

Introduction to Programming

Lecture 2-3: The Unix Shell

Clément Mazet-Sonilhac
clement.mazet@sciencespo.fr

Sciences Po Paris

Disclaimer

- Most of the material is drawn from the excellent course prepared by software carpentry
- In particular, most exercises are drawn from it (If you really want to learn something, don't look up the answers)
- Other source of inspiration is the very complete OpenClassrooms website

What and why ?

Unix, Shell & Bash ?

- The **Unix Shell** (\approx the terminal \approx the command line) : one of the first (and simplest ?) way to interact with your computer

What and why ?

Unix, Shell & Bash ?

- The **Unix Shell** (\approx the terminal \approx the command line) : one of the first (and simplest ?) way to interact with your computer
- Why should you use it ? Despite the (high) entry cost, it is :
 - ▶ fast and efficient
 - ▶ reusable
 - ▶ sometimes necessary
 - ▶ but, more importantly...

What and why ?

Unix, Shell & Bash ?

```
File Edit View Terminal Help
taufanlubis@toshiba:~$ sudo apt-get install terminator
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  python-keybinder
The following NEW packages will be installed:
  python-keybinder terminator
0 upgraded, 2 newly installed, 0 to remove and 4 not upgraded.
Need to get 202kB of archives.
After this operation, 1,815kB of additional disk space will be used.
Do you want to continue [Y/n]? y
Get:1 http://archive.ubuntu.com/ubuntu/ lucid/universe python-keybinder 0.0.4-1
[12.2kB]
Get:2 http://archive.ubuntu.com/ubuntu/ lucid/universe terminator 0.93-0ubuntu1
[190kB]
Fetched 202kB in 5s (37.2kB/s)
Selecting previously deselected package python-keybinder.
(Reading database ... 129972 files and directories currently installed.)
Unpacking python-keybinder (from .../python-keybinder_0.0.4-1_i386.deb) ...
Selecting previously deselected package terminator.
Unpacking terminator (from .../terminator_0.93-0ubuntu1_all.deb) ...
Processing triggers for desktop-file-utils ...
Processing triggers for python-gmenu ...
Rebuilding /usr/share/applications/desktop.en_US.utf8.cache...
Processing triggers for man-db ...
Processing triggers for hicolor-icon-theme ...
Processing triggers for python-support ...
Setting up python-keybinder (0.0.4-1) ...

Setting up terminator (0.93-0ubuntu1) ...
update-alternatives: using /usr/bin/terminator to provide /usr/bin/x-terminal-emulator (x-terminal-emulator) in auto mode.

Processing triggers for python-support ...
taufanlubis@toshiba:~$
```

- ... It makes you look smart

What and why ?

Unix, Shell & Bash ?

- UNIX is one of the first Operating System (1970)
 - The Shell is a command (instructions) **interpreter** to interact with UNIX
- ⇒ Specificity : run other programs rather than doing the calculations itself

What and why ?

Unix, Shell & Bash ?

- UNIX is one of the first Operating System (1970)
- The Shell is a command (instructions) **interpreter** to interact with UNIX
- ⇒ Specificity : run other programs rather than doing the calculations itself
- **Bash** (Bourne Again SHell) is the most common UNIX Shell (Must watch : [Revolution OS](#))
- Use the Unix shell from a command-line interface (CLI). The heart of the CLI : the REPL (read-evaluate-print loop)
- Why REPL : when the user types a command and then presses the Enter (or Return) key, the computer reads it, executes it, and prints its output.

What and why ?

Linux, Mac and Windows

- Linux and Mac OS include by default a UNIX-Shell which is Bash
- Windows OS doesn't (bad). But you can use an emulator (PuTTy, Cygwin, Mintty, Git, etc.)

