo {A..Z}`	Echo the range A..Z to *stdout*
`echo $((3*4))`	Arithmetic Expression - echo the result to stdout

Bash history

`history`	List the last commands for the current user
`!n`	Re-execute command number *n*
`!n:p`	Print command number *n* to the *stdout*
`sudo !!`	Re-execute the last command with sudo

Users and Groups

`adduser user1`	Add a new user called *user1 (this also creates a new group named user1)*
`adduser user1 --gid 1005`	Add a new user called *user1* and add him only to group 1005 (*gid*)
`adduser user1 --no-create-home`	Add a new user called *user1* without home folder
`adduser user1 --shell /bin/sh`	Add a new user called *user1* and set his login shell to */bin/sh*
`adduser user1 group1`	Add *user1* to *group1* (*user1* and *group1* must be existed)

Users and Groups (Cont.)

`deluser user1`	Remove user *user1*
`addgroup group1`	Add a new group named *group1*
`delgroup group1`	Remove group *group1*
`usermod -aG group1 user1`	Add user *user1* to *group1*
`usermod --shell /bin/sh user1`	Change *user1's* login shell to */bin/sh*
`usermod -d /home/user1-home user1`	Change *user1's* home folder to */home/user1-home*
`usermod -dm /home/user1-home user1`	Same as above but also move the directory content
`su user1`	Switch user to user1. Notice: you need to supply user1's password
`su -l user1`	Switch user to user1 and read */etc/profile*, *~/.bashrc*, *~/.bash_profile* and *~/.bash_login* (if the requested login shell is bash)
`sudo su -l`	Become the superuser (root). Notice: you need to supply your password
`exit`	Exit from the current user. This will take you back to the last user.

`sudo visudo`

Linux Permissions

Symbols

`u`	The user who owns the object
`g`	The group which owns the object
`o`	Neither the user nor the group
`a`	All - user, group and the others (ugo)
`r`	Read permission
`w`	Write permission
`x`	Execute permission
`+ / - / =`	Add, remove or set permission(s)

Octal Numbers

`---`	0
`--x`	1
`-w-`	2
`-wx`	3
`r--`	4
`r-x`	5
`rw-`	6
`rwx`	7

`chmod u+x file1`	Add the execution permission for the owner on file named *file1* (or directory)
`chmod g-w file1`	Remove the write permission for the group from file named *file1* (or directory)
`chmod o=r file1`	Set the permissions to read-only for the other on file named *file1* (or directory)
`chmod 777 file1`	Set full permissions for all on file named *file1* (or directory)
`chmod 764 file1`	Set full permissions for the owner, read and write for the group and read only for the others
`chmod +t dir1`	Add the sticky bit permission on *dir1*
`chmod u+s file1`	Set the *suid* permission on *file1* (not relevant for directories)
`chmod g+s dir1`	Set the guid permission on dir1
`chown user1:user1 file1`	Change the user and/or group ownership of *file1* to *user1* and group *user1*

Text Processing

cat file	View the content of *file*
cat -n file	View the content of *file* including line numbers
more file	Display the content of *file* from the first line. You can go down one line by pressing *Enter* or screen by hitting *space*. You can't go up with more.
less file	Display the content of *file* from the first line. You can go one line up or down by using the arrow keys or one screen by using *PgUp/PgDn*. To search within less, use */* followed by the requested pattern. To go to the next match, use *n* and to go to the previous match, use *N*
head file	Print the first 10 lines of *file*
head -*n*	Print only the first *n* lines of *file* (you can specify any number of lines)
tail file	Print the last 10 line of *file*
tail -*n* file	Print only the last *n* lines of *file* (you can specify any number of lines)
tail -f file	Print the last 10 lines of *file* and watch for changes in real-time
tail -f -n *n* file	Print only the last *n* lines of *file* and watch for changes in real-time. You can specify any number of lines including 0 for none – print only new lines.
grep pattern file	Search for the pattern *pattern* in *file.* By default, grep prints the whole line(s) contain the requested pattern.
grep -i pattern file	Search for the pattern *pattern* in *file* but ignore case-sensitive
grep -v pattern file	Search for the pattern *pattern* in *file* but print non-matching lines (**-v** invert-match)
grep -A *n* pattern file	Search for the pattern *pattern* in *file* and print *n* lines after (you can specify any number of lines)
grep -B *n* pattern file	Search for the pattern *pattern* in *file* and print *n* lines before (you can specify any number of lines)
cut -d "d" file -f *n* file	Cut the lines in *file* by delimiter and prints field *n*. You can specify any delimiter as the cutting point but only one character (default is *Tab*). You can print more than one field by using comma (-f 1,3,5)
cut -c *n* file	Cut the *n* character form every line in *file.* You also can specify a range *n1-n2*
sort file	Sort the content of *file* by alphabet order
sort -n file	Sort the content of *file* by numerical order
sort -u file	Sort by alphabet and removing duplicates within a file named *file*
uniq file	Removes any duplicated lines from *file* - matching lines are merged to the first occurrence
uniq -c file	Prints unique lines preceded by the number of the times the line occurs

Regular Expressions

BRE Metacharacters

Character	*Meaning*
`. (dot)`	Any character except line break
`^`	Beginning of the line. When [^] use for negation
`$`	End of line
`[ ]`	Character set - one of the characters in the bracket
`[x-y]`	Character range - any character within the range from x to
`[^ab]`	negation - match any character except the set
`\`	Backslash used for escaping

