

Programmation Procédurale

Feuille 4

Programmation d'ordre supérieur

Pointeurs

Exercice 1: une version simplifiée de printf

Il s'agit ici de coder des fonctions à nombre variable d'arguments, à l'aide des macros du fichier standard `<stdarg.h>`.

- coder la fonction `void CatStrings(char *str1, ...)` qui affiche à la suite tous ses paramètres jusqu'à trouver le pointeur nul. Par exemple pour afficher "essai" on veut pouvoir coder:
`CatStrings("es", "sai", NULL)`.
- coder la fonction `void Printf(char *format, ...)` qui se comporte comme `printf` et reconnaît dans son format les séquences `"%%"`, `"%c"`, `"%s"`, `"%x"` et `"%d"`.

Exercice 2: la fonction map

Écrire la fonction `map` qui applique une fonction sur un tableau d'entiers. Cette fonction prend en paramètre une fonction `fct`, un tableau d'entiers et sa taille. La fonction `fct` prend un entier en paramètre et retourne un entier. Soit le tableau `tab`, tel que

```
tab = {1, 2, 3 , 4, 5};
```

alors l'appel `map(carre, tab, 5)` (où `carre` est une fonction qui renvoie le carré de son paramètre) modifie le tableau `tab` en

```
tab = {1, 4, 9 , 16, 25};
```

Pour tester votre programme, vous écrirez une fonction qui affiche le contenu de votre tableau. Votre fonction d'affichage pourra être écrite à l'aide de la fonction `map`!

Exercice 3: la fonction iterate

Écrire une fonction `iterate` qui calcule l'itération d'une fonction à deux arguments sur un tableau d'entiers. La fonction `iterate` prend en paramètre une fonction `f`, un tableau d'entiers et sa taille. La fonction `f` prend deux entiers en paramètre et retourne un entier. Si le tableau contient les valeurs $x_0, x_1, x_2, \dots, x_n$, la fonction `iterate` retourne la valeur

$$f(f(\dots f(f(x_0, x_1), x_2), \dots), x_n).$$

Si la taille du tableau est respectivement 1, 2, 3 ou 4 la fonction `iterate` retourne respectivement les valeurs $x_0, f(x_0, x_1), f(f(x_0, x_1), x_2)$ et $f(f(f(x_0, x_1), x_2), x_3)$.

Comment utiliser cette fonction `iterate` pour calculer, le maximum, la somme ou le produit des éléments d'un tableau?

Exercice 4: Fonctions str...

Réécrire les fonctions suivantes à l'aide de pointeurs (char *):

```
char *strcpy(char *dest, const char *src);
size_t strlen(const char *s);
char *strdup(const char *s);
char *strchr(const char *s, int c);
```

Consulter le manuel Unix pour vérifier leur comportement.

Exercice 5: Commande echo

Ecrire la commande echo de Unix en gérant l'option -n. On rappelle que le nombre et la valeur des arguments d'un programme sont accessibles au travers des 2 premiers paramètres de la fonction main.

Exercice 6: Analyseur d'options UNIX

Ecrire la commande option qui analyse les options et les paramètres passés sur la ligne de commande et qui les affiche sur la sortie standard. On se comportera ici suivant les conventions Unix habituelles:

- une option courte commence par un tiret (“-”) et fait un caractère.
- plusieurs options courtes peuvent être concaténées (e.g. “-abc” est équivalent à “-a -b -c”)
- une option longue commence par un double tiret (“--”)
- les options sont placées avant les paramètres
- la fin des options est marquée par la présence du premier paramètre ou par la rencontre de la séquence spéciale “--”

Exemple

```
$ options -xdf -y - -kj toto -z
Option courte: x
Option courte: d
Option courte: f
Option courte: y
Option courte: k
Option courte: j
Argument: toto
Argument: -z
$ options -ab --ab foo -ab
Option courte: a
Option courte: b
Option longue: --ab
Argument: foo
Argument: -ab
$ options -- -a
Argument: -a
$
```

Exercice 7: Commande cat

En utilisant l'analyseur d'option de la question précédente, écrire une version simplifiée de la commande cat. Ce programme ne gèrera que les options '-n' et '-E' de la commande standard (Cf. manuel). Rajouter ensuite la commande '-h' permettant d'afficher de l'aide sur la commande.