Les Fichiers

Cours Algorithmique et Programmation C Avancées

Module Informatique I I

Plan

- GENERALITES
- Types d'accès:
- Codage de l'information

  (1- En binaire : 2- en ASCII)
- MANIPULATION DES FICHIERS
- Exercices d'application

Généralités

- Un fichier est un ensemble d'informations stockées sur une mémoire de masse (disque dur, disquette, bande magnétique, CD-ROM).
- En C, un fichier est une suite d'octets. Les informations contenues dans le fichier ne sont pas forcément de même type (un char, un int, une structure ...)
- Un **pointeur** fournit l'adresse d'une information quelconque.

- On distingue généralement deux types d'accès:

1- Accès séquentiel (possible sur tout support, mais seul possible sur bande magnétique):

Pas de cellule vide.

- On accède à une cellule quelconque en se déplaçant (via un pointeur), depuis la cellule de départ.
- On ne peut pas détruire une cellule.
- On peut par contre tronquer la fin du fichier.
- On peut ajouter une cellule à la fin.

2- Accès direct (RANDOM I/O) (Utilisé sur disques, disquettes, CD-ROM où l'accès séquentiel est possible aussi).

Cellule vide possible.

- **On peut directement accéder à une cellule.**
- **On peut modifier (voir détruire) n'importe quelle cellule.**
- **Il existe d'autre part deux façons de coder les informations stockées dans un fichier :**

1- En binaire :

- **Fichier dit « binaire », les informations sont codées telles que. Ce sont en général des fichiers de nombres. Ils ne sont pas listables.**

2- en ASCII :

- Fichier dit « texte », les informations sont codées en ASCII. Ces fichiers sont listables. Le dernier octet de ces fichiers est EOF (caractère ASCII spécifique).

MANIPULATION DES FICHIERS

- **Opérations possibles avec les fichiers: Créer - Ouvrir - Fermer - Lire - Ecrire - Détruire - Renommer. La plupart des fonctions permettant la manipulation des fichiers sont rangées dans la bibliothèque standard STDIO.H, certaines dans la bibliothèque IO.H pour le BORLAND C++.**
- **Le langage C ne distingue pas les fichiers à accès séquentiel des fichiers à accès direct, certaines fonctions de la bibliothèque livrée avec le compilateur permettent l'accès direct. Les fonctions standards sont des fonctions d'accès séquentiel.**

MANIPULATION DES FICHIERS

1 - Déclaration:

1 - Déclaration: FILE *fichier; /*majuscules obligatoires pour FILE */

- **On définit un pointeur. Il s'agit du pointeur représenté sur la figure du début de chapitre. Ce pointeur fournit l'adresse d'une cellule donnée.**
- **La déclaration des fichiers doit figurer AVANT la déclaration des autres variables.**

MANIPULATION DES FICHIERS
2 - Ouverture:

2 - Ouverture: FILE *fopen(char *nom, char *mode);

On passe donc 2 chaînes de caractères

- **nom: celui figurant sur le disque, exemple: « a :\toto.dat »**

MANIPULATION DES FICHIERS

mode (pour les fichiers TEXTES) :

- **« r » lecture seule**
- **« w » écriture seule (destruction de l'ancienne version si elle existe)**
- **« w+ » lecture/écriture (destruction ancienne version si elle existe)**
- **« r+ » lecture/écriture d'un fichier existant (mise à jour), pas de création d'une nouvelle version.**
- **« a+ » lecture/écriture d'un fichier existant (mise à jour), pas de création d'une nouvelle version, le pointeur est positionné à la fin du fichier.**

MANIPULATION DES FICHIERS

mode (pour les fichiers BINAIRES) :

- « rb » lecture seule
- « wb » écriture seule (destruction de l'ancienne version si elle existe)
- « wb+ »lecture/écriture (destruction ancienne version si elle existe)
- « rb+ » lecture/écriture d'un fichier existant (mise à jour), pas de création d'une nouvelle version.
- « ab+ »lecture/écriture d'un fichier existant (mise à jour), pas de création d'une nouvelle version,
- le pointeur est positionné à la fin du fichier.

MANIPULATION DES FICHIERS

A l'ouverture, le pointeur est positionné au début du fichier (sauf « a+ » et « ab+ »)

Exemple : **FILE *fichier ;**

fichier = fopen(« a :\toto.dat », « rb ») ;

MANIPULATION DES FICHIERS

3 - Fermeture:

3 - Fermeture: **int fclose(FILE *fichier);**

Retourne 0 si la fermeture s'est bien passée, EOF en cas d'erreur.

