Linux Basics — TTM4175

David Palma

1 Introduction to Linux

1.1 Exercises

Connect your Raspberry PI to the monitor and keyboard but **do not** connect the mouse nor the network/ethernet cable. Finally, connect the power cable (micro-USB) to your Pi.

These exercises should be completed in teams.

1.1.1 Exercise 1 — First boot

First boot and basics.

1. Type on your keyboard the keys `Ctrl+Alt+F2` (i.e. press `Ctrl`, `Alt` and `F2` simultaneously).

> Your display should now show a black and white command line interface, a terminal or virtual console, with a login screen.

2. Log in using "pi" as the username (type and press `enter`) and the password "reactive4115".

> Password inputs in Linux are usually "silent" (i.e. don't worry if you don't see your password being typed in!).

Welcome to `bash`! One of the most famous Linux shells, an interactive terminal where we can run multiple commands or programs. A **Unix** shell is a user interface (usually interactive) to interpret commands and give instructions to the operating system using **GNU** or other software. It can also be seen as a programming language but we won't be looking at that. The `bash` is the Bourne Again Shell, an evolution from `sh` or the **Bourne Shell** (from Stephen Bourne) and there are several others (e.g. `zsh` is a very popular choice too).

---

3. Print "Hello World!" in your terminal by typing the command `echo 'Hello World!'` followed by the `enter` key.

> All commands should be followed by the `enter` or carriage return key.

```
1
2
3
4
5
```

4. Alternate between terminals using `Alt+F3` (F4, F5, ...) or `Alt+Left/Right arrows` and login again.

> Linux is a multitasking/multiuser operating system which means that it allows several processes and users to work simultaneously.
>
> **Challenge:** enter the command **`sleep 60`**, notice how the terminal is occupied, type `Ctrl+z` to move it to the background and **`fg`** to bring it back to the foreground (you can use this with any command and alternatively you may cancel the **`sleep`** command with `Ctrl+c`).

5. Logout from one of the terminals using the commands **`logout`**, **`exit`** or by pressing `Ctrl+d`.
6. Linux is case sensitive, meaning that the command **`echo`** cannot be typed as `Echo`. What happens if you enter the command `Echo`?

```
1
2
3
```

7. When typing a command, **`bash`** (the shell we are using), allows saving time with shortcuts such as the `Tab` key to auto-complete and with `Ctrl+a/e` commands and move in the command line. Type `ec+Tab`. What happens?

```
1
2
3
4
```

8. Press the `up` key one or more times. What happens?

> With this shortcut you can press the `enter` key to confirm your command and `Ctrl+c` to cancel.

```
1
2
3
```

9. Press `Ctrl+r` followed by "Hello" and `enter`. What happens?

> Press `Ctrl+r` multiple times to find other matches, `enter` to confirm, `Ctrl+c` to cancel and the arrow keys to edit.

```
1
2
3
```

10. An important aspect of GNU/Linux is the availability of documentation for all (or almost all) installed commands. Type `man bash`, scroll through using the arrow or space keys, pay attention to the **definitions and reserved words**, leave by pressing `q`.

> **Challenge:** A command similar to `man` but with typically much more detail is `info`, learn how to use it with `info info`.

11. Almost all commands in Linux support the argument "--help" or the option "-h" for a very brief explanation. Try `bash --help`.

> Arguments and options are additional keywords given to commands.

To sum-up, the simplicity of Unix systems, or the **Unix philosophy**, aims at systems composed of programs that:

1) are simple and elegant (do one thing and do it well);
2) work together;
3) use universal interfaces (e.g. text).

1.1.2 Exercise 2 — File Operations

1. In a new terminal log in again with the user `pi`.
2. List all the files in the current directory by issuing the `ls` command.

```
1
2
3
4
5
6
```

3. Find additional options for the command `ls` (use `--help` or `man ls`).

```
1
2
3
4
5
6
```

4. Identify the current working directory using the command `pwd`.

> In Linux files and directories are organised in a hierarchy, or in a tree, and the root of the tree is represented by "/". This tree is actually a way of representing a disk or disk partitions from a computer, and we can have several ones (typically found in `/dev/`, such as `/dev/sdb`), *mounted* in our system.

```
1
2
3
4
5
```

5. Change your current directory by using the `cd` command and check your new working directory again with `pwd` (e.g. `cd Desktop`).

```
1
2
3
4
5
```

6. Return to the previous directory using `cd ..`

> If we use `ls -a` we'll see *all* existing files and folders, even the hidden ones, as well as the `..` which corresponds to the parent folder, or the folder that contains our current location (the `.` is our current location).

