1 UNIX & Emacs: Controlling the Machine

If you are taking this course probably this is your first time confronted to computers using operative systems other than Windows or MAC. Probably you don't have enough working experience to recognize the meaning the following words: Unix, Linux, Ubuntu, GNU. If you do, congratulations! this part of the course should be easier for you.

UNIX[1] is the name of a family of operating systems very common in academic circles and high performance engineering. If you use a commercial email provider, you can be sure that the messages you received today were handled by computers under some flavor of UNIX. If you are thinking about going into the research path of astrophysics, high energy physics, earth sciences, computational physics or theoretical chemistry (just to name a few) you are bound to make use of systems under UNIX. It could be either GNU/LINUX or MAC operating systems.

The purpose of this document is to give you a starting point and get you working as soon as possible in UNIX systems.

1.1 The Console

In UNIX environments you have to learn to control the machine. This can be done with text. That actually means typing text that can be understood by the machine.

The `Terminal` is the place to do that. Type text and feed it into the machine. As a result you might get it to do what you need.

Let's assume for a moment that you have logged on and have a terminal open. There must be a cursor showing the place where the text will be written. It looks like this:

```
forero@compufis:~>
```

Now write the following existential text and hit the return key

```
forero@compufis:~>whoami
```

The terminal will reply back to you by telling you the username you have. If its different from your username it means that you are logged into somebody else's account. In my case I get in the terminal:

```
forero
forero@compufis:~>
```

[1] Wikipedia says: Unix (officially trademarked as UNIX, sometimes also written as Ux in small caps) is a multitasking, multi-user computer operating system originally developed in 1969 by a group of AT&T employees at Bell Labs, including Ken Thompson, Dennis Ritchie, Brian Kernighan, Douglas McIlroy, Michael Lesk and Joe Ossanna.

and the terminal is ready to accept more commands.

This is a summary of the commands you will find yourself using most of the time. Each command includes a brief explanation of its meaning and some useful options to that command. To fully understand them you have to explore by yourself.

Directories

- `pwd`: present working directory
- `ls`: list directory contents. `ls -l` shows ownership and size of each file.
- `mkdir dirname`: create a new directory `dirname`
- `cd dirname`: change your directory to `dirname`.
- `rmdir dirname`: remove the directory `dirname`. The directory has to be empty. Use `rmdir -r dirname` to recursively remove `dirname`.

Files

- `touch file.txt`: changes the timestamp in `file.txt`. If `file.txt` doesn't exist, it will create it.
- `mv file1.txt file2.txt`: renames `file1.txt` as `file2.txt`. It will overwrite `file2.txt`, if it already existed.
- `mv file.txt dirname`: moves `file.txt` into `dirname`.
- `cp file.txt file2.txt`: copies the contents `file.txt` into `file2.txt`. It will overwrite `file2.txt`, if it already existed.
- `rm file.txt`: deletes `file.txt`
- `tar -cvf compressed.tar files`: used to compress the `files` into `compressed.tar`
- `tar -xvf compressed.tar`: used to expand the file `compressed.tar`.
- `gzip file.txt`: compresses `file.txt` into `file.txt.gz`
- `gunzip file.txt.gz`: expands the file `file.txt.gz`.

Other useful commands to manipulate file content

- `less file.txt`: allows a quick view of `file.txt`.
- `head file.txt`: first ten lines in a file.
- `tail file.txt`: last ten lines in a file
- `cat filename1 filename2`: concatenate and print files.
- `tac filename1 filename2`: concatenate and print files in reverse.

- `grep pattern file.txt`: print lines in `file.txt` matching `pattern`. It is case sensitive.
- `wc file.txt`: print line, word and byte counts in `file.txt`.

More details on grep

- `grep -o partofline file.txt`: prints the subsection in the line that matches `partofline`
- `grep -i string file.txt`: selects the case insensitive search
- `grep -iw word file.txt`: selects the line that matches the word "word".
- `grep -v pattern file.txt`: prints the lines that do not match `pattern`.

Another useful command to do quick hacks is `awk`. It is very useful to deal with files with columns.

- `awk '{print $1,$2;}' file.txt`: prints the columns 1 and 2 from `file.txt`.
- `awk '{if($3>30)print $0}' file.txt` : prints all the lines that have the values in the column 3 greater than 30
- `awk '{if($2=="Bogota")print $0}' file.txt`: prints all the lines that have the word `Bogota` in the second column.

These commands are even more useful when you use wildcards.

- `ls *.py`: lists all the files ending in `.py`.
- `mv *.py dirname`: moves all the files ending in `.py` into dirname

External world

- `ssh machinename`: connects to another machine.
- `wget https://www.host.com/file.txt`: grabs the `file.txt` from the server `www.host.com`.
- `scp username@machine:file.txt ./`: copies `file.txt` from `machine` to the local directory.
- `rsync username@machine:file.txt ./`: copies `file.txt` from `machine` to the local directory. It can resume the transaction if it dropped before.

System

- `top`: Shows the resources in your machine.
- `ps -eaf` : Shows the processes running in the machine.
- `kill -9 PID`: Kills a process.
- `chmod [ugo+-rwx] file.txt`: chages read/write/execution permission.

Redirecting output

- `>` Connects a command to a file. Redirects the output of the command on the left to the file on the right. This overwrites the contents in the file
- `>>` Same as the previous command with the difference that the output is appended at the bottom of the file.
- `|` The Pipe. Connects two commands. Redirects the flux of characters to feed other commands.

Useful commands and concepts

- `man command`: shows the manual for `command`.
- Tab completion: Hit Tab in the middle of a command/filename and it will give you options to complete it.
- `history`: lists the latest commands.
- `Ctrl-r`: reverse search in the command history.

1.2 emacs: a preferred text editor

```
emacs new_file.txt &
```

- `Ctrl-x-s`: Save
- `Ctrl-x-c`: Close the editor
- `Ctrl-k` cut the text in the line after the cursor
- `shift-arrow` select text
- `Ctrl-w` cut highlighted text
- `Ctrl-y` paste
- `Ctrl-_` undo
- `Meta-x query-replace` query replace.

Other common options for people writing code are vim (UNIX) and TextWrangler (MAC).