TP 6 : Révision

Programmation en C (LC4)

Semaine du 2 mars 2008

Ce TP sera corrigé par l'ordinateur. L'objectif est de vous donner une note indicative correspondant à votre niveau actuel. Pour que tout fonctionne correctement, il vous suffit de nommer vos fichiers et les fonctions comme indiqué dans l'énoncé. Vous devez créer un fichier par exercice. Pour connaître votre note, téléchargez le fichier suivant :

```
http ://www.tabareau.fr/tp6.tar
```

Copiez ce fichier dans votre répertoire de travail (celui où se trouve vos fichiers .c) et tapez :

```
# tar xf tp6.tar
# make -f notes
```

Attention, il faut que vous ayez commenté la fonction "main" de vos fichiers ".c". Ouvrez un fichier pour tous les exercices, si vous voulez tester vos fonctions au fur et à mesure.

Exercice 1 — ***tableau.c (très facile)*** Ecrire une fonction
`int compte_zero(int t, int* tab)`
qui compte le nombre de zéro d'un tableau *tab* de taille t.

Exercice 2 — ***copie.c (facile)*** Ecrire une fonction
`int* copie (int t, int* tab)` qui
alloue un nouveau tableau de taille `t` et y copie le contenu de `tab`.

Exercice 3 — ***fusion.c (moyen)*** Ecrire une fonction
`int* fusion (int t1, int* tab1, int t2, int* tab2)`
qui alloue un nouveau tableau de taille $t1 + t2$ et y copie dans l'ordre croissant les éléments de $tab1$ et $tab2$. On suppose que $tab1$ et $tab2$ sont triés par avance.

Exercice 4 — ***chaine.c (facile)*** Ecrire une fonction `char* copie_chaine (char* s)` qui alloue une nouvelle chaîne de caractère semblable à celle passée en paramètre.

Exercice 5 — ***bits.c (moyen)*** Écrire une fonction
`char* chaine_binaire (char zero, char un, int n)`
qui construit une chaîne de caractère correspondant à la représentation binaire de n. On utilisera *zero* pour représenter 0 et *un* pour représenter 1. On suppose que n est stocké sur 32 bits.

Exercice 6 — ***pointeur.c (moyen)*** Écrire une fonction
`int** tri (int t, int* tab)`
qui calcule un tableau de pointeur de taille t à partir de *tab*, lui aussi de taille t. Dans la case i du tableau, on trouve un pointeur vers le plus petit élément de *tab* qui est strictement plus grand que $tab[i]$. Si cet élément n'existe pas alors cette case vaut NULL.