7. Enter the command `cd notworking`. What happens?

```
1
2
3
4
5
```

8. Enter the command `cd ..`, repeat the `pwd` command, what's the result?

```
1
2
3
4
5
```

9. Enter the command `ls`. What do you see?

```
1
2
3
4
5
```

10. Enter the command `cd`. What's your current directory?

> Users have a home folder, which generically corresponds to their root, and they can easily navigate there with the shortcut `~`.

```
1
2
3
4
5
```

11. Enter the command `cd ../../`. What's your current directory?

```
1
2
3
4
5
```

12. Enter the command `cd root`. What's the result?

```
1
2
3
4
5
```

13. Enter the command `cd ~`. What's you current directory?

```
1
2
3
4
5
```

14. Create a new directory, or folder, called “test” with the command `mkdir test`. Change your working directory to this folder.

```
1
2
3
4
5
6
7
```

15. Create two empty files with the command `touch` (e.g. `touch file1`) and verify with `ls`.

```
1
2
3
4
5
6
7
```

16. Remove one of the files by issuing the command `rm` and then verify again.

> **Note:** the file has been effectively deleted (or unlinked to be precise), there’s no trash or recycle bin, so take care or use the option “-i” to enable confirmation before deletion.

```
1
2
3
4
5
6
7
```

17. Go to the parent directory (i.e. the previous folder), confirm your location (you should see the “test” folder you created).

```
1
2
3
4
5
6
```

18. Attempt to delete the "test" folder with `rmdir test` (remember we still have a file inside this folder). What happens?

```
1
2
3
4
5
6
```

19. Delete the "test" folder by first removing the remaining file with `rm ./test/file2` and then using `rmdir`.

> **Note:** if we are sure that we want to delete everything inside a directory we can use the command `rm -r` but it's dangerous because it removes everything inside that folder. Moreover, if by mistake we delete system files/folders we may end up with a destroyed system.

```
1
2
3
4
5
6
```

20. Create a new file named "notaudio.mp3"

```
1
2
3
4
5
6
```

21. Create a copy of the new file in the folder "Music" using the name "na_copy.mp3", by typing the following command `cp notaudio.mp3 ./Music/na_copy.mp3`. Verify that it is created in the right folder with the right name.

```
1
2
3
4
5
6
```

22. Move the file “notaudio.mp3” to the “Music” folder using the command `mv`.

> When moving or copying files, if we only specify the destination and omit the name, the origin name of a file/directory will be used (if a different path is used).

```
1
2
3
4
5
```

23. Rename the file “na_copy.mp3” to “notaudio2.mp3” also by using the command `mv`.

> **Hint 1:** Don’t forget to use the `Tab` key to speed up your typing.
>
> **Hint 2:** `cp` and `mv` can also use the option “-i”, which is helpful for avoiding overwriting already existing files in the destination target (similar to the `rm` command for confirmation).

```
1
2
3
4
5
6
```

1.1.3 Exercise 3 — Viewing and editing files

View the content of files in the terminal and/or edit them.

1. Print the output of the system’s log, which is stored in “/var/log/messages”, using the command `cat`.

> Due to the importance of this file, its contents can also be seen by simply issuing the command `dmesg`. To scroll up and down we can use `Shift+pgup` and `Shift+pgdown`.
>
> An alternative to `cat` is `nl`, if we want to see line numbers in the output.

```
1
2
3
4
5
6
```

2. To determine how many lines a file has the command `wc` can be used. How many lines does your log have?

```
1
2
3
```

3. To print the first or last few lines of a file the commands **`head`** and **`tail`** can be used respectively. How many lines do you see?

```
1
2
3
4
```

4. Open the log file with the viewer **`less`** and find the available options by pressing `h`. How do you exit the viewer?

```
1
2
3
4
```

5. Inside **`less`** search for the line in the log file that contains the text “Linux version”. What’s the line number and Linux version?

