note: spaces don’t matter!

note2: ‘directory’ is a synonym of ‘folder’. They are the only things that can have children. All the rest is files.

note3: **ctrl-l** cleans the command window.

note4: **esc-shift-.** inserts the file used in the last command you did wherever your cursor is.

note5: to be able to use the command **install**, which will choose the latest version, you need to have **brew**.

Nano:

**ctrl-o** execute what’s done in nano

**ctrl-x** gets you out of nano

**.** current directory

**..** parent directory

**../..** grandparent directory

**#** *comment* comment

*command* **>** newfile

create a file wherever I am; independently execute the command; and then print the product of it in the file I have just created.

Watch out: It will empty newfile before printing the product from the command!

**echo** *command* **>** newfile

create a file just using redirection without using nano wherever I am currently.

*string* **>>** newfile

append *string* in newfile at the end of the file (it requires complicated commands to edit newfile from the terminal 🡪 which is why we use nano!)

if I write ‘>’ instead of ‘>>’, it will erase what was before in newfile and replace it by *string*.

**alias rm=’rm –i’**

create an alias for ‘rm –i’. everytime I’ll type ‘rm’, it will execute ‘rm –i’.

**cat** *file*  show text content of a file

**cd** *folder* change directory

**cd /***folder*absolute path (when no ‘/’ it means relative path). So if I want to change directory with the absolute path, I have to write the whole directory, wherever I’m located.

**cd ..** go back to the parent of the folder I’m in.

**cd ../..** cd back 2 directories

**cd** (without anything behind) brings you back to your root folder

**cd ~** same

**cd -** going back to the previous directory

**cp** *file1 file2* copy file1 and name this copy file2

**echo** *command*

show what the command would do without executing it. For ex,

*echo ls F\** will output *ls* and then expand what F\* contents.

**file** *folder* determine type of file

**ls** list content

**ls –F** list content with ‘\’ if the file is a folder

**ls –l** list content with size of each folder

**ls -lh**

**ls** *folder* list content of the file without changing directory

**ls** F\* list all files where I am starting with F

**man page** *command*

Documentation about the command (‘q’ to get out)

**/**-f will look for ‘-f’ in the manual.

**mkdir** *folder* create a folder in the directory I’m in

**mv** *file1 file2*rename file1 into file2

**mv** *file* move the file in the current directory

**open** open a file

**pwd** present working directory

**rm** *folder*remove directory (won’t work unless it’s empty)

**rm –r** *folder* remove directory recursively, ie means “remove everything that is inside the folder”, it’s a repetition process of deletion.

**rm –rf** *folder*remove directory and *–f* implies that you don’t want him to ask you for confirmation.

**rm -rf­** is the same as **rm –r –f**

**rmdir** *folder* delete a folder in the directory I’m in (works only if the folder is empty).

**touch** *file* create an empty file

**wc** *file* get info about a file

**wc –l** *file* how many lines in the file

**wc –l \*.txt** show number of lines of all txt files

**wc –l \*.txt | sort**

sort the output of ‘wc –l \*.txt’ per number of lines

**wc –l \*.txt | sort –r | head**

gather files by their number of lines

Q1: difference between > and >>?

Q2: how does he know where to create the file? Wherever I am currently.

Q3: why was it executed since I’ve used echo?

$ echo ls *#will just print whatever comes after as a string*

ls

$ echo ls > file *#> takes me out of echo*

It will create a file wherever I currently am and print “ls” in it.

$ ls > file

It will create a file and print all the content of where I am in the file.

**Writing a program (ie a bash program):**

$ echo *command* > *filename* *#will create a file wherever I am and print the command in it.*

$ bash *filename* *#will run line by line what is in filename*

Advantage of nano:

$ nano *filename*  *#open filename in nano*

I can edit whatever and however I want in the file. nano is a text editor, which directly answers to the terminal, unlike another text editor that I would have to call from the terminal with open.

$ bash *filename*

Notes:

. stand for current directory, meaning that the command cd . says move me where I am, ie. do nothing.

.. stands for parent directory, which is the directory where current directoryis...

You can go to the parent directory using cd .. and delete my\_test\_directory using the command rm my\_test\_directory. Oups ! It doesn't work because you must specify an option to rm by adding -r so it goes recursive on the indicated file since it's a folder.

Every terminal command has a help page, called man page. Check it for [rm](http://www.linuxmanpages.com/man1/rm.1.php), [ls](http://www.linuxmanpages.com/man1/ls.1.php), or any new command you want to know more about.