Fichiers binaires

Un fichier binaire contient du code binaire. On ne peut pas visualiser son contenu avec un éditeur de texte. Lorsqu'une variable est écrite dans un fichier binaire, on écrit directement la valeur exacte de la variable, telle qu'elle est codée en binaire en mémoire. Cette manière de stocker les données est plus précise et plus compacte pour coder des nombres. Les fonctions de lecture et d'écriture dans un fichier binaire sont fread et fwrite qui lisent et écrivent des blocs de données sous forme binaire.

Ouverture et fermeture d'un fichier binaire :

On doit déclarer un pointeur de fichier et l'ouvrir avec la fonction fopen.

```
FILE *f ;
f = fopen(''fiche.txt'',''rb'') ;
```

Après avoir utilisé un fichier, il faut le refermer en utilisant fclose. La fonction fclose prend en paramètre le pointeur de fichier et ferme le fichier.

Lecture dans un fichier binaire :

Pour lire dans un fichier binaire, on lit en général dans le fichier les éléments d'un tableau. Chaque élément du tableau est appelé un bloc. Chaque bloc possède une taille en octets. Par exemple, un char correspond à un octet, un float correspond à 4 octets, etc.
La fonction sizeof donne la taille de chaque type. Par exemple, sizeof(char) vaut 1, et sizeof(float) vaut 4.
La fonction de lecture fread prend en paramètre le tableau, la taille de chaque bloc, le nombre de blocs à lire (nombre d'éléments du tableau), et le pointeur de fichier.
On peut lire une variable x avec la fonction fread. Il suffit pour cela de mettre l'adresse &x de la variable à la place du tableau et de mettre le nombre de blocs égal à 1 ; les données sont alors transférés dans la variable.
La fonction fread retourne le nombre d'éléments effectivement lus. Si ce nombre est inférieur au nombre effectivement demandé, soit il s'est produit une erreur de lecture, soit la fin du fichier a été atteinte.

Ecriture dans un fichier binaire :

Pour écrire dans un fichier binaire, on utilise la fonction fwrite qui transfère des données de la mémoire centrale vers la un fichier binaire.

Comme la fonction fread, la fonction fwrite prend en paramètre le tableau, la taille de chaque bloc, le nombre de blocs à écrire et le pointeur de fichier.

La fonction fwrite retourne le nombre d'éléments effectivement écrits. Si ce nombre est inférieur au nombre effectivement demandé, il s'est produit une erreur d'écriture (fichier non ouvert, disque plein …).

Exemple :

Ecrire un programme qui lit un tableau de nombres réels et les sauvegarde dans un fichier binaire. Le format du fichier se présente de la façon suivante : le nombre d'éléments en aucun cas ne dépassant 100 réels, puis les éléments à la suite dans le fichier.

```
#include <stdio.h>
#include <stdlib.h>
void lecturetab(int *n, float t[100])
{int i ;
 puts(''Entrer le nombre d'éléments'') ;
scanf(''%d'', n) ;
for (i=0 ;i<*n ;i++)
  scanf(''%f'',&t[i]) ;
}
void sauvegarde(float t[100], int n, char  nom[30])
{ FILE *f ;
 if ((f=fopen(nom, ''wb''))==NULL)
   {puts(''permission refusée ou répertoire inexistant'') ;
    exit(1) ;
    }
   fwrite(&n, sizeof(int), 1, f) ;
   if (fwrite(t, sizeof(float), n, f) != n)
     puts(''Erreur d'écriture dans le fichier'') ;
   fclose(f) ;
  }
 main()
{int n ;
 float t[100] ;
 lecturetab(&n, t) ;
 sauvegarde(t, n, ''fichier'') ;
}
```

Accès directe dans un fichier binaire :

La fonction fseek permet de se positionner dans un fichier, en modifiant la position courante pour pouvoir lire ou écrire à l'endroit souhaité. Lorsqu'on écrit sur un emplacement, la donnée qui existait éventuellement à cet emplacement est effacée et remplacée par la donnée écrite.

int fseek(FILE *f, long offset, int origine) ;

la fonction modifie la position du pointeur f d'un nombre d'octets égal à offset à partir de l'origine.

L'origine peut être :

SEEK_SET : on se positionne par rapport au début du fichier ;

SEEK_END : on se positionne par rapport à la fin du fichier ;

SEEK_CUR : on se positionne par rapport à la position courante actuelle (position avant l'appel de fseek).

Exemple :

Soit un fichier d'entiers. La fonction prend un entier i et permet de modifier le (i+1) éme entier du fichier.

```
void changenb(FILE *f, int i)
{int n, nouv ;
 fseek(f, i*sizeof(int), SEEK_SET) ;
fread(&n, sizeof(int), 1, f) ;
printf(''l'ancien entier vaut %d\n'', n) ;
puts(''entrer la nouvelle valeur '') ;
scanf(''%d'', &n) ;
fseek(f, -sizeof(int), SEEK_CUR) ;
fwrite(&nouv, sizeof(int), 1, f) ;
}
```

Exemple :

Création séquentielle d'un fichier d'entiers (différent de zéro) avec prise en compte des erreurs.

```
#include <stdio.h>
#include <stdlib.h>
main()
{char nom[21] ;
 int n, ret ;
 FILE * sortie ;
 printf(''nom du fichier à créer'') ;
 scanf(''%20s'', nom) ;
 sortie = fopen(nom, ''wb'') ;
 if (sortie ==NULL)
   {printf(''ERREUR'') ;
    exit(1) ;
   }
 do
 {printf(''donner un entier'') ;
  scanf(''%d'', &n) ;
  if (n)
    {ret = fwrite(&n, sizeof(int), 1, sortie) ;
     if ( ! ret)
        printf(''ERREUR D'ECRITURE'') ;
     }
   }
   while (n && ret) ;
   fclose(sortie) ;
  }
```

Exemple :

Lecture séquentielle d'un fichier d'entiers avec prise en compte des erreurs.

```
#include <stdio.h>
#include <stdlib.h>
main()
{char nom[21] ;
 int n;
 FILE * entree ;
 printf(''nom du fichier à lister'') ;
 scanf (''%20s'', nom) ;
 entree = fopen(nom, ''rb'') ;
 if (entree == NULL)
   {printf(''ERREUR D'OUVERTURE'') ;
    exit(1) ;
    }
 while ((fread(&n, sizeof(int), 1, entree)) && !feof(entree))
     printf(''%d\n'', n) ;
  fclose(entree) ;
  }
```