TP - 4

2 octobre 2014

L'objectif de ce TP est de programmer quelques fonctions simples en C pour vous familiariser avec sa syntaxe. Vous aurez besoin des points de syntaxes donnés dans le cours de Jean Goubault. Je vous met également un fichier à disposition ici qui contient de petits exemples de programmes C utilisant les constructions `if, for, while, switch` ainsi que des tableaux statiques.

Dans ce TP, vous n'avez pas et vous ne devez pas utiliser de *pointeurs* (ou du moins en déclarer un). On laisse ça pour la prochaine séance.

Exercice - 1 *Compilation*

Comme pour le dernier TP, on commence par apprendre à compiler et exécuter un simple Hello World. Il n'y a pas de Top Level pour le C alors cette fois on est obligé. C'est un argument supplémentaire pour que vous compreniez qu'il faut *vraiment* écrire vos programmes OCaml dans un source pour le compiler ensuite !

1- Ouvrez un fichier `hello.c` et recopiez ce programme :

```
#include <stdio.h>

int main () {
  printf("Coucou\n");
  return 0;
}
```

2- Dans un terminal, entrez `$ gcc -o hello.out hello.c` pour compiler votre programme vers l'exécutable `hello.out`.

3- Lancez l'exécutable avec `$ ./hello.out`.

Exercice - 2 *On commence doucement*

1- Ecrivez une fonction qui calcule le maximum de 3 entiers. Testez.

2- Ecrivez une fonction qui calcule la somme de 3 entiers. Testez.

3- Ecrivez une fonction qui calcule le reste modulaire de deux entiers sans utiliser l'opérateur %. Test...

4- Ecrivez une fonction qui calcule le maximum d'un nombre quelconque d'entiers donnés sous la fome d'un tableau d'entiers. Son *prototype* doit être :

```
int max_t (int tab[], int taille)
```

- Le type `int tab[]` dénote un argument `tab` correspondant à un tableau d'entiers.

5- Testez cette fonction en l'appelant sur les arguments de l'exécutable (ex. `$ ./exo2.out 10 -18 14` doit renvoyer 14).

- Le prototype typique d'une fonction main est : `void main (int argc, char *argv[])`. L'argument `argc` (pour *argument count*) contient le nombre de paramètres passés à l'exécutable (ex. avec `$ ./a.out param1 param2`). L'argument `argv` (pour *argument values*) est un tableau contenant les `argc` paramètres sous la forme de chaînes de caractères. Vous pouvez transformer cette chaîne en entier avec `atoi()` (pour *ascii to int*). Attention, l'exécutable prend toujours un premier paramètre implicite correspondant au nom de l'exécutable.

Exercice - 3 *Cribles*

1- Répondez à la question Ex.2/1 du TP1 en utilisant un tableau.
2- En utilisant le crible d'Erastostène, affichez tous les nombres premiers plus petits que 1000.
3- (*) Affichez un diagramme (en bâton ?) représentant la proportion de nombres premiers par tranches de 1000 entre 2 et 10000.

Exercice - 4 *Bitmap*

Dans cet exercice, on va exploiter les *bitmap* pour résoudre quelques problèmes simples. Un *bitmap* est une structure de données permettant de stocker de façon très compacte des tableaux de booléens. L'idée est simplement de représenter un tableau de bits par un entier dont la représentation binaire correspond au tableau. L'objectif est de recoder quelques fonctions de base pour les bitmap. Vos fonctions doivent être très efficaces (la plupart du temps en un cycle CPU). Conseil : pensez aux opérateurs bits à bits.
1- Ecrivez une fonction qui change la valeur d'une case d'un bitmap.
2- Ecrivez une fonction qui affecte un certain bit pour une certaine position dans un bitmap.
3- Ecrivez une fonction qui concatène deux bitmap.
4- Ecrivez une fonction qui transforme un tableau de bool en un bitmap et inversement.
5- Reecrivez le crible d'Erastostène avec un bitmap.
6- Mettez en valeur le gain en temps de cette version. A votre avis, d'où vient ce gain ?

Si vous voulez approfondir ce langage par vous même, je vous conseille de piocher ce que vous voulez dans ce cours.