> **Hint:** Use **`less`** `-N` and its search functionalities. Replace *pattern* by “Linux version”.

```
1
2
3
4
```

6. Search for the total number of processors mentioned in the log file using the pattern "Total.*processor". How many processors are activated?

> **Note:** This exercise is using a regular expression pattern, matching all strings that contain “Total” and “processor” with anything in between.

```
1
2
3
4
```

7. Instead of viewing or printing entire files `grep` can be used to efficiently search for patterns inside files. Search for the total number of processors using `grep`. What option can you use to also print the line number?

```
1
2
3
4
5
6
7
```

8. Knowing the line number of interest, it's easy to return to `less` and verify surrounding context by typing `:` followed by the line number. How can we use `grep` to also show this context? For example 5 lines before and after the line of interest?

```
1
2
3
4
5
6
7
```

Other widely used programs are `sed` and `awk`. They are both quite powerful and allow editing the files we are working with. In particular, `sed` is a typical solution for finding and replacing parts of a file **without having to manually edit** them, while `awk` implements the AWK programming language and is more suited for managing files organised in columns or using specific separators. Remember that `grep` is the best option for extracting text but we can also return to the log file example using both `sed` and `awk`:

```
sed -n '/total.*processor/Ip' /var/log/messages
```

or

```
awk '/[tT]otal.*processor/' /var/log/messages
```

Apart from simple command line tools for viewing and editing files based on filters and expressions, there are also several Command Line Interface (CLI) editors that can be used, examples are `vim`, `nano` and `emacs`.

---

9. In your home directory enter the command `nano myfile.txt`, which will open the editor `nano` and type in "This editor is great!". Save and exit `nano`.

> **Hint:** You can get help and a list of shortcuts by pressing `^G`. This means pressing `Ctrl+g` and `M-r` means `Meta+r` (the meta key is usually the Alt key).

```
1
2
3
4
5
```

10. Open again the same file with `nano` and press `Ctrl+\`, type "great" and press `enter`, type "ok" and `enter`, press "y". What happens? Exit `nano` (choose to save or not).

```
1
2
3
4
5
```

11. Open "myfile.txt" with the command `vi`, press `i` to enter the *INSERT mode*, edit the file (write something) and press `Esc Esc` to return to the *COMMAND mode*, followed by `:q` to exit the editor. What happened? Follow the editor's suggestion to exit without writing the file.

```
1
2
3
4
5
```

1.1.4 Exercise 4 — Linux Permissions

Permissions, users and groups.

In Linux each file has a set of permissions or access mode (read, write and execute), as well as an owner and group owner. This is one of the reasons why Unix/Linux are also considered to be secure, providing quite a robust line of defence that separates files and folders from users, groups and applications. Nowadays most operating systems have something like this, or similar.

---

1. Try to list the contents of the directory in "/root/" using the user "pi". What happens?

> With the command `ls -ld /targetdir/` you can verify the permissions of a directory (first column) as well as its owner (3rd col.) and group (4th col.)
>
> You should see something like: `drwx------ 2 root root 4096 Aug 27 14:00 /root`. In the first col. the "d" stands for directory and the remaining 9 characters are the permissions for the user ("rwx"), for the group ("---") and for others ("---"), explained after the next exercise.

```
1
2
3
4
5
6
7
```

2. Find your username and associated groups using the commands `whoami` and `groups`

```
1
2
3
4
5
6
7
```

With the information about users, groups and permissions we can better manage who accesses what and prevent mistakes or malicious actions. To change permissions we can use the command `chmod` followed by a flag for the user (**u**), group (**g**) and others (**o**), a sign for adding or removing permissions (**+** and **-** respectively), and the permission we want to edit: read (**r**), write (**w**) or execute (**x**).

In all Linux machines there is a super user, or "root". This user can access all the directories and files from other users, regardless of the permissions, and even change/create new users and change their passwords. To run commands with administrator or root privileges we use the command `sudo` or change our terminal with `su` and run multiple commands as root. The command `su` allows us to change our current terminal not only to root but also to any other existing user, provided we know the login name and password (e.g. `su - palma`). However, if we are the super user, we can simply change user without need for any password (**Remember:** with great power comes great responsibility).

