Université Paul Sabatier TOULOUSE III

L2 INFO S3

Programmation impérative en langage C

feuille de **TP n° 8 :** Fichiers texte

---

1 Fichiers textes

Lors de l'exécution d'un programme, les données stockées en mémoire sont perdues dès la sortie du programme. Par contre, les fichiers sur support magnétique (bande, disquette, disque) sont conservables, mais au prix d'un temps d'accès aux données très supérieur. On utilisera dans ce qui suit des fichiers textes (par opposition à binaire), séquentiels (par opposition à accés direct) et bufférisés : Les opérations d'entrée / sortie sur ces fichiers se font par l'intermédiaire d'un "buffer" (bloc en mémoire) géré automatiquement ; ceci signifie qu'une instruction d'écriture n'impliquera pas une écriture physique sur le disque mais dans le buffer, avec écriture sur disque uniquement quand le buffer est plein.
Selon le système d'exploitation, les fins de ligne sont gérées différemment :

- Sous DOS et Microsoft Windows, la fin de ligne est indiquée par un retour chariot suivi d'un saut de ligne (CRLF).
- Sous Unix, la fin de ligne est indiquée par un saut de ligne (LF).

Les fichiers sont identifiés par un pointeur sur une structure FILE (prédéfinie par un typedef dans stdio.h). Les fonctions disponibles sont prototypées dans stdio.h.

1.1 Ouverture du fichier

```
FILE* le_fic;
le_fic= fopen("C :\\essai.txt", "r");
```

2 étapes :

- Un fichier en C est associé à une variable de type FILE* : identificateur de fichier ou nom interne
- L'ouverture se fait grâce à la fonction fopen qui prend 2 paramètres :
  1. Le nom externe du fichier avec son chemin
  2. Le type d'accès :
     Les trois principaux sont :
     · **r** :lecture seule
     · **w** :écriture, si le fichier existe il est d'abord vidé
     · **a** :append, écriture à la suite du contenu actuel et création si inexistant

Remarque : En cas d'erreur, le pointeur NULL est retourné, le type d'erreur est donné dans la variable globale `errno`, détaillée dans errno.h.

```
#include <errno.h>
if (le_fic==NULL) printf("numero d'erreur=%d\n",errno);
```

Le numéro de l'erreur obtenue permet de connaitre la cause de l'erreur : voir errno.h.

1.2 Lecture dans le fichier

La fonction fscanf, analogue à scanf, permet de lire des données dans un fichier. Sa syntaxe est semblable à celle de scanf :

```
fscanf(nom interne,"chaîne de contrôle",argument-1,...,argument-n)
```

Les spécifications de format sont ici les mêmes que celles de la fonction scanf.

1.3 Ecriture dans le fichier

La fonction fprintf, analogue à printf, permet d'écrire des données dans un fichier. Sa syntaxe est :

```
fprintf(nom interne,"chaîne de contrôle",expression-1, ..., expression-n)
```

- Les spécifications de format utilisées pour la fonction fprintf sont les mêmes que pour printf.
- rend le nb d'octets écrits, ou EOF si erreur.

- Les \n sont transformés en CR/LF.

1.4 Fermeture du fichier

```
FILE* le_fic;
le_fic= fopen("C :\\essai.txt", "r");
...
fclose(le_fic);
```

1.5 Destruction physique d'un fichier

On peut à partir d'un programme C supprimer un fichier ou bien encore le renommer :

```
rename("essai.txt", "nouveau.txt");
remove("nouveau.txt");
```

1.6 Détection de fin de fichier

```
FILE* le_fic ;
le_fic= fopen("C :\\essai.txt", "r") ;
...
if ( !feof(le_fic)) printf("pas fini") ;
fclose(le_fic) ;
```

