

# Web Programming Lab

## Lesson 0 - Introduction to Linux & Java

---

Giacomo Bergami

October 10, 2017

University of Bologna

# Introduction

## Personal Information:

- My web address:  
`http://jackbergus.github.io/teaching/LPI17/`
- My e-mail: `giacomo.bergami2@unibo.it`

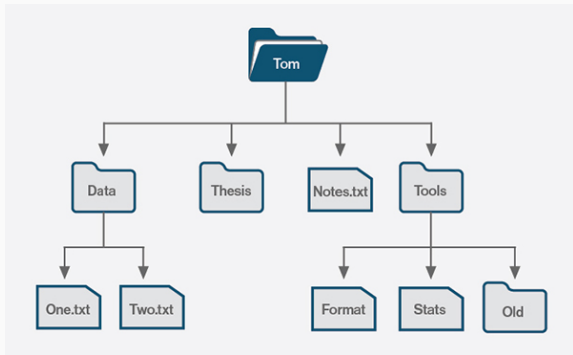
## Exercise 0:

- For these lab sessions, we're going to use the `/public/lpi07/slot0k` folder.
- Create a folder called `surname.name`
- Copy the script `creation.sh` your freshly created directory.
- Run it.

## Exercises 1-6

- Open the terminal.
- The terminal starts to work in your home directory, `~`, that is `/home/students/name.surnamek/`
- `pwd` returns your current directory
- `cd folder` stands for “change directory”, it allows you to move throughout the file system.
- `mkdir folder` allows to create a new folder.
- `touch file` allows to either create a new file, or to update its “timestamp”

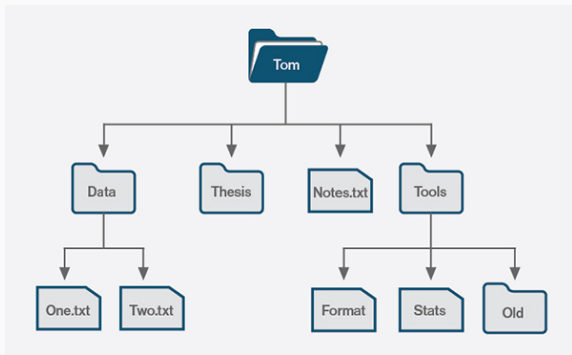
# Exercises 1



**Figure 1:** An example of your home folder (Tom)

**Move to folder Data:**

# Exercises 1

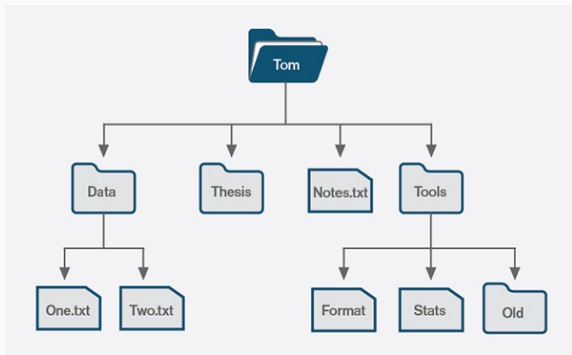


**Figure 1:** An example of your home folder (Tom)

**Move to folder Data:** `cd Data`

You cannot `cd` a file (`cd Notes.txt` is not allowed)

