Les Fichiers

Langage C

Sommaire

- Ouverture d’un fichier
- Fermeture d’un fichier
- Ecrire dans un fichier
- Lire dans un fichier
- Se déplacer dans un fichier
- Renommer / Supprimer un fichier

Ouverture d'un fichier

- **FILE* fopen(char *nomDuFichier, char *modeDouverture);**
- Modes d'ouverture:
  - **r:** lecture
  - **w:** écriture
  - **a:** ajout
  - **r+:** lecture + écriture
  - **w+:** écriture + lecture
  - **a+:** lecture + écriture à la fin

Existe également en mode binaire (wb, rb…)

- **Important:** Toujours tester l'ouverture du fichier.

Fermeture d’un fichier

- **int fclose(FILE* pointeurSurFichier);**

- Valeurs de retour:
  - 0: si la fermeture a fonctionné
  - EOF: si la fermeture a échoué (define situé dans stdio.h)

- Test de fermeture optionnel car généralement la fermeture se passe toujours bien.

Ecriture dans un fichier

- Il existe 4 fonctions:
  - **fputc**: écrit un seul caractère dans le fichier
  - **fputs**: écrit une chaîne dans le fichier
  - **fprintf**: écrit une chaîne « formatée » dans le fichier
  - **fwrite**: écrit des données dans un fichier binaire

fputc

- **Prototype: int fputc(int caractère, FILE* pointeurSurFichier);**
- Valeurs de retour:
  - Si l'écriture a échoué:
    - EOF
  - Sinon
    - Autre valeur

```
 4  int main(void)
 5  {
 6      FILE* fichier = NULL;
 7
 8      fichier = fopen("test.txt", "w");
 9
10      if(fichier != NULL)
11      {
12          fputc('A', fichier);
13          fclose(fichier);
14      }
15
16      return 0;
17  }
18
```

fputs

- **Prototype: int fputs(const char* chaine, FILE* pointeurSurFichier);**
- Valeurs de retour:
  - Si l'écriture a échoué:
    - EOF
  - Sinon
    - Valeur positive

```
 4    int main(void)
 5    {
 6        FILE* fichier = NULL;
 7
 8        fichier = fopen("test.txt", "w");
 9
10        if(fichier != NULL)
11        {
12            fputs("Hello World!", fichier);
13            fclose(fichier);
14        }
15
16        return 0;
17    }
18
```

fprintf

- **Prototype: int fprintf(FILE* pointeurSurFichier, const char* chaine, …);**
- Valeurs de retour:
  - Si l'écriture a échoué:
    - Valeur négative
  - Sinon
    - Le nombre total de caractères écrits

```
 4  int main(void)
 5  {
 6      FILE* fichier = NULL;
 7
 8      fichier = fopen("test.txt", "w");
 9
10      if(fichier != NULL)
11      {
12          int nombre = 10;
13
14          fprintf(fichier, "Le nombre vaut %d !", nombre);
15          fclose(fichier);
16      }
17
18      return 0;
19  }
20
```

fwrite

- **Prototype: int fwrite(void* donnees, int taille, int nbElementsFILE* pointeurSurFichier);**
- A utiliser pour les fichiers binaires.
- Valeurs de retour:
  - Renvoie le nombre d'éléments écrits

```c
#define NB_ELTS 5
void main (void)
{
    FILE* fic ;
    short int tablo[NB_ELTS] = {1,2,3,4,5 } ;
    /* Ouverture du fichier (en écriture binaire) : */
    fic = fopen( "exemple.dat", "wb") ;
    if ( fic==NULL )
    {
        printf("Ouverture du fichier impossible !");
        exit(0);
    }
    /* Ecriture dans le fichier (ici, deux fois la même donnée, de deux façons différentes) : */
    /* Voici 2 façons différentes de stocker un tableau (la 1ère est plus claire) : */
    fwrite ( tablo, sizeof(short int), NB_ELTS, fic );
    /* on stocke NB_ELTS éléments de taille fournie par sizeof */
    fwrite ( tablo, 1, sizeof(tablo), fic );
    /* on stocke un nombre d'octets égal à sizeof(tablo) */
    /* Fermeture du fichier : */
    fclose( fic ) ;
}
```