Il faut toujours fermer un fichier à la fin d'une session. mode (pour les fichiers TEXTE) :

- Exemple : **FILE *fichier ;**

```
fichier = fopen(« a :\toto.dat », « rb ») ;
/* Ici instructions de traitement */
 fclose(fichier) ;
```

MANIPULATION DES FICHIERS

4 – Destruction 5 - Renommer: : 6 - Positionnement

4 - Destruction: int remove(char *nom);

Retourne 0 si la fermeture s'est bien passée.

Exemple : **remove(« a :\toto.dat ») ;**

5 - Renommer: int rename(char *oldname, char *newname);

Retourne 0 si la fermeture s'est bien passée.

6 - Positionnement du pointeur au début du fichier:

void rewind(FILE *fichier);

MANIPULATION DES FICHIERS
7- Ecriture dans le fichier:

- **int putc(char c, FILE *fichier);**

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

- Exemple : **putc('A', fichier) ;**

MANIPULATION DES FICHIERS
7- Ecriture dans le fichier:

- **int putw(int n, FILE *fichier);**

Idem, n de type int, le pointeur avance du nombre de cases correspondant à la taille d'un entier (4 cases en C standard).

Retourne n si l'écriture s'est bien passée.

- **int fputs(char *chaîne, FILE *fichier);** idem avec une chaîne de caractères, le pointeur avance de la longueur de la chaine ('\0' n'est pas rangé dans le fichier).

Retourne EOF en cas d'erreur.

- Exemple : **fputs(« BONJOUR ! », fichier) ;**

MANIPULATION DES FICHIERS
7- Ecriture dans le fichier:

- **int fwrite(void *p,int taille_bloc,int nb_bloc, FILE *fichier);** p de type pointeur, écrit à partir de la position courante du pointeur fichier nb_bloc X taille_bloc octets lus à partir de l'adresse p. Le pointeur fichier avance d'autant.
- Le pointeur p est vu comme une adresse, son type est sans importance.

  Retourne le nombre de blocs écrits.
- Exemple: taille_bloc=4 (taille d'un entier en C), nb_bloc=3, écriture de 3 octets.

```
int tab[10] ;
fwrite(tab,4,3,fichier) ;
```

MANIPULATION DES FICHIERS
7- Ecriture dans le fichier:

- **int fprintf(FILE *fichier, char *format, liste d'expressions);** réservée plutôt aux fichiers ASCII.

- Retourne EOF en cas d'erreur.
  Exemples: **fprintf(fichier,"%s","il fait beau");**
  fprintf(fichier,%d,n);
  fprintf(fichier,"%s%d","il fait beau",n);

- Le pointeur avance d'autant.

MANIPULATION DES FICHIERS

8 - Lecture du fichier:

- int getc(FILE *fichier); lit 1 caractère, mais retourne un entier n; retourne EOF si erreur ou fin de fichier;
  le pointeur avance d'une case.

Exemple: char c ;

c = (char)getc(fichier) ;

- int getw(FILE *fichier); idem avec un entier; le pointeur avance de la taille d'un entier.

Exemple: int n ;

n = getw(fichier) ;

- char *fgets(char *chaine,int n,FILE *fichier); lit n-1 caractères à partir de la position du pointeur et les range dans chaine en ajoutant '\0'.

MANIPULATION DES FICHIERS

8 - Lecture du fichier:

int fread(void *p,int taille_bloc,int nb_bloc,FILE *fichier);
analogue à fwrite en lecture.
Retourne le nombre de blocs lus, et 0 à la fin du fichier.

int fscanf(FILE *fichier, char *format, liste d'adresses);
analogue à fprintf en lecture.

MANIPULATION DES FICHIERS
9 - Gestion des erreurs:

- **fopen** retourne le pointeur NULL si erreur
(Exemple: impossibilité d'ouvrir le fichier).

- **fgets** retourne le pointeur NULL en cas d'erreur ou si la fin du fichier est atteinte.

- la fonction **int feof(FILE *fichier)** retourne 0 tant que la fin du fichier n'est pas atteinte.

- la fonction **int ferror(FILE *fichier)** retourne 1 si une erreur est apparue lors d'une manipulation de fichier, 0 dans le cas contraire.

MANIPULATION DES FICHIERS

10 - Fonction particulière aux fichiers à acces direct:

- **int fseek(FILE *fichier,int offset,int direction)** déplace le pointeur de offset cases à partir de direction.

  Valeurs possibles pour direction:

  0 -> à partir du début du fichier.

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

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

- Retourne 0 si le pointeur a pu être déplacé;

Exercices d'application

Série d’exercices