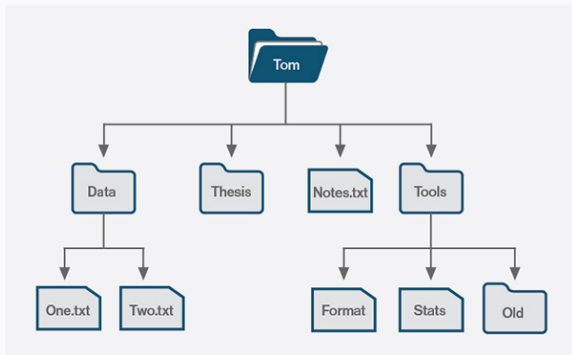
## Exercises 2



**Figure 2:** An example of your home folder (Tom)

**What is the result of `pwd`?**

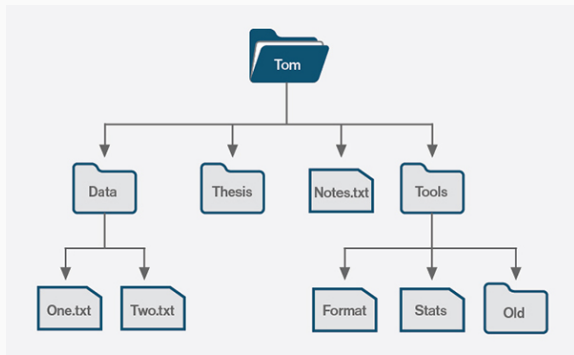
## Exercises 2



**Figure 2:** An example of your home folder (Tom)

**What is the result of `pwd`? `/home/students/Tom/Data`**

## Exercises 3

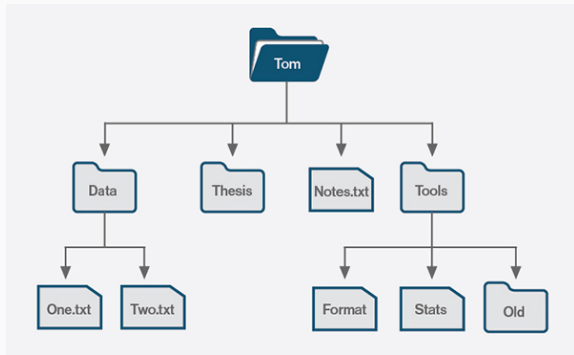


**Figure 3:** An example of your home folder (Tom)

**Create a new file Three.txt:**



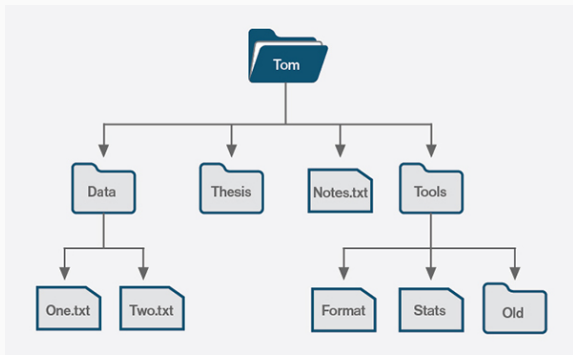
## Exercises 3



**Figure 3:** An example of your home folder (Tom)

**Create a new file `Three.txt`:** `touch Three.txt`

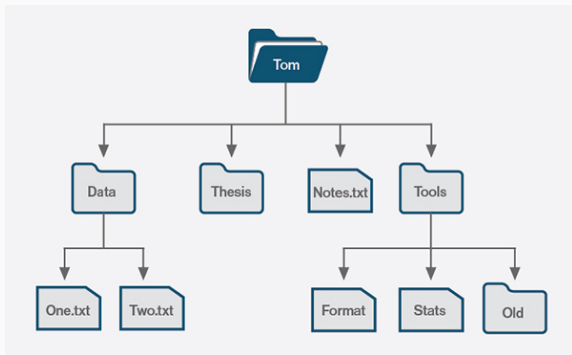
## Exercises 4



**Figure 4:** An example of your home folder (Tom)

**Move back to the parent folder:**

## Exercises 4

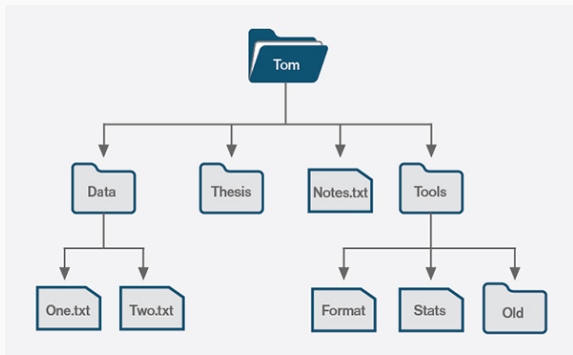


**Figure 4:** An example of your home folder (Tom)

**Move back to the parent folder:** `cd ..`

Two dots `..` always represents the parent folder.

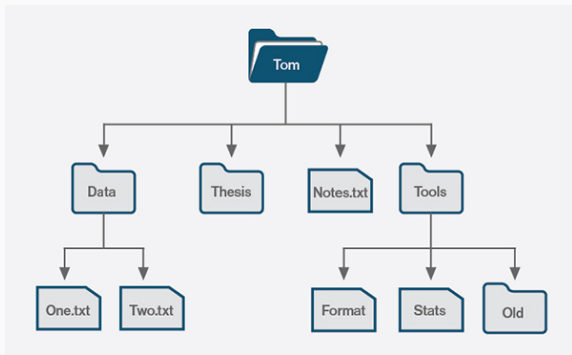
## Exercises 5



**Figure 5:** An example of your home folder (Tom)

**Move to folder Old**

## Exercises 5

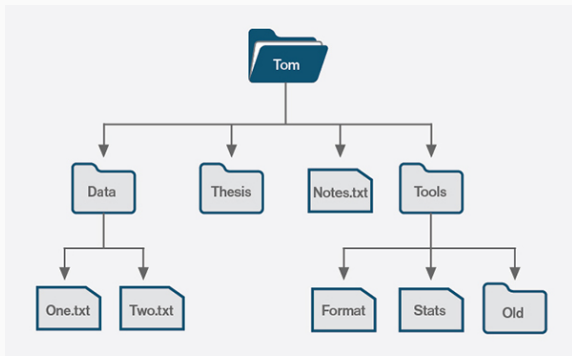


**Figure 5:** An example of your home folder (Tom)

**Move to folder Old** `cd Tools/Old`

One backslash / represents the path separator. A path connects distinct folders with a “father-of” relationship. `..` can be also

