UNIVERSITÉ DU LUXEMBOURG

Hacking in Bash

Programming Fundamentals 2

Goals

- ✯ Understanding the basic principles of the shell, the editor Sublime Text, git and the Markdown format.
- ✯ Learn all the necessary tools needed to submit a laboratory in PF2 (or, why make it easy when we can make it complicated :-)).
- ✯ **Relevant videos**:
  - Control your computer like a hacker.
  - How to submit a laboratory.

Deliverables

1. The code on your Github repository generated by clicking here: `https://classroom.github.com/a/E6_SWbmw` (wait until exercise 5).
2. **Reviewer:** Léo Gamboa dos Santos (LeoLuxo on Github).
3. **No automated review**

Exercise 1 – Pallet Town

(...where all great adventures start...)

We are going to use a few UNIX commands, it is wise to record those in an "engineering notebook", especially the problems you had and how you solved them. I also advise you to watch the video "Control your computer like a hacker" first, unless you already know Bash.

1. Open a terminal using the shortcut `ctrl+alt+t` (Linux) or through the graphical interface.
2. The terminal allows us to navigate in the directories and files of your system. Let's write our first commands:
   - Create a directory `BICS` with `mkdir BICS`[1].
   - Go inside this directory with `cd BICS`[2].
   - Create another directory `PF2`.

[1] mkdir = make directory.
[2] cd = change directory.

- Go to the parent directory with `cd ..` and type `ls`[3]. What does this command do?
- Now type `cd ~`. You can check where you are using the command `pwd`[4], where are you? What does ~ mean?
- Go to the directory `PF2`.

3. Test the following commands, and use `pwd` each time to verify where you are:

    1. `cd`
    2. `cd .`
    3. `cd ..`
    4. `cd ../..`
    5. `cd /`
    6. `cd /home/`
    7. `cd ~`
    8. `cd ~/BICS/PF2`
    9. `cd ~/BICS/../BICS/PF2`

    Which paths are *relative* and which ones are *absolute*? What does the path `/` represent? What about the single dot `.`?

4. Test the following commands:

    1. `ls *`
    2. `ls .`
    3. `ls ..`
    4. `ls ../..`
    5. `ls ~`
    6. `ls /usr`
    7. `ls /usr/*`
    8. `ls /u*r/*`
    9. `ls /u*/*`

    What does the star `*` means?

5. Go to the directory `~/BICS/PF2`, and verify if you succeeded using `pwd`. Type the command `subl -n .` which open a new window (option `-n`) of the Sublime Text editor. Create and save a file in which you list things, such as your favourite musics.

6. Now, we do not bother describing all commands in details each time. Nonetheless, here are two ways to help you in your quest to become fluent in Bash:

    - The hacker way (aka. RTFM way) is to use the command `man` (stands for manual) which allows us to get the documentation of a command. For instance, if you type `man cd`, you will have information on `cd`. However, you will *really* have *all* the information which might feel a bit overwhelming at this stage.
    - Otherwise, Google is your friend, and it can always help you if you ask a good question (not always easy). A technique is to look for usage examples, *e.g.*, type "examples cd command linux".

7. Now let's check your skills with a small challenge! Using the command `wc`, count the number of *words* of the file you created above. Next, find all the lines containing the letter "e" using the command `grep`.

First level completed!

[3]ls = list [files].
[4]pwd = print working directory.

Exercise 2 – You can do everything from the terminal!

1. You can open any application from the terminal. You can type `nautilus .` in the terminal to open the current directory in your files browser[5]. Sometimes the commands of the applications do not exactly coincide with their real names, *e.g.*, `subl` stands for Sublime Text. Hence, you actually need to remember those names (or record them in your notebook, remember you have a notebook?!). You should notice that if you do not close Nautilus, the terminal becomes "unresponsive", and you cannot use it anymore. Indeed, the terminal will block until you close Nautilus, at which point you will be able to type more commands. Try it out!
2. There is also another way to "kill" programs directly from the terminal, which can be useful if you run a buggy program (possibly made by you). Try to open Nautilus again and press `ctrl+c` in the terminal to close Nautilus immediately.
3. You can also run a program from the terminal without blocking the terminal. Try to run `nautilus &` with the ampersand character `&` at the end of the command. It will run the program in the background of the terminal; you can put it to the foreground (and close it using `ctrl+c`) using the command `fg`—meaning foreground, you guessed it!