1.7 Exemples

✓ **Exemple 1** *Le programme suivant lit le contenu du fichier texte entree, et le recopie caractère par caractère dans le fichier sortie :*

```
#include <stdio.h>
#include <stdlib.h>
#define ENTREE "entree.txt"
#define SORTIE "sortie.txt"

int main(void)
{
FILE *f_in, *f_out ;
f_in = fopen(ENTREE,"r") ;
if (f_in == NULL)
    {
    fprintf(stderr, "\n Erreur : Impossible de lire le fichier %s\n",ENTREE) ;
    exit(EXIT_FAILURE) ;
    }
f_out = fopen(SORTIE,"w") ;
if (f_out == NULL)
    {
    fprintf(stderr, "\n Erreur : Impossible d'ecrire dans le fichier %s\n",  SORTIE) ;
    exit(EXIT_FAILURE) ;
    }
do
    {
    char c = fgetc(f_in) ;
    if ( !feof(f_in)) fputc(c, f_out) ;
    }
while ( !feof(f_in)) ;
fclose(f_in) ;
fclose(f_out) ;
return(EXIT_SUCCESS) ;
}
```

◇ Exercice **1** : Fichier d'indicatifs téléphoniques internationaux : le fichier texte "paysphone.bip" contient sur chaque ligne le nom d'un pays, sa capitale ainsi que l'indicatif téléphonique qui lui est associé ; par exemple il peut contenir :

Italie ;	Rome ;	39
Espagne ;	Madrid ;	34
France ;	Paris ;	33
Cuba ;	La Havane ;	53
Russie ;	Moscou ;	7
Royaume-Uni ;	Londres ;	44
Turkmenistan ;	Achgabat ;	993
Etats unis ;	Washington ;	1
Australie ;	Camberra ;	61
Moldavie ;	Chisinau ;	373

On souhaite écrire un programme qui gère ce fichier. L'utilisateur doit pouvoir tant qu'il le souhaite :
– déterminer si un pays tapé au clavier est déjà présent dans le fichier.
– ajouter un pays au fichier en tapant les renseignements correspondants au clavier.
– trouver dans le fichier puis afficher l'indicatif téléphonique d'un pays tapé au clavier.

Vous devez :
- afficher un menu du type suivant :

```
Choisissez une option :
   1. Vérifier si un pays est dans la base
   2. Ajouter un pays à la base
   3. Trouver l'indicatif d'un pays
   4. Quitter le programme
```

- utiliser une structure à 3 champs par pays :

```
typedef struct
{    char nom[20+1] ;
     char capitale[20+1] ;
     int indicatif ;
} T_pays ;
```

- imaginer des sous-programmes adaptés à l'exercice que vous prendrez soin de tester au fur et à mesure de leur écriture.

Remarques :

- Au départ, le fichier n'existe pas ; c'est au programme de le créer.
- On supposera que les noms de pays et de capitale ne dépassent pas 20 caractères.(char nom[21])
- Les indicatifs téléphoniques sont des nombres compris entre 1 et 999

◇ Exercice **2** :

1. Ecrire un programme qui lit le contenu d'un fichier texte et :
   - compte le nombre total de mots
   - compte le nombre total de phrases
   - écrit dans le fichier "liste_mots.txt" chaque mot rencontré : on reviendra à la ligne aprés chaque mot rencontré.

   On utilisera au minimum les 2 fonctions booléennes suivantes :
   · bool caractere_dans_mot(unsigned char c) qui détermine si le caractère c est susceptible d'être dans un mot ; il peut s'agir :
     · d'une lettre minuscule ou majuscule
     · d'un tiret (mot composé)
     · d'une lettre accentuée
   · bool fin_de_phrase(unsigned char c) qui détermine si le caractère c marque la fin d'une phrase : point, point d'exclamation,point d'interrogation.

   Exemple :

```
Taper le nom du fichier a traiter : test.txt
Nombre de mots : 47
Nombre de phrases : 2
```

2. Modifier le programme existant pourqu'il affiche également le nombre d'occurrences de mots en fonction de leur nombre de lettres :

```
Taper le nom du fichier a traiter : test.txt
Nombre de mots : 47
Nombre de phrases : 2
Nombre moyen de mots par phrase : 23.5
Longueur moyenne des mots : 4.468
Fréquence en fonction de la longueur :
1 : 6
2 : 9
3 : 4
4 : 8
5 : 5
6 : 4
7 : 5
9 : 3
10 : 2
11 : 1
```

3. Est-il vrai que le nombre moyen de mots par phrases est deux fois plus important dans "Albertine disparue" de Marcel Proust que dans "Germinal"(1885) de Zola(1840-1902) ?

4. Dans lequel de ces deux romans trouve t-on le mot le plus long ? Y-en-a-t-il plusieurs ? Quels sont-ils ?