How to properly use Linux, i3, Vim, Git, Makefile

1 Introduction

Welcome to the seminar of S4, this week you are going to learn some useful notion for debugging your programs more easily, know everything on Linux and i3, gain efficiency and speed on vim, create Makefile and use git like to perfection. If you have any suggestion/note on the subject or the class, please send a email at any assistant.

At first let's explore the world of Linux and more especially the world of open source.

2 What's Linux ?

According to *Wikipedia* about Linux :

> Linux is a family of open source Unix-like operating systems based on the Linux kernel. Linux is typically packaged in a Linux distribution. Distributions include the Linux kernel and supporting system software and libraries, many of which are provided by the GNU Project. Many Linux distributions use the word "Linux" in their name, but the Free Software Foundation uses the name GNU/Linux to emphasize the importance of GNU software

So Linux is a family of open source which regroup a lot of distribution. Distribution is an operating system made from a software collection, which is based upon the Linux kernel and, often, a package management system. For the famous one you have :

- *Ubuntu*
- *Arch Linux*
- *Debian*
- *CenterOs*

For example, the computer at EPITA are based on Arch Linux. This distro is famous for lightweight and flexible that tries to Keep It Simple. There are many more distro that follow differents philosophies, Try to find other distro, some are very fun to explore.

2.1 GNU Project

Previously we talk about the *GNU Project*. Its goal is to give computer users freedom and control in their use of their computers and computing devices by collaboratively developing and publishing software that gives everyone the rights to freely run the software, copy and distribute it, study it, and modify it. GNU software grants these rights in its license.[1]

[1]For more info check on the GNU website `https://www.gnu.org/home.en.html`

2.2 Linux Kernel

One more thing, we must talk about the *Linux Kernel*. Linux kernel is a free and open-source, Unix-like operating system kernel. The Linux family of operating systems is based on this kernel and is deployed on personal computers, servers distributions, on various embedded devices such as routers, wireless access points, etc...You can check the github of the Kernel Linux to see the size of the project. [2]

3 Git

3.1 Introduction

In EPITA, you use *Git* to make your project and keep version of it. But this tool is very powerful and hard to control. It is a distributed version-control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. In this part, we will show you how to use it correctly.

3.2 Tips/Advice

First if you never use git or you barely know some command, we advice you to train on this website : `https://learngitbranching.js.org/`. It will help you on the basic and more of git. Now there are some tips to help you in way to use better git.

3.3 How to make good commit ?

Some of you make commit like

- *Fix bugs*
- *It works but I don't know why*

This kind of commit MUST be avoid. A commit represent a checkpoint about you project, so you need to details what you do or what you fixed. To help you, there are some good style of making commit, like :

- You can start by writing the feature you fixed or worked on, and next to it, try to describe what you do. For example, if you fix a segfault about you fibonacci exercise :

```
commit e1d4884d3b2c9aae5a386dd430ce21cc571d4b97 (head -> master)
author: login_x <login_x@epita.fr>
date:   mon jan 6 03:21:50 2019 +0100

fibonacci: fix segfault when type bad value
```

[2] https://github.com/torvalds/linux

- Commit is like an email you write but it includes your modification, you can start by add some information into [...], If we retake our example:

```
commit e1d4884d3b2c9aae5a386dd430ce21cc571d4b97 (head -> master)
author: login_x <login_x@epita.fr>
date:   mon jan 6 03:21:50 2019 +0100

[fibonacci][FIX]: fix segfault when type bad value
```

  You notice that we type two information : the feature, what we did and a description. This looks very annoying to write but trust us, it will useful when you start to work on big project.

3.4 Git Flow

Branch are a very useful feature which can help you when implementing multiple feature without destroy the part of other member of your team. The above website show you show to create and use branch, but like commit, there are a way to use branch, it's called *git flow* `https://danielkummer.github.io/git-flow-cheatsheet/index.html`.
Check this website to get more information about it. We strongly advice you to use this method, it will help you on how to organize your branches and the working tree of your git project

3.5 Use an editor to create commit

One thing that is annoying, is to type this command:

```
42sh$ git commit -m "My big commit message"
```

Write commit on the command line is not a good way, because it's not very adapted to write message. But we can tell git to use an editor instead of the command line. You just need to run this command:

```
42sh$ git config --global core.editor your_favorite_editor
```

Or manually add the following to your /.gitconfig:

```
[core]
editor = your_favorite_editor
```

Now to commit your message, you just need to run:

```
42sh$ git commit
```

4 i3

4.1 Introduction

According to the i3 website[3], i3 is a tiling window manager, completely written from scratch. i3 is primarily targeted at advanced users and developers. Based upon the experiences of i3 developer. The main goals are :

- Implement multi-monitor correctly, that is by assigning each workspace to a virtual screen. Especially make sure that attaching and detaching new monitors like video projectors works during operation and does the right thing.
- Only add features which benefit many people, instead of going to great lengths to support rarely used workflows.
- One rule : Don't be bloated, don't be fancy.

First if you are not on i3, you need to switch. Because for the next year, you will have no choice and it's better to start using it now than during exams.