OR Windows users also can install Ubuntu (one of the best Linux distribution) on their computer

Fundamentals (I)

The File System

- The file system : part of the OS responsible for managing files and directories

Fundamentals (I)

The File System

- The file system : part of the OS responsible for managing files and directories
- How to navigate in this file system ?
 - ▶ cd (Change Directory) : most essential command to navigate in the file system
 - ▶ ls (List) : display all the files and directories inside the current working directory
 - ▶ pwd (Print Working Directory) : print the path of current working directory

Fundamentals (I)

The File System

- The `pwd` command display the path of the current working directory
- Ex : `/Users/Clément/Dropbox/Teaching/IP/2019/2-unix`

Fundamentals (I)

The File System

- The `pwd` command display the path of the current working directory
- Ex : `/Users/Clément/Dropbox/Teaching/IP/2019/2-unix`
 - ▶ At the top is the root directory (in red) that holds everything else. We refer to it using a slash character / on its own

Fundamentals (I)

The File System

- The `pwd` command display the path of the current working directory
- Ex : `/Users/Clément/Dropbox/Teaching/IP/2019/2-unix`
 - ▶ At the top is the root directory (in red) that holds everything else. We refer to it using a slash character / on its own
 - ▶ Inside that directory are several other directories, including `Users` (where users' personal directories are located), in green.

Fundamentals (I)

The File System

- The `pwd` command display the path of the current working directory
- Ex : `/Users/Clément/Dropbox/Teaching/IP/2019/2-unix`
 - ▶ At the top is the root directory (in red) that holds everything else. We refer to it using a slash character / on its own
 - ▶ Inside that directory are several other directories, including `Users` (where users' personal directories are located), in green.
 - ▶ Our current working directory is stored inside `/Users` - because `/Users` is the first part of its name - and `/Users` is stored inside the root directory, its name begins with /

Fundamentals (I)

The File System

- The `cd` command allows you to navigate through the repositories
- Ex : `cd + Desktop`, `cd + .` (also, try `cd + ~` or simply `cd`), `cd + ..`

Fundamentals (I)

The File System

- The `cd` command allows you to navigate through the repositories
- Ex : `cd + Desktop`, `cd + .` (also, try `cd + ~` or simply `cd`), `cd + ..`
- **Exercise 1** : Using `cd`, `ls` and `pwd` find your current working directory, navigate to the desktop and print your directory.

Fundamentals (II)

Create a directory / a file

- **How to create a directory ?**

⇒ Use the command `mkdir + NameOfTheFolder`

- Try to not use spaces in your files and directories names, EVER

Fundamentals (II)

Create a directory / a file

- **How to create a directory ?**

⇒ Use the command `mkdir + NameOfTheFolder`

- Try to not use spaces in your files and directories names, EVER

- **Exercise 2 :** Go to your desktop and create a folder called IntroProg2019

Fundamentals (II)

Create a directory / a file

- **How to create a directory ?**

- ⇒ Use the command `mkdir + NameOfTheFolder`
- Try to not use spaces in your files and directories names, EVER
- **Exercise 2 :** Go to your desktop and create a folder called IntroProg2019
- **How to create a file ?** Most generic command is `touch`
- ⇒ Ex : `touch a-blank-file.txt` or `nano a-blank-file.txt`. (This will create a blank file. Work with any extension (.txt, .csv, .jl etc.))

Fundamentals (II)

Create a directory / a file

- **How to create a directory ?**

- ⇒ Use the command `mkdir + NameOfTheFolder`
- Try to not use spaces in your files and directories names, EVER
- **Exercise 2 :** Go to your desktop and create a folder called IntroProg2019
- **How to create a file ?** Most generic command is `touch`
 - ⇒ Ex : `touch a-blank-file.txt` or `nano a-blank-file.txt`. (This will create a blank file. Work with any extension (.txt, .csv, .jl etc.))
- **Exercise 3 :** Create a file called `test-file.txt` into the folder `IntroProg2019`

Fundamentals (II)

Remove a directory / a file

- You can **remove** files with the command `rm` and directory with `rm -r`
- **Be extremely careful** : the Unix shell doesn't have a trash bin. Deleting is permanent
- `rm -r` can be a very dangerous command, since you might remove loads of files. To be certain that you are not removing too many files, you can type `rm -r -i thesis`, and the Shell will ask you whether you want to remove this or that file

Fundamentals (III)

