Programmation C
TP n° 10 : Fichiers

Ceci est un sujet d'appoint, la priorité est de finir les exercices 1, 2, 3 et 4 du TP précédent.

Exercice 1 : Lecture et écriture

1. Écrire un programme `copy.c` qui attend deux noms de fichier en argument, `fic1` et `fic2`, et recopie le contenu de `fic1` dans `fic2` en écrasant le contenu original de `fic2`, s'il y en avait un. Si `fic2` n'existe pas il doit être créé. Si un seul nom de fichier est passé en argument, `fic`, est donné en argument alors il faudra copier l'entrée standard dans `fic`. Utiliser les fonctions `fputc` et `fgetc`.
2. Écrire un programme `copy2.c` qui se comporte comme `copy` et qui calcule la taille du fichier grâce à `fseek` et `ftell` ; alloue un tampon de cette taille et réalise une seule lecture et une seul écriture pour faire la copie.
3. En utilisant la fonction `time` de la librairie `time.h`, comparer les temps d'exécution des deux programmes précédents sur les mêmes fichiers.
4. Écrire un programme `my_cat.c` qui attend un nom de fichier et affiche le contenu en numérotant les lignes. Si aucun argument n'est passé en paramètre, il faudra lire sur l'entrée standard. Vérifier que le résultat est le même qu'en utilisant la commande `cat -n`. Les lignes sont comptées à partir de 1.
5. Écrire un programme `somme.c` qui attend un nom de fichier en argument. On supposera que le fichier ne contient que des entiers séparés par des caractères d'espacement (retour à la ligne, espace, etc. . .). Le programme doit calculer la somme de ces entiers et écrire le résultat dans un fichier `resultat`. Voici un exemple d'exécution :

```
$> cat data
1␣2
3␣
␣4
5
$> ./somme data ; cat resultat
15
```

6. Écrire un programme `get_line.c` qui attend un entier n et un nom de fichier en argument et affiche la ligne numéro n du fichier sur la sortie standard. Il faut : trouver la position du début de la ligne ; mesurer sa taille ; allouer un tampon suffisant ; repositionner le curseur et utiliser `fgets` pour lire la ligne en une fois. Si la ligne n'existe pas, le programme affiche un message d'erreur.
7. Écrire un programme `mini-grep.c` qui attend une chaîne `c` de caractères et nom de fichier et affiche sur la sortie standard toutes les lignes qui contiennent au moins une fois la chaîne `c`. Nous supposons qu'une ligne ne dépasse pas 512 caractères.
8. Écrire un programme `mini-sed.c` qui attend deux chaînes `c1` et `c2` et un nom de fichier `fic` en argument et remplace toutes les occurrences de `c1` par `c2` dans `fic`. Nous supposons qu'une ligne ne dépasse pas 512 caractères.

Exercice 2 : Pour aller plus loin

Nous souhaitons une solution pour que `mini-grep.c` et `mini-sed.c` fonctionnent avec des lignes de taille arbitraire. Deux solutions sont possibles. Soit calculer la taille de la ligne la plus longue et utiliser `fgets` pour manipuler le fichier ligne par ligne. Ou bien utiliser un tampon pour lire le fichier par morceaux de taille arbitraire et gérer le cas où le motif est coupé entre deux lectures. Par exemple : si on cherche le motif `"toto"` dans le fichier suivant en faisant des lectures avec un tampon de taille 5. Alors il sera découpé comme suit. Le motif `"toto"` est bien présent mais il est réparti sur deux lectures.

```
|Ce te|xte p|arle |de la| vie |de to|to et| de s|on fr|ère.
```