Programmation II
Examen

15 mai 2017

Aucun document autorisé. Durée 2h.
Page suivante se trouvent des prototypes de fonctions utilisables.
Dans tout code, en cas d'erreur (interne ou provenant d'une fonction de bibliothèque) on arrêtera le programme après avoir affiché un message approprié.

Exercice 1 : lecture d'un tableau depuis un fichier

Un fichier binaire (pas texte !) contient des entiers (positifs ou négatifs). Chacun fait 4 octets et est stocké conformément à l'architecture de la machine (*big endian* ou *little endian*).

Écrire une fonction qui prend un seul paramètre : le nom du fichier, puis alloue et retourne un tableau (dynamique) constitué des entiers du fichier. On retournera `NULL` en cas de fichier vide.

Exercice 2 : représentation d'une matrice

Disserter sur les différentes façons de stocker en mémoire une matrice réelle (cases de type `double`). Pour chaque possibilité, dessiner la représentation en mémoire (en précisant les segments mémoire) de la matrice `Mat` :

1.1	1.2
2.1	2.2

Exercice 3 : cosinus

Expliquer ce que doit faire un programmeur afin de pouvoir utiliser (proprement) la fonction cosinus dans un programme. Par exemple, il veut pouvoir écrire cette ligne pour calculer $cos\left(\frac{\pi}{7}\right)$:
`double pi7 = cos( M_PI / 7.0 );`
Justifier chaque chose.

Exercice 4 : arbre d'arité quelconque

1. Écrire le type `struct noeud` d'un nœud d'un arbre où chaque nœud contient une valeur de type `double` et peut avoir une arité quelconque, c'est-à-dire un nombre de fils inconnu à la compilation. De plus on doit pouvoir ajouter et supprimer des fils lors de l'éxécution
2. Écrire la procédure `void ajoutNoeud(struct noeud *pere, double val)` qui ajoute un fils au nœud `pere`. Ce nouveau fils est une feuille contenant la valeur `val`.
3. Écrire la fonction `int recherche(struct noeud *racine, double val)` qui renvoie `1` si la valeur `val` est contenue dans au moins un nœud de l'arbre dont la racine est passé en paramètre, `0` sinon.
4. Écrire la fonction `int prof(struct noeud *racine)` donnant la profondeur de l'arbre dont la racine est passé en paramètre, c'est-à-dire la longueur d'un plus long chemin entre une feuille et la racine. Un arbre à un seul nœud est de profondeur 1.
5. Dire comment un tel arbre pourrait être stocké dans un fichier et rechargé. On ne demande pas nécéssairement du code (bien que la réponse puisse être constituée de code seulement) mais de discuter sur les problèmes rencontrés et comment les résoudre.

Exercice 5 : tac

Écrire un programme (semblalble à l'utilitaire Unix `tac`) qui réalise une copie d'un fichier en inversant l'ordre des lignes. Par exemple le résultat de `./tac entree.txt sortie.txt` sera :

entree.txt	*sortie.txt*

`The only way to learn a new programming`
`language is by writing programs in it.`
`The first program to write is the`
`same for all languages:`
`   Print the words`
`      hello, world`
	`      hello, world`
	`   Print the words`
	`same for all languages:`
	`The first program to write is the`
	`language is by writing programs in it.`
	`The only way to learn a new programming`

On supposera, sans avoir à le vérifier, que les deux paramètres de la ligne de commande seront toujours donnés, et que les lignes du fichier d'entrée font 80 caractères maximum.

—

Fonctions utiles pour tous les exercices. Vous avez le droit d'utiliser d'autres fonctions de la bibliothèque standard. Vous pouvez poser pendant l'examen des questions de syntaxe auxquelles le manuel Unix `man` a la réponse.

```
#include <stdio.h>
#include <stdlib.h>
#include <err.h>

void *calloc(size_t nmemb, size_t size);
void err(int codeRetour, const char *format, ...);  // ajoute message selon errno
void errx(int codeRetour, const char *format, ...); // n'ajoute pas message selon errno
int fclose(FILE *fd);
int feof(FILE *fd);
char *fgets(char *s, int size, FILE *fd); // lit au max size-1 caracteres et ajoute \0
FILE *fopen(const char *path, const char *mode); // mode = "r" "w" "a" "r+" "w+" ou "a+"
int fputs(const char *s, FILE *fd);
size_t fread(void *ptr, size_t size, size_t nmemb, FILE *fd);
void free(void *ptr);
int fseek(FILE *fd, long offset, int whence); // whence = SEEK_SET SEEK_CUR ou SEEK_END
long ftell(FILE *fd);
size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *fd);
void *malloc(size_t size);
size_t strlen(const char *s);
```