TP1: Unix and C programming

Unix : first use

Remark : *you can skip this part if you are familiar with this*

a. Open a new session, create an account and connect you with this account. Create a directory `C_programming` and then a sub-directory `td1/`. All the files you will write today should be saved in this sub-directory.

b. Try the commands `ls`, `cd`, `mkdir`, `rm`, `echo`... (for example, in your home directory, create a directory `foo`, copy the file `/etc/passwd`, look at this with `less` and remove it).

c. What is automatic completion ? Do not forget to use it, it saves a lot of time.

d. What does the command `history` ? With >, direct the output of a command in a file of your choice. Use `grep` to find which are the commands typed containing "ls". Do something similar using | instead of >.

Your first C program

Reminder : save your files often.

a. Launch your favorite text editor (if you do not know what to use, you can try `emacs`). Copy the following program in a file.

```
#include<stdio>
#include<stdlib>

int main(){
  printf("Hello\n");
  exit(EXIT_SUCCESS);
}
```

Compile it using the following command and try to execute it.

```
gcc -W -Wall file.c -o hello
```

Why should you always use the options `-W` and `-Wall` ?

b. Create a file "Makefile" containing an entry to execute and an other to erase the program.

Printf, scanf, recursive functions

Let us define $u_{n+2} = u_{n+1} + u_n$ and $u_0 = u_1 = 1$ (the Fibonacci sequence).

a. Write a recursive function `int u(int n)` that computes u_n.

b. Add a command to print something (say n) each time your program enters the function `u()`. Is your program efficient ? What is the complexity of your algorithm ?

c. Write an iterative function to compute u_n. What is the complexity of this version ?

d. Using `scanf`, modify your program to print "n=" and then compute u_n for the value you just entered.

Program arguments

a. Write a new program with a function `int main(int argc, char ** argv)`. What are `argc` and `argv` ?

b. Write a program `add` that compute the sum of its arguments and print the result (for example `add 1 2.3 -1` should write 2.3). You can use the function `double atof(char *)`