1. File organization.

Programs and data are stored in documents (files). Files are stored in folders, called directories under Unix. Directories on their own can be put into other directories to create a (tree-like) branch structure. The top of this structure is formed by the home directory (Unix), which is the highest directory for any user; it is indicated with $HOME (depending on the system administration it can be stored either at the /users, /home or /Users disc). Also present at all times is the temporary directory (/tmp) in Unix, where temporary (administrative) files are stored. This /tmp directory should always, at all times, have enough space for system processes to write files, otherwise the machine will stop functioning correctly.

If no directory name is given for a certain file, then it is assumed that it is present in the current directory, the so-called working directory. When referring to a file in another directory, then the file name should be preceded by one or more directory names, which are connected by a slash "/" (e.g. /users/marcel/calcs/example). If no home directory is given (e.g. calcs/example), then the filename refers to a relative position with respect to the current working directory. The current directory is indicated by ./, for instance ls is the same as doing ls .; one directory higher is indicated by ../ (and can be combined in more than one, as in ../../); going one directory up is indicated by cd ../.

2. Some Unix commands.

cd	"Change directory", bv "cd ../davide" goes one directory up and then to the directory davide (if it exists).
ls	Gives a list of all files and directories within the current directory. Can also be used in combination with an explicit directory-name (e.g. ls /users/marcel/calcs).
mkdir mynew	Create new directory mynew within the current directory (as indicated above, this is the same as mkdir ./mynew). Can be done with relative paths, for instance for absolute paths (mkdir /users/marcel/calcs/tfg2021) or relative paths (e.g. mkdir ../tfg2021).
vi myfile	The general command to view and edit files. With this command files can be viewed, created, adapted. Can be used for multiple files (e.g. vi file1 file2 file3), and move between the files with :n (next). See p. 2 for vi commands.
cat myfile	Used to show the contents of a file called myfile.
head -n 3 myfile	Show the first 3 lines of the file myfile.
tail -n 3 myfile	Show the last 3 lines of the file myfile.
cp filein fileout	The copy file command. With this command files can be copied to a new file (e.g. cp file1 file1nw), or to a different directory (e.g. cp file1 file2 diffdirectory).
mv filein fileout	The move file command. With this command files can be renamed.
rm myfile	Removes file myfile. For removing directories, add -r option (rm -r mydir). The -f option forces the removal of protected files, the -i option asks for every file whether it should be removed.
chmod a+rx script	Make a file script readable and executable for all users (users can be divided in user (yourself), group, and others).
chmod a-w script	Protect a script from being overwritten accidently, for all users.
man comehere	Gives a description (the manual) of the command comehere, if available in the standard Unix manuals. Is very useful for information on what it does, how to use the command, what are the options, etc.

3. Environment variables and aliases.

Already indicated above was the use of $HOME as a shortcut for your home directory (e.g. /users/marcel); this is called an environment variable. Another useful variable is $PATH, which includes all directories to look for scripts and commands (the different directories are separated by a colon :, as in export PATH=/opt/local/bin:/opt/local/sbin:/Users/swart/bin). These environment variables are used extensively on Unix and on the computer clusters, because they allow for dynamically getting information. A third one could be the one related to the prompt which shows you in which directory you currently are: export PS1="\[\e[95;38;5;208m\]\u@\h \w\[\e[0m\] >> ", with which user account and on which machine.
A different way of making shortcuts, is by creating an alias, such as alias lt='ls -ltra'. If you have put this in your $HOME/.bashrc file, the Unix system recognizes it and you can use lt as if it were a native Unix command. The $HOME/.bashrc file contains all kinds of information which the operating system needs to serve you well, according to your preferences. Note that this file is specific for the Bo(u)rn(e) Again Shell (bash) shell, which is the most commonly used shell under Linux/Unix (although Apple in the latest operating systems moved to the more secure Z-shell, zsh).

4. Some vi (or vim) commands.

Escape	Make vi listen
^L or ^R	Redraw the screen (^ stands for {Ctrl-} plus another key, i.e. ^L is {Ctrl-L})
:q	Quit (without saving changes)
:q!	Quit (and ignore any changes made)
:w	Save changes made in the file
:wq (or ZZ)	Save your changes and Quit

-	Move up one line
return	Move down a line
space	Move forward one character
backspace	Move backward a character
h j k l	Move left, down, up, and right (respectively)
	(or arrow-keys, depends on computer, shell, machine you are using)
$	Move to end of a line
0	Move to beginning of a line
b	Move to beginning of current word
e	Move to end of current word
G	Go to the end of the file (last line)
1G	Go to the begin of the file (first line)
:33 <enter>	Go to line 33

H	Move to top of screen
M	Move to middle of screen
L	Move to bottom of screen
^F	Move forward a full screen {Ctrl-F}
^B	Move backward a full screen {Ctrl-B}
^D	Scroll down a half screen {Ctrl-D}
^U	Scroll up a half screen {Ctrl-U}

i	Insert text before current character
a	Insert text after current character
I	Insert text at beginning of line

A	Insert text at end of line
o	Open a new empty line after current line
O	Open a new empty line before current line
cc	Change the current line
cw	Change the current word
r	Replace current character by overwriting
R	Replace current and more characters by overwriting
x	Delete current character
X	Delete previous character
dd	Delete the current line
D	Delete rest of line including the current character
d/Something	Delete all lines until you find the text “Something”.
"ay55y	Place (**yank**) 55 lines in the buffer called **a**, including the current line
"cy/Ab<ENTER>	Place (**yank**) all lines in the buffer **c** until you find the text **Ab**
"ap	Print after the current line the contents of buffer **a**
	(note: these buffers work over different files, see below)
u	Undo the most recent change, including another undo
U	Will restore the current line to the way it was when you last came to it
.	Repeat last command

5. Find and replace.

:g/XY/s/XY/AB/g	For all lines replace text **XY** by **AB** (every time it is found on the lines)
:g/xY/s/xY/Ab/	For all lines replace text **xY** by **Ab** (first time it is found on each line)
:/Xy/s/Xy/aB/g	For the next line where **Xy** is found, replace **Xy** by **aB** (every time it is found on the line)
:/xy/s/xy/ab/g	For the next line where **xy** is found, replace **xy** by **ab** (only the first time it is found on the line)

6. Viewing several files.

vi File1 File2 File3

:n	Go to next file
:wn	Save changes and go to next file
:n!	Go to next file without saving changes

Note that you can **yank** lines within one file, go to the next one, and paste them there. The amount of lines that are coming along is often limited to 50 lines.