

LES FICHIERS

Un fichier est un ensemble d'informations stockées sur une mémoire de masse (disque dur, disquette, CD-ROM).

En C, un fichier est une suite d'octets. Les informations contenues dans le fichier ne sont pas forcément de même type (char, int, structure ...)

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**,

1 - Déclaration:

FILE *f; /* majuscules obligatoires pour FILE */

On définit un pointeur. Ce pointeur fournit l'adresse d'une cellule donnée.

2 - Ouverture:

f=fopen(char *nom, char *mode);

On passe donc 2 chaînes de caractères

nom: celui figurant sur le disque, exemple: « C:\toto.dat »

mode: c'est le mode d'accès.

"r"	ouverture d'un fichier texte en lecture
"w"	ouverture d'un fichier texte en écriture
"a"	ouverture d'un fichier texte en écriture à la fin

Remarque :

Ces modes d'accès ont pour particularités :

- Si le mode contient la lettre r, le fichier doit exister.
- Si le mode contient la lettre w, le fichier peut ne pas exister. Dans ce cas, il sera créé. Si le fichier existe déjà, son ancien contenu sera perdu.
- Si le mode contient la lettre a, le fichier peut ne pas exister. Dans ce cas, il sera créé. Si le fichier existe déjà, les nouvelles données seront ajoutées à la fin du fichier précédent.
- A l'ouverture, le pointeur est positionné au début du fichier

Exemple : **FILE *f ;**

f = fopen(« a :\toto.dat », « rb ») ;

3 - Fermeture:

int fclose(FILE *f);

Retourne 0 si la fermeture s'est bien passée.

Il faut toujours fermer un fichier à la fin d'une session.

Exemple : **FILE *f;**

f = fopen(C :\toto.dat , rb) ;

/* Ici instructions de traitement */

fclose(f) ;

4 - Destruction:

int remove(char *nom);

Retourne 0 si la fermeture s'est bien passée.

Exemple : **remove("C:\toto.dat") ;**

5 - Renommer:

int rename(char *oldname, char *newname);

Retourne 0 si la fermeture s'est bien passée.

6- Ecriture dans le fichier:

a- int putc(char c, FILE *f);

Ecrit la valeur de c à la position courante du pointeur, le pointeur avance d'une case mémoire.

Exemple : **putc('A', f) ;**

b- int fputs(char *chaîne, FILE *f);

idem avec une chaîne de caractères, le pointeur avance de la longueur de la chaîne ('\0' n'est pas rangé dans le fichier).

Exemple : **fputs("BONJOUR", f) ;**

c- int fwrite(*p,int taille_bloc,int nb_bloc,FILE *f);

p de type pointeur, écrit à partir de la position courante du pointeur f **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.

Exemple: taille_bloc=4 (taille d'un entier en C), nb_bloc=3.

int tab[10] ;

fwrite(tab,4,3,f) ;

d- int fprintf(FILE *f, char *format, liste d'expressions);

réservée plutôt aux fichiers ASCII.

Exemples: **fprintf(f,"%s","il fait beau");**

fprintf(f,%d,n);

fprintf(f,"%s%d","il fait beau",n);

Le pointeur avance d'autant.

7 - Lecture du fichier:

int getc(FILE *f);

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(f) ;

char *fgets(char *chaine,int n,FILE *f);

lit n-1 caractères à partir de la position du pointeur et les range dans chaine en ajoutant '\0'.

int fread(void *p,int taille_bloc,int nb_bloc,FILE *f);

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.

8 - 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 *f)** retourne 0 tant que la fin du fichier n'est pas atteinte.

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