TP 7 – Fichiers

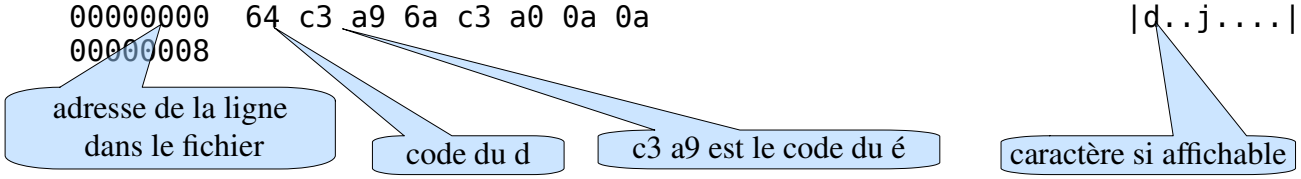
1. Exercice - encodage des fichiers

1.1. Avec gedit saisissez le mot « déjà » dans un fichier que vous nommerez « deja_utf8.txt » en l'enregistrant en choisissant « encodage utf-8 ».
Dans une console, taper

```
hexdump -C deja_utf8.txt
```

ce qui vous donnera :

```
00000000  64 c3 a9 6a c3 a0 0a 0a                           |d..j....|
00000008
```



1.2. Faites la même chose en enregistrant cette fois-ci votre fichier en choisissant « encodage iso8859-15 ». Analyser le fichier de nouveau avec hexdump.

1.3. Éditer ce programme (disponible dans l'archive) avec gedit et l'enregistrer en choisissant « encodage utf-8 », puis le compiler et le tester.

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


/** main program for debug purpose
@return : void
*/
int main(int argc, char** argv) {
    char ch[]="déjà" ;
    printf("%s %d\n", ch, strlen(ch)) ;
    return (EXIT_SUCCESS) ;
}
```

1.4. Ouvrir ce programme avec gedit et enregistrez le sous ex1_iso.c en choisissant « encodage iso8859-15 ». Le compiler le puis le tester. Dans le menu Terminal de la console choisissez « encodage iso-8859-15 » puis tester de nouveau le programme.

2. Recherche dans un fichier

Vous trouverez 3 programmes dans l'archive :
ex21.c : crée 2 fichiers, l'un texte, l'autre binaire, contenant les nombres de 45 à 54.
ex22_bin.c : ouvre et affiche le contenu du fichier binaire.
ex22_txt.c : ouvre et affiche le contenu du fichier texte.

1) **Modifier** le programme ex21.c afin qu'il tire 100 nombres aléatoires compris entre -300 et +300 et les stocke dans un fichier binaire et dans un fichier texte. Vous utiliserez les programmes ex22_bin.c et ex22_txt.c pour afficher ces nombres.

Les 2 fichiers contiennent les mêmes valeurs mais codées différemment. Ouvrez le fichier texte en double cliquant dessus pour vérifier qu'il contient les mêmes valeurs que celles affichées. Notez également que la taille du fichier binaire est de 400 octets *(pourquoi d'ailleurs ?).*

2) **Écrire** la fonction *`void afficherNtxt (FILE* f, int n )`* qui affiche le nième nombre du fichier texte *f*. Tester cette fonction dans le programme ex22_txt.c. Vous utiliserez la fonction standard *`fseek`* afin de vous positionner au début du fichier.

3) **Écrire** la fonction *`void afficherNbin (FILE* f, int n )`* qui affiche le nième nombre du fichier binaire *f*. Tester cette fonction dans le programme ex22_bin.c. Vous utiliserez la fonction standard fseek afin de vous positionner sur le Nième élément.

Pourquoi ne peut-on pas utiliser fseek pour se positionner sur le nième élément du fichier texte ?

3. Fichiers CSV

Les fichiers CSV sont des fichiers rudimentaires qui peuvent être interprétés par des tableurs comme Excel. CSV veut dire « Comma Separated Value », ie. valeurs séparées par des virgules.
Les valeurs des cellules sont séparées par des virgules et les lignes par des saut de ligne '\n' .

Par exemple, pour représenter le tableau suivant :

échantillon	tension	courant
1	5.0	1.0
2	4.9	1.1
3	5.1	0.9
4	3.2	1.3

le fichier CSV (qui est un fichier texte) contiendra :

```
"échantillon", "tension", "courant"
1, 5.0, 1.0
2, 4.9, 1.1
3, 5.1, 0.9
4, 3.2, 1.3
```

Recopier le fichier texte ci-dessus dans un éditeur comme geany et nommer le mesures.csv. Ouvrez le maintenant avec Libre office et vérifier la présentation sous forme de tableau.

Écrire un programme qui génère un fichier nommé « multiplication.csv ». Ce fichier sera au format CSV et contiendra la table de multiplication des nombres de 1 à 10. Vérifiez avec Libre Office que vous pouvez ouvrir le fichier. Vous pouvez également ouvrir votre fichier CSV avec gedit en faisant « clic-droit, ouvrir avec ».

4. Exercice - lecture séquentielle d'un fichier texte.

Écrire un programme qui lit un fichier texte et produit un autre fichier texte contenant un mot par ligne. On utilisera les étapes suivantes :

- Tant qu'il y a un mot à lire :
  - lire un mot (utilisation de fscanf)
  - l'écrire dans le fichier destination
  - écrire « \n » dans le fichier destination

Pour lancer votre programme, vous taperez dans une fenêtre de commandes :

```
prog74.bin  fichier.txt  fichierMot.txt
```

5. Exercice – allocation dynamique de chaînes de caractères (pour les plus rapides).

Écrire un programme qui lit un fichier texte et produit un autre fichier texte dans lequel tous les mots sont mélangés. On utilisera les étapes suivantes :

- utilisation de la fonction *stat (ou fstat)* pour connaître la longueur du fichier (s'inspirer du programme donné en exemple dans l'archive du TP) ;
- allocation dynamique d'un tableau d'octets de la longueur du fichier ;
- lecture en une seule fois du fichier pour le stocker dans le tableau (utilisation de la fonction fread) ;
- compter le nombre de mots
- créer un tableau de mots par allocation dynamique. L'espace réservé à chaque mot sera lui aussi dynamique.
- produire un fichier contenant tous les mots rangés dans un ordre aléatoire.

Le nom du fichier source sera demandé à l'utilisateur et le nom du fichier produit sera celui du fichier source suffixé de « .mel ». Exemple « monFichier.txt » deviendra « monFichier.txt.mel »