Exercise 3 – The cycle of life: create, move and delete

1. Check out the commands (and their options) `ls`, `cp`, `mv` et `rm`[6]. Create a directory `test` in the directory `PF2` and create several files there. You can use the command `touch file1.txt` to create an empty file. Move all files to the parent's directory and then delete them all.
2. Create the file `~/.bashrc` if it does not already exist. You can write commands in this file which will be executed each time you open the terminal. For instance, if you add `cd BICS` inside, each time you open a terminal, you will end up in the `BICS` directory.

   To avoid deleting a file by mistake, we can use the commands `rm`, `mv` and `cp` with the option `-i` ("interactive"). Let's define several aliases for these three commands including the option `-i` by adding those to the file `.bashrc` using the command `alias`.

Exercise 4 – Unlimited power!

Think about a use case where Bash could be useful for you, in your everyday life. Then, show which command can fulfill your needs. Search Google for useful commands, do not repeat a command you already have used here, be creative :)

Exercise 5 – Configure Sublime Text

1. Install Sublime Text `https://www.sublimetext.com/download`.
2. Set up Sublime Text.
   1. Open `Preferences > Settings`.
   2. The left panel is the default setting, and the right panel is your customized preferences which override the default.
   3. Add the following in the right panel:

[5]On macOS type `open .` to open the current directory in the *Finder*, however you will still be able to use the terminal after you opened, making the rest of this exercise irrelevant, you can try to open another app behaving similarly to Nautilus on Linux.

[6]cp = copy, mv = move, rm = remove.

```
{
  "tab_size": 2,
  "translate_tabs_to_spaces": true,
  "trim_trailing_white_space_on_save": true
}
```

This allows us to indent with two spaces by default.

3. Before starting the tutorial, we must install an extension of Sublime Text called *package control*. For that, follow the steps given here: `https://packagecontrol.io/installation`

Exercise 6 – Sublime Tutor

Sublime Tutor is a package for Sublime Text to teach you basics and advanced functionalities of Sublime Text. Press `ctrl+shift+p` on Linux (Mac: `cmd+shift+p`), and type `install package`, in the new command palette, type `Sublime Tutor` and click on it. Follow the steps of the tutorial.

Exercise 7 – Configuring Git

1. We are now going to clone the Github repository in order to upload the exercises online (and much more!). A first step is to configure SSH (it is a secure way to connect your computer to Github). With the little we already learnt about bash, you should be able to follow this tutorial: `https://docs.github.com/en/authentication/connecting-to-github-with-ssh`. There is a lot of information about SSH, and not everything might be clear but that's normal, you need to build up your computer-related vocabulary.
2. Now that we have SSH ready, we can clone the Github repository of this lab:
   - Generate your Github repository for this lab by clicking here: `https://classroom.github.com/a/E6_SWbmw`
   - On the Github webpage, accept the assignment, wait a few seconds and refresh the page, click on the link that just appeared.
   - You are now on the webpage of the newly created git repository, and we want to "download" this repository onto our computer (in Git term, this operation is called *cloning*).
   - Use the SSH authentication mode to clone the git repository using `git clone <PUT YOUR LINK HERE> hacking_in_bash` where your link is obtained by clicking on "Code" in the Github interface, ssh tab, and should have the form `git@github.com:uni-lu/ai-lol2d-ptal.git` for instance.
   - You can enter the cloned repository using `cd hacking_in_bash` and check what is inside using the commands we just learnt.

Exercise 8 – Learning Git

Git is a powerful tool but takes a little bit of time to master. Create a directory `BICS/PF2/git_tutorial`, and follow the tutorial `https://githowto.com/setup` or, alternatively, play the "game" *Oh My Git!* (`https://ohmygit.org/`). You need to learn, *at least*, how to commit, push, pull.

Exercise 9 – How to submit a laboratory

Now that you (almost) master Git, it is time to put your skills to the test. This is the procedure to submit all laboratories of PF2! If you mess up, we will not be able to review your code :-(So be serious about it!

1. Go to the git repository of this laboratory.

2. Using Sublime Text, open the file `README.md` of the current repository[7] and record your answers to questions 1.2, 1.3, 1.4, 1.7, 3 and 4, using at least one commit per answer, therefore at least 6 commits). **Explicitly write down all the commands (even the `cd`, `ls`, etc.) you used to answer the questions.**
3. Push the file, use `git status` to make sure you "committed" everything.
4. In the Github interface, on the page of your repository, go to the "Pull Requests" tab. There, on the right side, you should be able to click a (rather small) button "request review" next to the reviewer name.

Reviews take time, so make sure you have carefully checked your laboratories before submitting to avoid multiple roundtrips between reviewers and you.

[7]This file is a Markdown file, you can check its format at `https://docs.github.com/en/github/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax`