Programmation

En Langage C

Chap 6 – Les Fichiers

Cours TD TP

Objectif :

conserver et exploiter les données de manière permanente à l'aide des fichiers

Sommaire

Chap VI : Les fichiers

- Définition
- Buts recherchés
- Caractéristiques et classification
- Manipulation d'un fichier
- Fonctions d'écriture
- Fonctions de lecture
- Positionnement dans un fichier
- Application

Chap 6 : Les fichiers

Définition

Un fichier est un ensemble **structuré** de données stocké sur un support externe (disquette , disque dur, …). Un fichier structuré est composé d'une suite **d'enregistrements** homogènes qui regroupent le plus souvent plusieurs composantes appelées **champs**.

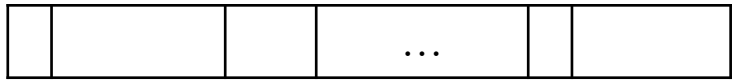
Buts

- Garantir la **persistance** des données (les données sont préservées après l'exécution)
- Volume de données variable et plus grand que pour les structures de données statiques (tableaux).

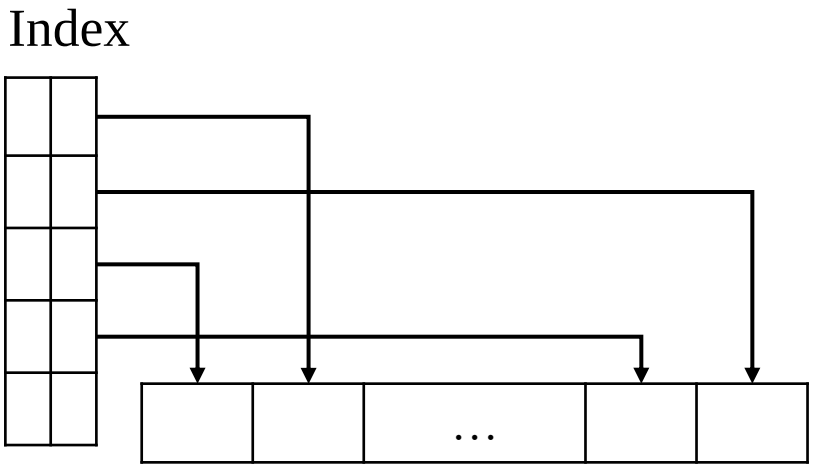
Caractéristiques

Les fichiers sont caractérisés par deux notions :

- **le mode d'organisation** : est la méthode choisie pour ranger les données dans le fichier (séquentiel, indexé,...) ;



Séquentiel



Indexé

- **le mode d'accès** : est la méthode choisie pour rechercher un enregistrement en cas de lecture ou pour rechercher son emplacement en cas d'écriture (séquentiel, direct, indexé, ...).

Chap 6 : Les fichiers

☞ On distingue généralement deux types d'accès :

Accès séquentiel :

- Pas de cellule vide.
- On accède à une cellule en se déplaçant, depuis la cellule de départ.
- On ne peut pas détruire une cellule.
- On peut ajouter une cellule à la fin.

Accès direct (RANDOM I/O)

- Cellule vide possible.
- On peut directement accéder à une cellule.
- On peut modifier n'importe quelle cellule.

Remarque : Le langage C ne distingue pas les fichiers à accès séquentiel des fichiers à accès direct.

☞ Il existe d'autre part deux façons de coder les informations stockées dans un fichier :

En binaire :

Fichier dit « binaire », les informations sont codées en brut sans aucune transformation. Ils ne sont pas éditables.

en ASCII :

Fichier dit « texte », les informations sont codées en ASCII. Ces fichiers sont éditables. Le dernier octet de ces fichiers est EOF.

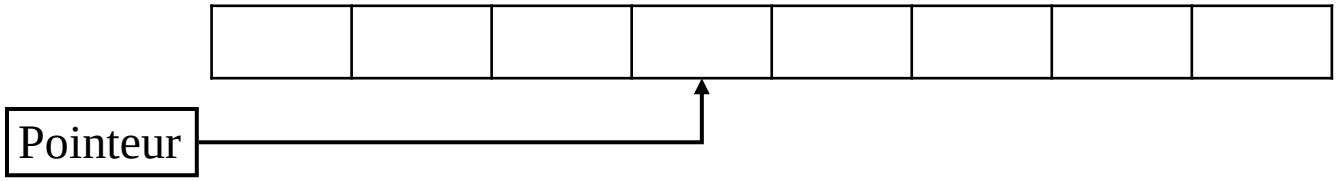
Chap 6 : Les fichiers

Manipulation des fichiers

Les opérations possibles avec les fichiers sont : Créer, Ouvrir, Fermer, Lire, Ecrire, Détruire, Renommer en utilisant des fonctions rangées dans STDIO.H

Déclaration

Consiste à définir un pointeur qui permet de fournir l'adresse d'une cellule donnée dans le fichier ouvert.



La déclaration d'un tel pointeur se fait par :

```
FILE   *pointeur  ;   /* majuscules obligatoires pour FILE */
```

Ouverture

Permet d'associer le pointeur déclaré de type FILE avec le fichier à manipuler et de préciser les possibilités d'accès (lecture, écriture ou lecture/écriture). L'ouverture se fait par :

```
PointeurFILE = fopen (nomFichier, mode) ;
```

Avec :

NomFichier : Nom du fichier figurant sur le disque, exemple: "a :\Etudiants.dat"

Mode : chaîne de caractères qui détermine le mode d'accès

Chap 6 : Les fichiers

Mode		Description
Fichier Texte	Fichier Binaire	
r	rb	lecture seule. Le fichier doit exister. Pointeur au début
w	wb	écriture seule (destruction de l'ancienne version si elle existe, création du fichier si non). Pointeur au début
a	ab	écriture d'un fichier existant, pas de création d'une nouvelle version. Pointeur à la fin du fichier.
r+	rb+	lecture/écriture d'un fichier existant (mise à jour), pas de création d'une nouvelle version. Pointeur au début
w+	wb+	lecture/écriture (destruction ancienne version si elle existe, sinon création du fichier). Pointeur au début.
a+	ab+	lecture/écriture d'un fichier existant (mise à jour), pas de création d'une nouvelle version. Pointeur à la fin du fichier.

Remarque : **fopen** renvoie l'adresse du flux s'il est créé si non elle renvoie **NULL**

Fermeture:

Permet de supprimer le flux associé au fichier ouvert après avoir achever les transferts de données survenus. La fermeture d'un fichier ouvert se fait par :

```
fclose(Pointeurfichier);
fcloseAll();              /* pour fermer tous les flux ouverts*/
```

Ecriture

Fichiers texte

Ecrire un caractère

```
int putc(char c, FILE *Pointeurfichier);
```

Ecrit la valeur de la variable caractère c à la position courante du pointeur , le pointeur avance d'une case mémoire. Retourne le caractère écrit sinon **EOF** en cas d'erreur.

Ecrire un entier

```
int putw(int n, FILE *Pointeurfichier);
```

Ecrit l'entier n, le pointeur avance du nombre de cases correspondant à la taille d'un entier. Retourne n si l'écriture s'est bien passée sinon retourne **EOF**.

Ecrire une chaîne de caractères

```
int fputs(char *S, FILE *Pointeurfichier);
```

Ecrit la chaîne de caractères S, le pointeur avance de la longueur de la chaîne. Retourne le dernier caractère écrit sinon retourne **EOF** en cas d'erreur.

Ecrire de données formatées

```
int fprintf(FILE *Pointeurfichier, char *format, liste d'expressions);
```

Ecrit des données formatées (caractère %c, entier %d, réel %f, chaîne %s), le pointeur avance du nombre d'octets écrits. Retourne le nombre d'octets écrits ou EOF en cas d'erreur.

Fichiers binaire

```
int fwrite(void *p,int taille_bloc,int n,FILE *Pointeurfichier);
```

Ecrit à partir de la position courante du pointeur Pointeurfichier n blocs de taille taille_bloc lus à partir de l'adresse p. Le pointeur Pointeurfichier avance d'autant d'octets écrits. Retourne le nombre de blocs écrits.

Lecture

Fichiers texte

Lire un caractère

```
int getc(FILE *fichier);
```

renvoie le caractère situé à la position pointée par Pointeurfichier ou EOF si erreur ou fin de fichier; le pointeur avance d'un octet.

Lire un entier

```
int getw(FILE *Pointeurfichier);
```

Renvoie l'entier situé à la position pointée par Pointeurfichier ou EOF en cas d'erreur ou fin du fichier; le pointeur avance de la taille d'un entier

Lire une chaîne de caractères

```
char *fgets(char *S,int n,FILE *Pointeurfichier);
```

Lit n-1 caractères à partir de la position du pointeur et les range dans S en ajoutant '\0'. Retourne NULL en cas d'erreur

Ecrire de données formatées

```
int fscanf(FILE *Pointeurfichier, char *format, liste d'adresses);
```

Lit des données depuis la position du pointeur PointeurFchier dans des variables dont les adresses sont indiquées. Les données lues sont convertie en fnction des format indiqués par format. La fonction retourne le nombre de champs lus.

Fichiers binaire

```
int fread(void *p,int taille_bloc,int n,FILE *Pointeurfichier);
```

lit à partir de la position courante du pointeur Pointeurfichier n blocs de taille taille_bloc et les place dans l'adresse p. Le pointeur Pointeurfichier avance d'autant d'octets lus. Retourne le nombre de blocs lus.

Positionnement dans un fichier – Accès direct

Au cours de la manipulation d'un fichier, il existe toujours une position courante du pointeur de lecture/écriture. Cette position représente le numéro de l'octet pointé. Elle est modifiée suite à chaque opération de lecture/écriture. Elle est incrémentée du nombre d'octets lus ou écrits.

Fonction fseek

```
int fseek(FILE *fichier,long offset,int Mode);
```

déplace le pointeur de offset cases à partir en fonction de la valeur de Mode.

Valeurs possibles pour Mode sont :

SEEK_SET = 0 -> à partir du début du fichier.

SEEK_CUR =1 -> à partir de la position courante du pointeur.

SEEK_END =2 -> en arrière, à partir de la fin du fichier

Fonction rewind

```
void rewind(FILE *fichier);
```

Repositionne le fichier à son début.

Fonction ftell

```
long ftell(FILE *fichier);
```

Retourne la position courante par rapport au début du fichier