TP9 - Manipulation de fichiers

Langage C (LC4)

Semaine du 29 mars 2010

1 Déplacement dans un fichier

Les fonctions utilisées jusqu'ici parcourent un fichier séquentiellement du début à la fin, en déplaçant implicitement un curseur dans le fichier. On peut jouer avec ce curseur grâce à la fonction `int fseek(FILE *flot, long decalage, int origine)`. La valeur renvoyée est 0 en cas de succès, autre chose sinon. Cette fonction permet de se positionner en `origine + decalage`. Le paramètre `decalage` est exprimé en octets et `origine` peut prendre les 3 valeurs suivantes :

`SEEK_SET` : début du fichier ;

`SEEK_CUR` : position courante ;

`SEEK_END` : fin du fichier ;

Pour connaître la position courante à partir du début du fichier, on utilise `long ftell(FILE *f)`.

Question 1. Écrivez une fonction `void moities(FILE *f1, FILE *f2, FILE *f3)` qui écrit la première moitié de `f1` dans `f2` et la seconde moitié de `f1` dans `f3`.

Question 2. Écrivez une fonction `int main(int argc, char *argv[])` qui appelle la fonction `moities` sur trois fichiers dont les noms sont passés en argument au programme.

Question 3. Écrivez une fonction `void renverse(FILE *f1, FILE *f2)` qui lit le texte contenu dans `f1` et l'affiche à l'envers caractère par caractère sur `f2`.

On définit un type liste de caractères ainsi :

```
typedef struct carac *buffer;
struct carac {
  char val;
  buffer suivant;
};
```

On utilisera cette structure de données comme un tampon du genre LIFO (last in first out).

Question 4. Écrivez une fonction `buffer ajouter(char c, buffer b)` qui ajoute `c` en tête de `b` et renvoie un pointeur vers la tête du buffer.

Question 5. Écrivez une fonction récursive `void ecrire(buffer b, FILE *f)` qui écrit le contenu de `b` dans `f` et libère la mémoire occupée par le buffer `b`.

Question 6. En utilisant `buffer`, écrivez une fonction **`void`** `renverse_lignes(FILE *f1, FILE *f2)` qui écrit dans `f2` les lignes de `f1` en partant de la dernière pour finir par la première.

Question 7. Même question, mais sans utiliser `buffer`. Pour attraper une ligne, vous pourrez utiliser `fscanf` avec le format `%m[^\n]` ; on rappelle que le `m` vous dispense de l'allocation de mémoire.

Question 8. En utilisant `buffer`, écrivez une fonction **`void`** `droite_gauche(FILE *f1, FILE *f2)` qui réécrit dans `f2` les lignes de `f1` dans l'autre sens.

Question 9. Même question, mais sans utiliser `buffer`.

Question 10. Discutez des avantages et des inconvénients des 2 méthodes (avec ou sans `buffer`).