---

3. Run the command `sudo whoami`. What's the result? What does this mean?

```
1
2
3
4
5
6
7
```

4. List the contents of the directory in `/root/` using `sudo`. How many files are in that directory?

```
1
2
3
4
5
```

5. Change the permissions of the folder "/root/" with the command `chmod o+r` so that other users can read the content of the folder. Check the new permissions with `ls -ld`. List its contents **without** using `sudo`.

> **Note:** The new permissions include an "r" in the permissions for others.

```
1
2
3
4
5
6
7
```

6. Change your current user to root with the command `sudo su` and confirm it with `whoami`.

> **Note:** You should be very careful when working as root and should avoid it unless it is completely necessary!

```
1
2
3
4
5
6
7
```

7. Revert the permissions of the `/root/` folder, without using **`sudo`**, so other users cannot read it ("o-r"). Exit the "root terminal" by entering the **`exit`** command in the terminal.

```
1
2
3
4
5
```

1.1.5 Exercise 5 — Personalisation

1. Change the default password (**`passwd`**) of the user "pi".

> **Milestone:** make sure you have changed the password before continuing (e.g. login in another terminal using the new password). If you skip this step your Raspberry Pi will be **vulnerable to remote attacks** as soon as we connect it to the Internet.

```
1
2
3
4
```

2. Create a new user (**`useradd`**) using a login name of your choice, while also including the user in the already existing groups "dialout,audio,video". Alternatively, create a user and add it to these groups afterwards (**`usermod`**). Don't forget to set the password for the new account! What options did you use?

> **Hint:** remember to use Linux's existing help commands and that you'll need administrator privileges.

```
1
2
3
4
5
```

3. Change terminal and login with the new user name.

```
1
2
3
4
5
```

4. Add the new user to the list of *sudoers*
    - Return to the first terminal;
    - Add write permissions to the file **010_pi-nopasswd** `/etc/sudoers.d/` (**`chmod`**);
    - Edit the file to include your new user;
    - Remove the write permissions (**`chmod`**) and return to the previous terminal.

```
1
2
3
4
5
```

5. Change your *hostname* to match your team number (e.g. "Team1"), based on the following steps:
    - find and edit the files **hostname** and **hosts** in "/etc";
    - replace any entries with your current *hostname* with the new one;
    - reboot the system with the **`reboot`** command or with **`shutdown -r`** .

> With the **`shutdown`** command we can turn off the system completely, or halt, by typing **`shutdown -h now`** , reboot, schedule a shutdown or cancel a scheduled shutdown.

```
1
2
3
4
5
```

6. How could you repeat the previous step without manually editing each file? (answer either by explaining the overall idea or with the actual command(s) you would use)

```
1
2
3
4
5
```

7. What would happen if you added "ntnu.no" after your hostname in your hosts file?

```
1
2
3
4
5
```

1.1.6 Exercise 6 — Networking

Connecting the Raspberry Pi to the Internet.

1. Verify your system's IP addresse(s) by issuing the command `ip address`. How many interfaces do you see? Note the information for "eth0".

> **Hint:** Existing interfaces are numbered and named (e.g. "1: XXX:"). IP(v4) addresses and the respective mask follow the format XXX.XXX.XXX.XXX/YY.

```
1
2
3
4
5
6
7
8
```

2. What's the route your Raspberry Pi will follow to reach the Internet? Find out with `ip route`.

```
1
2
3
4
5
6
7
```

3. Connect the ethernet cable to your Raspberry and note the changes in your IP addresses and routes.

> **Note:** By default, the Linux distribution we are using has a running service, or daemon, that automatically configures network interfaces whenever a new link is found. This service is called `dhcpcd` and runs a networking protocol called DHCP.

```
1
2
3
4
5
6
7
8
```

4. Check if you have Internet connectivity using the command `ping` (stop it with `Ctrl+c`). This will use the ICMP protocol to contact an Internet server of your choice (e.g. `ping ntnu.no` or `ping 8.8.8.8`).

> **Milestone:** if this command fails please ask for help before continuing.

```
1
2
3
4
5
```

5. Use `wget` to download a *zip* file from:

- https://www.iik.ntnu.no/ttm4175/wp-content/uploads/2018/08/ttm4175-files.zip

```
1
2
3
4
5
```

6. Create a new user named "guest" with the password "komtek18!", making sure you also create a home folder (`-m` option in `useradd`).