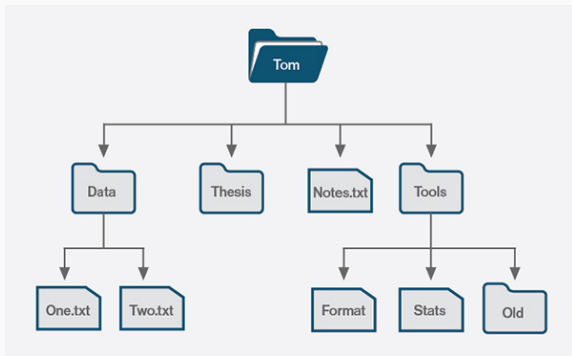
## Exercises 6



**Figure 6:** An example of your home folder (Tom)

**Move to folder Format**

## Exercises 6



**Figure 6:** An example of your home folder (Tom)

**Move to folder Format** `cd ../Format`

When you want to open your editor and start with a new file, you can directly type `youreditor newfile.ext`. Those are some examples:

- `gedit HelloWorld.java`
- `jedit server.c`
- `geany MapReduce.cpp`

**NOTE:** if you want to still use the terminal after launching the command, put `&` after launching the command.



## Exercise 7

- The command to remove used to remove elements within the filesystem is `rm`
- The command `rm file` removes only files.
- In order to remove a folder, you have to remove all the files and folders contained inside it. Hence, you have to type

```
rm -r folder
```

where `-r` stands for “recursively”

- **Exercise:** create a file (and a folder) and then remove it.

# Programming languages

- A programming language ( $\mathcal{L}$ ) is a human readable language, that provides an abstraction from low level machine operations.
- All the programs  $p$  written in a given language  $\mathcal{L}$  cannot be read directly by your computer. Such programs have to be translated into a machine readable form.
- The compiler  $\mathcal{C}$  is a program (function) converting a program  $p$  from a language  $\mathcal{L}$  into a (machine readable) language  $\mathcal{M}$ .  
 $\mathcal{C}: \mathcal{L} \rightarrow \mathcal{M}$ .
- An interpreter  $\mathcal{I}_{\mathcal{M}}^{\mathcal{L}}$  for a language  $\mathcal{L}$  is a program written in the machine language  $\mathcal{M}$  that runs a program  $p$  written for a language  $\mathcal{L}$ .

*Some help:* <https://drive.google.com/open?id=0B5EQQQtU0zzpWlp3V0RjNXBnNEk>

- The C compiler gcc directly compiles the program written in C into a machine readable language  $\mathcal{M}$ . Such compiler is a function  $\text{gcc}: \mathcal{C} \rightarrow \mathcal{M}$ .
- The program  $\text{gcc}(p)$  requires no interpreter.

## Programming language: Java (1)

- The Java compiler `javac` compiles the program written in Java into an intermediate bytecode  $\mathcal{JVM}$ . Such compiler is a function  $\text{javac}: \text{Java} \rightarrow \mathcal{JVM}$ .
- The programs compiled for the  $\mathcal{JVM}$  cannot be directly read by the machine  $\mathcal{M}$ , and hence it requires an interpreter  $\mathcal{I}_{\mathcal{M}}^{\mathcal{JVM}}$ , called **Java Virtual Machine** (java).
- The compiled program  $\text{javac}(p)$  has to be run with such interpreter:

$$\mathcal{I}_{\mathcal{M}}^{\mathcal{JVM}}(\text{javac}(p))$$

## Programming language: Java (1)

- The Java compiler `javac` compiles the program written in Java into an intermediate bytecode  $\mathcal{JVM}$ . Such compiler is a function  $\text{javac}: \text{Java} \rightarrow \mathcal{JVM}$ .
- The programs compiled for the  $\mathcal{JVM}$  cannot be directly read by the machine  $\mathcal{M}$ , and hence it requires an interpreter  $\mathcal{I}_{\mathcal{M}}^{\mathcal{JVM}}$ , called **Java Virtual Machine** (java).
- The compiled program  $\text{javac}(p)$  has to be run with such interpreter:

$$\mathcal{I}_{\mathcal{M}}^{\mathcal{JVM}}(\text{javac}(p))$$

## Programming language: Java (1)

- The Java compiler `javac` has to be invoked on a given source code file:

```
javac HelloWorld.java
```

- The Java interpreter `java` tries to find the generated class file:

```
java HelloWorld
```

- **NOTE:** even though `javac HelloWorld.java` generates a file named `HelloWorld.class`, you shall not invoke `java HelloWorld.class`.

# Coding Exercises

1. Create a program that prints “Hello World”.
  - The class is the minimal functional unit of an Object Oriented language.
  - Each program must have an entry point, called `main`.
  - Use `System.out.println` to print a string.
2. Create a never-terminating program, and kill it.
  - Use `while (<condition>) { <do> }` to create a endless loop cycle.
  - Booleans are `true` and `false`.

All the software used for these lessons is provided at <https://github.com/jackbergus/LPI07/tree/master/Lesson00>