BRE Metacharacters

Character	*Meaning*
`|`	Alternation - OR operand
`( )`	Contents of group
`*`	Match an element zero or more times
`?`	Match an element zero or one time
`+`	Match an element one or more times
`{m1, m2}`	Match the preceding element if it occurs at least m1 times and no more than m2 times
`grep -E`	Use ERE standard in the grep argument

Searching for Files

which	small tool used for search for programs located
`whereis -b`	search only for binary or executable files
`whereis -m`	search only for man pages
`locate file`	search for a file location
`locate -b "\pattern"`	Match only the base name against the specified patterns
`find`	powerful tool for searching for files and folders [1]
`find . -type f`	Search only for files only, starting at current location.
`find . -type d`	Search only for directories, starting at the current location.
`find . -name file.name`	Search by name
`find . –iname file.name`	Search by name ignore later case (capital end lower case)
`find /etc -maxdepth 1`	limit find only to /etc, not including subdirectories. Higher number will search deeper in /etc
`find /etc -mindepth 2`	option to tell find search only in subdirectories
`find -size -10M`	search for files by size using
`find -size +10M -size -500M`	Search for files larger than 10M and smaller than 500M
`find . -type f -empty`	search for empty files
`find -type f -mtime 1`	search for files that have been modified in the last 24 hours [2]
`find -type f -mmin -40`	*mmin* operator is the same as *mtime* but in minutes
`find . -type f -cmin -3`	*ctime* and *cmin* allow to search files that have been changed **or that their attributes** have been changed.
`find . -type f -amin -2`	search for files by access time - *atime* for days and *amin* for minutes.
`find . -type f -user admin`	search for files and by owner username
`find . -type f -perm 644`	search for files and by permissions, in this case: *rw-r--r--*
`find . -type f -perm /o=w`	search for files and by permissions, in this case write permission for owner
`find -print -exec file {} \;`	search and **run the *file* command** on each of the matches [3]

`find . \( excretion1 \) -or`
`\( excretion2 \)`
	Using **or** operator in fin command.

[1] Note: this line will find all directories and files starting at the root folder and will find all the content of the machin

[2] Note: modify means that the content of the files has been changed (someone edited the file). This doesn't include any changes made in the file attributes.

[3] Note: the **{} \;** excretion is part of the –exec option

Package Management

Update and install

sudo apt update	Update the apt packges [1]
sudo apt install <pkg_name>	Instal apt packeg
sudo yum check-update	
sudo yum update	Update RedHat-style systems packages
sudo yum install <pkg_name>	Install a new package in Red Hat-style systems using the high-level tool yum:

Search and status

dpkg -s <pkg_name>	Package status
rpm -q httpd	To check the status of a package on Red Hat-style systems, we can use the low-level tool rpm
apt search	Debian-style search in the packages metadata
yum search	Red Hat-style search. **yum info** can be used also
sudo dpkg -l	List installed packages in Debian-style
rpm -qa	List installed packages in Red Hat, CentOS, and Fedora.

Deletes

sudo apt remove	deletes the package without any configuration files that were installed
Sudo apt purge	deletes the package and all the configuration files
sudo yum remove	In Red Hat-style, yum remove is **not guaranteed to preserve configuration** files.
sudo rmp -e <pkg_name>	A low lavel tool that keep the configuration files

Controlling Processes

ps	most used command to examine runing processes have invoked from the current terminal
ps x	list all the processes belonging to the current user.
htop	**htop** is the more advantage version of the **top** command (note that you need to install it first).
<process> &	To invoke a process as a background job
jobs	gives us a way to list all the jobs that have been invoked from the terminal, including background job
fg %1	bring the process to the foreground. **%n** is the job number
bg %1	restore the process as a foreground job like **fg**

[1] Note: The available packages' indexes are fetched from the location(s) specified in **/etc/apt/sources.list**

Kill

```
user@linux:~$ kill -l
1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL      10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

`kill -<sig_num> <PID>`	You can send a different signal with the kill command, you need to specify the number of the signal [1]
`kill -2`	Interrupt the process. This is the signal that you send when you press **Ctrl + C** in the terminal.
`kill -15`	This the default signal for the kill command. This signal will kill programs that still “alive” enough to receive signals.
`kill -9`	This signal is unique in the term that is not sent to the target process. The kernel immediately kills the process. Keep in mind that a program can handle signals in different ways, so in case you can't terminate a program by sending INT or TERM signals, use the kill signal instead.
`kill -18`	Continue. If the process is in stop mode and receives a CONT signal, it will continue running as usual.
`kill -19`	Stop. This signal is the same signal that you issue when you press **Ctrl + Z** in a terminal.
`killall -<signal_number> <program_name>`	It’s also possible to send a signal to multiple instances of the same program using the killall command.

***systemctl** (systemd services management)* [2]

`sudo systemctl start app`	start a systemd service
`sudo systemctl stop app`	top a currently running service
`sudo systemctl restart app`	restart a running service

`sudo systemctl enable app`	start services automatically at boot
`sudo systemctl disable app`	disable the service from starting automatically
`systemctl status application`	check the status of a service

`systemctl list-units`	To see a list of all of the active units that systemd knows about
`systemctl list-units --all --state=inactive`	Lists all units and indicate the LOAD, ACTIVE, or SUB states that we wish to see (ex: all inactive units)
`systemctl list-units --type=service`	tell systemctl to only display units of the type we are interested in. Ex: only active service units, we can use

[1] Note: Ctrl + D is not a kill signal. It will send a “end of file” to a process that is wating for user input

[2] Note: If systemctl das not work you need to use the init method by using the **”`service`”** command