> **Note:** this new user will be used for another team to login into your system.

```
1
2
3
4
5
```

7. In the Linux distribution we are using the home directories give read and execute privileges to "group" and "others". Change the permissions of the home directory of the user "pi" so that your guest cannot see its contents.

```
1
2
3
4
5
```

8. Find a team close to you and exhange your IP addresses. Use the received address, together with the command `ssh` to connect to the other team's computer (e.g. `ssh guest@ipaddress`).

> **Note:** a warning message should appear the first time you connect since the certificates being used have never been exchanged before. After logging in you should notice a different hostname.

```
1
2
3
4
5
6
7
```

9. Send a message to all the users/terminals of the remote machine by using the command `wall`. What happened? Check with the other team.

> **Note:** if nothing happened on the remote side, ask the other team to type the command `mesg y` in their terminal.

```
1
2
3
4
5
6
7
```

10. Instead of sending a message to all the users, use the command `who`, or the command `w` to find out who's in the system and send a message directly to their terminal using the command `write` and `Ctrl+d` to terminate the message.

> **Note:** if you receive the output "write: *user* has messages disabled on ttyX" ask your colleagues to type `mesg y` in their terminal.

```
1
2
3
4
5
6
7
```

11. In a different terminal (or exit the `ssh` session), copy the file "myfile.txt" created earlier to the other team's machine using the command `scp`. Verify that the home folder of the "guest" user has a copy of this file afterwards.

```
1
2
3
4
5
6
7
8
9
10
11
```

12. Return to the terminal with the remote session (or use `ssh` again), and try the following commands replacing the *user* and *tty* with the information obtained when checking `who`'s logged in:

- `echo "Redirecting to a new file (destructive)." > redir.txt`
- `echo "Redirecting, don't care if it already exists!" > redir.txt`
- `echo "Appending if it exists, creating if not." >> append.txt`
- `echo "Appending if it exists, creating if not." >> myfile.txt`
- `write user tty < myfile.txt`

What happened? Were the new files created locally or remotely?

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
```

13. Try the commands:

- `cat myfile.txt | write user tty`
- `sed 's/ /_/g' myfile.txt | write user tty`

What happened? Was "myfile.txt" modified by any of the commands?

```
1
2
3
4
5
6
7
8
9
10
11
```

Before commands are executed, their input and output can be redirected using a special notation interpreted by the shell. ">" and "<" are symbols used for enabling this redirection, opening and closing files for the respective shell execution environment.

"|" is the *pipe* symbol, used in the famous Unix pipeline, in which the output of one command becomes the input of a second. This allows using text as an interface between different programs and the creation of more complex programs based on simpler ones.

1.1.7 Exercise 7 — More GNU/Linux

Additional experiments with common GNU/Linux programs.

1. Extract the *zip* file downloaded earlier with the command `unzip`.

```
1
2
3
4
5
6
7
```

2. Verify the size of the created folder and *zip* file with the commands:

- `ls -ld ttm4175-files*`
- `ls -ld --block-size=K ttm4175-files*`

Explain what you observe.

```
1
2
3
4
5
6
7
```

3. In the previous exercise, how can you use the command `sed 's/.* \([0-9]*K\).*/\1/'` the see the size only?

```
1
2
3
4
5
6
7
```

4. As you may have noticed, the uncompressed folder is smaller than the compressed file. Or is it? Use the command `du`, with the necessary options, to verify the actual disk usage of the files in the folder.

> In Linux "everything" is a file, even a directory, and the command `ls` shows the size of the file corresponding to the directory, not its files.

```
1
2
3
4
5
6
7
8
9
10
11
```

5. Enter the "ttm4175-files" directory and experiment with different commands. For example, you can execute the following commands and register what happens with each:

- `ln -s a/very/strange/directory/structure struct`
- `ls *.mp3`
- `file *.mp3`
- `ls *one*`
- `find . -name "*.txt"`
- `find . -type f -name "*.txt."`
- `find . -type f -name "*.txt" -print`
- `find . -type f -name "*.txt" -print -exec head '{}' \;`
- `find /usr -type f -executable -name find`
- `whereis find`
- `time find /usr -type f -executable -name find`
- `time whereis find`

> **Hint:** Use the documentation to find what the commands do and experiment with commands you already know (`ls` , `pwd` , etc.).

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
```