Then you should read the user's guide of i3, available at this address `https://i3wm.org/docs/userguide.html`

4.2 Exercises

Now that you know a little more about i3: let start some exercises :

4.2.1 Look my screen

You need to add a shortcut to look you screen when press : *MOD + Shift + x*. You can use the *i3lock(1)* program

4.2.2 I care about my eyes

Readshift is a program that set color temperature of display according to time of day. You need to start *redshift(1)* at the start of your i3 session, with this parameters :

```
42sh$ redshift -l 48.8534:2.3488 # Force redshift to use time of Paris
```

To check if you program is started, run this command and check if you see *redshift* :

```
42sh$ ps -aux | less
```

[3]https://i3wm.org/

4.2.3 Add more thing...

Try to customize you i3 like you will, change the font size, add more shortcut, change color, ect...

5 Vim

5.1 Introduction

Vim is a powerful text editor, it's designed for use both from a command-line interface and as a standalone application in a graphical user interface. It has great feature but it's very hard to handle. We strongly advice you try using it, even if it's very weird at first, but let's try to configure vim.

5.2 Configure vim

This is basic configuration for vim :

/afs/.config/vimrc

```
set number      " show number line
set smartindent " activate smart indentation among the language you use
set autoindent  " keep indentation on a new line

set encoding=utf-8
set ruler " show the positon of your cursor
set shiftwidth=4 " set indentation at 4 spaces
set expandtab " set spaces instead of tabs

syntax on " enable syntax coloration

set colorcolumn=80 " draw a line at 80th column
```

You can add more configuration by searching on the net, or via the help page of vim by running :

```
:help
```

5.3 Some advanced features

In vim, you have advanced features which are very useful to know :

- You can split your vim screen by running the command :

```
:vs " For vertical split
```

or you can type :

```
:sp " For horizontal split
```

To move between the buffer, you need to press the keybinding : *Ctrl + w* then *h or l* or the left arrow or right arrow.

- Vim has also completion for anything : word, file, ect... In insert mod, press the keybinding: *ctrl + n* or if you need to complete about file press *Ctrl + x* then *Ctrl + f*. They are many more completion, check the help page by running :

```
:help 'complete'
```

- Vim can run *make(1)* by typing the command :

```
:make
```

Of course, vim must be at the same path as your Makefile. Then vim parse error message of your copiler and set the cursor at the first error. To move to the next type :

```
:cn
```

6 Exercises

6.1 Exercice 0: Bash things ...

Tips: *mkdir(1), cd(1), tree(1), ls(1)*

To start and do things properly, open a terminal and create in your afs a 'SEMINAR' folder where you will do all exercises of the seminar. You can also create a folder for each day's exercises.
... Done ? Now you can start !

6.2 Exercice 0.1: Vimtutor

Nothing more to say, Do vimtutor. Just run this command to begin the tutorial:

```
42sh$ vimtutor
```

6.3 Exercice 1: Make your first Makefile

asm_print.c

```c
#include "stdio.h"

void print_asm(void)
{
    printf("ASM 2022 <3");
}
```

asm_print.h

```c
#ifndef ASM_PRINT_H
#define ASM_PRINT_H

void print_asm(void);

#endif /* ASM_PRINT_H */
```

main.c

```c
#include "asm_print.h"

int main(void)
{
    print_asm();
    return 0;
}
```

Create the above files and create a simple makefile that generates the **asm2022** executable.

6.4 Exercice 2: Warm-up

For each exercise:

- Solve the problem (i.e. find an algorithm)
- Implement your solution and the **Makefile**
- Test it, debug it
- See if you can do better (optimization)

Gdb tags : -Wall -Wextra -Werror -std=c99 [-fsanitize=address]

6.4.1 Fibonacci sequence (as always :))

Write a function that computes the Fibonacci sequence.

```
unsigned long int fibonacci(int n);
```

```
asm2022$ ./fibo 3
2
asm2022$ ./fibo 11
89
asm2022$ ./fibo 90
2880067194370816120
asm2022$ ./fibo_rec
"Only one argument is needed"
```

6.4.2 Power

Write a function that prints a^b. Your code has to be optimized.

```
int my_pow(int a, int b);
```

6.4.3 Count digits

Write the function that prints the number of digits.

```
int count_digit(int a);
```

```
asm2022$ ./count_digit 1234
4
asm2022$ ./count_digit 42
2
asm2022$ ./count_digit
"Only one argument is needed"
```

6.4.4 Reverse string

Write the function that print the reversed string.

```
1 char *reverse_string(char *str);
```

```
asm2022$ ./rev_str 'toto'
otot
asm2022$ ./rev_str 'ASM2022asm'
msa2202MSA
asm2022$ ./rev_str '*_3210123_*'
*_3210123_*
asm2022$ ./rev_str
"Only one argument is needed"
```

6.4.5 All permutations

Given a string (without repeating characters), print, one by one, all the permutations of the characters in that string. The order of permutations is not important.

```
1 void all_permutations(char *str);
```

```
asm2022$ ./allperm
asm2022$ ./allperm "abc"
abc
acb
bac
bca
cab
cba
```

6.5 More ?

If you want to train more on that matter, you can check website like:

- Codingame
- Leetcode
- Interviewcake
- ...