Move or copy a file

- **How to rename or move a file?** Use `mv + name-file-to-be-rename + new-name`

Fundamentals (III)

Move or copy a file

- **How to rename or move a file?** Use `mv + name-file-to-be-rename + new-name`
- Careful `mv` will silently overwrite any existing file with the same name. Add the flag `-i` for the Shell to warn you if `mv` is going to overwrite any file

Fundamentals (III)

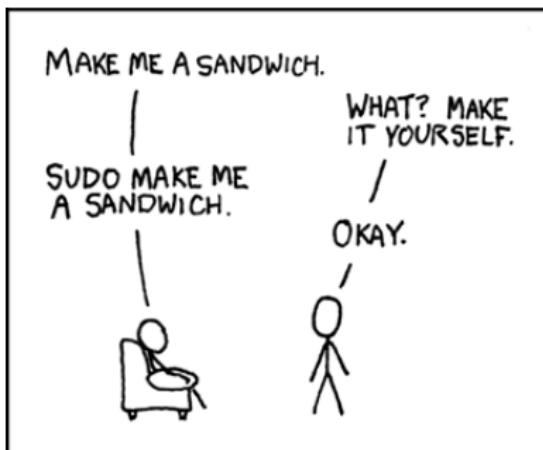
Move or copy a file

- **How to rename or move a file?** Use `mv + name-file-to-be-rename + new-name`
- Careful `mv` will silently overwrite any existing file with the same name. Add the flag `-i` for the Shell to warn you if `mv` is going to overwrite any file
- **Exercise 4 :** in the directory `IntroProg2019`, create `draft.txt`. Then, rename it to `quote.txt`. Finally, move `quote.txt` from the `IntroProg2019` directory to its parent directory, the Desktop.
- `cp` works like `mv`, but makes a copy of the file instead of (re)moving it

Fundamentals (IV)

Substitute user do

- Useful : sudo + command ("superuser do", or "switch user do") allows a user with proper permissions to execute a command as another user, such as the superuser.



Fundamentals (V)

Choose a good Text Editor!

- Sublime Text 3.0
- Notepad++
- etc.

Final Exercise

Customise your prompt !

- Navigate to your home with `cd ~`
- Type `ls -la` and look for `.bash_profile` file
- If it does not exists, create one. If it exists, just open it with command `open` (or `explorer` for those using Windows)
- Write in the file `export PS1 = "\u \$"`
- try to add `\d`, `\h`, `\W` etc.

Why is this useful ?

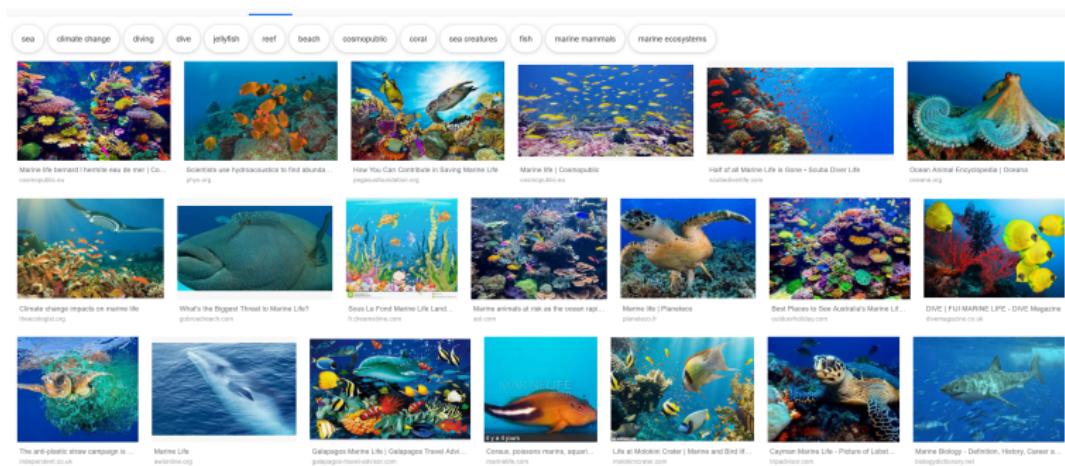
Context I

- Now we need an example : go to my website (cms27.github.io)
- In the Teaching section, click on Data Shell to download a zip file called data-shell.
- Save it on your desktop.