Lire dans un fichier

- Il existe 4 fonctions:
  - **fgetc**: lit un caractère
  - **fgets**: lit une chaîne
  - **fscanf**: lit une chaîne formatée
  - **fread**: lit des données d'un fichier binaire

fgetc

- **Prototype: int fgetc(FILE* pointeurSurFichier);**
- Valeurs de retour:
  - Si la lecture a échoué:
    - EOF
  - Sinon
    - Le caractère qui a été lu

```
 4  int main(void)
 5  {
 6      FILE* fichier = NULL;
 7      int caractere = 0;
 8
 9      fichier = fopen("test.txt", "w");
10
11      if(fichier != NULL)
12      {
13          caractere = fgetc(fichier);
14          printf("%c", caractere);
15
16          fclose(fichier);
17      }
18
19      return 0;
20  }
```

fgets

- **Prototype: char* fgets(char* chaine, int nbCaracteres,FILE* pointeurSurFichier);**
- Valeurs de retour:
  - Si la lecture a échoué ou qu'aucun caractère n'a pu être lu (EOF):
    - NULL
  - Sinon
    - La même valeur que le paramètre « chaine »

```
4   #define TAILLE_MAX 500
5
6   int main(void)
7   {
8       FILE* fichier = NULL;
9       char chaine[TAILLE_MAX] = "";
10
11      fichier = fopen("test.txt", "w");
12
13      if(fichier != NULL)
14      {
15          fgets(chaine, TAILLE_MAX, fichier);
16          printf("%s", chaine);
17
18          fclose(fichier);
19      }
20
21      return 0;
22  }
```

fscanf

- **Prototype: int fscanf(FILE* pointeurSurFichier, const char* format, …);**
- Valeurs de retour:
  - Si la lecture a échoué:
    - EOF
  - Sinon
    - Le nombre de paramètres lus

```
 4  int main(void)
 5  {
 6      FILE* fichier = NULL;
 7      int nombres[3] = {0};
 8
 9      fichier = fopen("test.txt", "w");
10
11      if(fichier != NULL)
12      {
13          fscanf(fichier, "%d %d %d !", &nombres[0], &nombres[1], &nombres[2]);
14          printf("Les nombres sont: %d %d %d", nombres[0], nombres[0], nombres[0]);
15
16          fclose(fichier);
17      }
18
19      return 0;
20  }
```

fread

- **Prototype: int fread(void* donnees, int taille, int nbElementsFILE* pointeurSurFichier);**
- A utiliser pour les fichiers binaires.
- Valeurs de retour:
  - Renvoie le nombre d'éléments lus

```c
#define TAILLE_BUF 4 /* valeur quelconque (en général, beaucoup plus grande) */
void main (void)
{
    FILE* fic ;
    short int buffer[TAILLE_BUF]; /* ce tableau mémorisera les valeurs lues dans le fichier */
    short int i, nb_val_lues = TAILLE_BUF ;
    /* Ouverture du fichier (en lecture binaire) : */
    fic = fopen( "exemple.dat", "rb") ;
    if ( fic==NULL )
    {
        printf("Ouverture du fichier impossible !");
        exit(0);
    }
    /* Lecture dans le fichier : */
    printf("\n Liste des valeurs lues : \n");
    /*Remplissage du buffer et traitement, autant de fois que nécessaire jusqu'à la fin fichier : */
    while ( nb_val_lues == TAILLE_BUF ) /* vrai tant que fin du fichier non atteinte */
    {
        nb_val_lues = fread( buffer, sizeof(short int), TAILLE_BUF, fic);
        /* Traitement des valeurs stockées dans le buffer (ici, un simple affichage) : */
        for (i=0; i<nb_val_lues; i++) printf( "%hd", buffer[i] );
    }
    /* Fermeture du fichier : */
    fclose( fic ) ;
}
```

Se déplacer dans un fichier

- Trois fonctions à connaître:
  - **ftell**: indique à quelle position nous sommes actuellement
  - **fseek**: positionne le curseur à un endroit précis dans le fichier
  - **rewind**: remet le curseur au début du fichier

Ftell

- Renvoie la position actuelle du curseur dans un « long »
- **Prototype: long ftell(FILE* pointeurSurFichier)v**

Rewind

- Equivalente au **fseek** pour nous renvoyer à la position 0 du fichier
- **Prototype: void rewind(FILE* pointeurSurFichier)**

fseek

- **Prototype: int fseek(FILE* pointeurSurFichier, long deplacement, int origine)**

- Déplacement: peut-être positif, nul ou négatif
- Origine: Peut contenir 3 valeurs:
  - **SEEK_SET**: indique le début du fichier
  - **SEEK_CUR**: indique la position actuelle du curseur
  - **SEEK_END**: indique la fin du fichier

- Exemples:
  - fseek(fichier, 2, SEEK_SET);
  - fseek(fichier, -6, SEEK_CUR);
  - fseek(fichier, 0, SEEK_END);

Renommer / supprimer un fichier

- **rename:** *permet de renommer un fichier*

  int rename(const char* ancienNom, const char* nouveauNom);

- **remove:** *permet de supprimer un fichier (attention, aucune confirmation n'est demandée et le fichier sera définitivement supprimé)*

  Int remove(const char* fichierASupprimer);