Why is this useful ?

Context II

- Nelle is a marine biologist who collected 1520 samples on marine life stuff (text, pictures, maps, etc.)



Why is this useful ?

Filters : a simple example

- **Exercise 1** : Count the number of words in a file ?
 1. go to the folder molecules
 2. display the files
 3. use the command `wc` to count the number of words in each file

Why is this useful ?

Filters : a simple example

- **Exercise 1** : Count the number of words in a file?
 1. go to the folder molecules
 2. display the files
 3. use the command `wc` to count the number of words in each file
- Answer : `wc *.pdb`

Why is this useful ?

Filters : a simple example (cont'd)

- **Wildcard ?** Wildcards are a form of shortcut.

Why is this useful ?

Filters : a simple example (cont'd)

- **Wildcard ?** Wildcards are a form of shortcut.
- * : matches zero or more characters ; e.g. *.pdb will be a list containing ethane.pdb, propane.pdb etc. Similarly, p*.pdb will only contain pentane.pdb and propane.pdb
- ? : matches one character
- Wildcards can be combined : e.g. p*.p*
- [xy] matches either x or y

Why is this useful ?

Filters : a simple example (cont'd)

- **Wildcard ?** Wildcards are a form of shortcut.
- * : matches zero or more characters ; e.g. *.pdb will be a list containing ethane.pdb, propane.pdb etc. Similarly, p*.pdb will only contain pentane.pdb and propane.pdb
- ? : matches one character
- Wildcards can be combined : e.g. p*.p*
- [xy] matches either x or y
- **Exercise 2** : Ex : in north-pacific-gyre/2012-07-03/, list all the text files ending with A or B (i.e. exclude those ending with Z)

Why is this useful ?

Filters : a simple example (cont'd)

- **Wildcard ?** Wildcards are a form of shortcut.
- * : matches zero or more characters ; e.g. *.pdb will be a list containing ethane.pdb, propane.pdb etc. Similarly, p*.pdb will only contain pentane.pdb and propane.pdb
- ? : matches one character
- Wildcards can be combined : e.g. p*.p*
- [xy] matches either x or y
- **Exercise 2** : Ex : in north-pacific-gyre/2012-07-03/, list all the text files ending with A or B (i.e. exclude those ending with Z)
- **Exercise 2 bis** : Go back to folder molecule and let's find which of these files is the shortest !

Why is this useful ?

Loops

- Why should you use loops ?
 1. Allow us to execute commands repetitively
 2. Reduces the amount of typing
 3. Less typing = less likely to do a typing mistakes

Why is this useful ?

Loops

- **Exercise 3** : Nelle would like to copy several files
- Go to the directory `data-shell/creatures`
- Try `cp *.dat original-*.dat` (This command should not work !)

Why is this useful ?

Loops

- **Exercise 3** : Nelle would like to copy several files
 - Go to the directory `data-shell/creatures`
 - Try `cp *.dat original-*.dat` (This command should not work !)
- ⇒ When `cp` receives more than two inputs, it expects the last input to be a directory where it can copy all the files it was passed. Since there is no directory named `original-*.dat` in the `creatures` directory we get an error.

Why is this useful ?

Loops

- **Exercise 3** : Nelle would like to copy several files
 - Go to the directory `data-shell/creatures`
 - Try `cp *.dat original-*.dat` (This command should not work !)
- ⇒ When `cp` receives more than two inputs, it expects the last input to be a directory where it can copy all the files it was passed. Since there is no directory named `original-*.dat` in the `creatures` directory we get an error.
- Solution : **use a loop !**

Why is this useful ?

Loops

- Example of a loop :

```
for filename in basilisk.dat unicorn.dat
do
    cp $filename original-$filename
done
```

- Astuce : always write the beginning (do) and the ending of the loop (done) at the beginning

Why is this useful ?

Loops

- **Exercise 4** : for each files in the creatures directory, print the name of the file (may want to check the echo command), and the last 20 lines of the file (have a look at the tail command)

Why is this useful ?

Loops

- **Exercise 4** : for each files in the creatures directory, print the name of the file (may want to check the echo command), and the last 20 lines of the file (have a look at the tail command)
- **Exercise 5** : write another loop to show that your previous code indeed printed only 20 lines per file

Nelle's original problem

- **Remember :** Nelle wants to run a statistical program, goostats, on the sample contained in north-pacific-gyre/2017-07-03
- **Info :**
 - ▶ run only the program on the sample finishing by A or B
 - ▶ hint 1 : goostats takes two inputs : i) the file on which to run the program and ii) the file that's going to store the results to run an external program
 - ▶ hint 2 : bash goostats filename resultfile

Nelle's original problem

- **Remember :** Nelle wants to run a statistical program, goostats, on the sample contained in north-pacific-gyre/2017-07-03
- **Info :**
 - ▶ run only the program on the sample finishing by A or B
 - ▶ hint 1 : goostats takes two inputs : i) the file on which to run the program and ii) the file that's going to store the results to run an external program
 - ▶ hint 2 : bash goostats filename resultfile
- **Exercise 6 :** Write a program that's going to run goostats on each file of the sample, and store each results in a different file, with name stats-ORIGINAL_NAME_OF_THE_FILE. Debug as much as possible your code.

Finding things

grep

- `grep` = "global/regular expression/print", a finder in Shell
 - Example : find the word "The" in a Haiku (in the folder writings)
 - Try `grep The haiku.txt`
 - retry with `grep -w The haiku.txt`. What the difference ?
- !! if expressions use "my expression"

Finding things

Flags

- Several **flags** that are useful
 - ▶ -w : the expression is searched for as a word
 - ▶ -n : each output line is numbered
 - ▶ -i : perform case insensitive matching
 - ▶ -v : reverse search : selected lines are those not matching any of the specified patterns

Finding things

Flags

- Several **flags** that are useful
 - ▶ -w : the expression is searched for as a word
 - ▶ -n : each output line is numbered
 - ▶ -i : perform case insensitive matching
 - ▶ -v : reverse search : selected lines are those not matching any of the specified patterns
- Example : Try `grep -n -w -i "the" haiku.txt`
- More complex : `grep -E '^.{o}' haiku.txt`. What does this command do ?

Finding things

find

- grep finds patterns in files ; find finds pattern in folders (try `find .`)
- Again, some flags are really useful :
 - ▶ `-type d` : lists only the directories
 - ▶ `-type f` : lists only the files
 - ▶ `-name some_name` : list only the files matching `some_name`
- + You can combine : Example : Try `c -l $(find . -name '*.*.txt')`

Others useful commands

- Use the keyboard's arrows to retrieve past commands
- Stop a running program at any point in time using `Ctrl + c`
- `Ctrl + a` and `Ctrl + e` brings the pointer to the beginning and the end of the line respectively `history` lists the last few hundred commands, and `!123` runs the command associated with line 123.

Others funny (but very useless ?) commands

- Install and run sl
- Try telnet towel.blinkenlights.nl
- Compute factors using the command factor + X
- etc.

Shell scripts

- The UNIX-shell was supposed to be **reusable and efficient** but we have to retype command again and again !?
- ⇒ **Write Shell scripts !**
- A shell script : a small programs that contains a list of Shell commands that end with .sh

Shell scripts

How to ?

1. Open your text editor
2. Write your bash command as if it was in the terminal
3. Save the file, ex : `ScriptName.sh`
4. Run the file in the terminal typing : `bash ScriptName.sh`

Shell scripts

How to ?

- **Exercise 7** : Write a bash script that select the 15 first lines of the file octane.pdb and only display the last 5 (hint : use head and tail)

Shell scripts

How to ?

- **Exercise 7** : Write a bash script that select the 15 first lines of the file octane.pdb and only display the last 5 (hint : use head and tail)
- **Exercise 8** : Write a bash script that select the 15 first lines of any file *.pdb that you choose and only display the last 5 (hint : use head and tail) + don't forget to document your code !