Les fichiers

DAKKAR Borhen-eddine

Lycée le Corbusier

BTS SN-IR

Table des matières

Table des matières

Les fichiers

- Les données des programmes que nous avons vus jusqu'à présent ont été soit affectées en interne dans le programme, soit entrées de manière interactive pendant l'exécution du programme.
- En C il est possible de stocker des données en dehors d'un programme. Ces données externes permet à un programme d'utiliser les données sans avoir à les recréer.
- Tout ensemble de données stockées sous un nom commun sur un un support de stockage est appelé un fichier de données (data file).
- Nous allons voir les instructions C nécessaires pour créer des fichiers, pour écrire et pour lire des données.

Création et utilisation de fichiers de données

- Un fichier de données est physiquement stocké par un ordinateur en utilisant un nom de fichier unique.
- La plupart des ordinateurs exigent un nom de fichier qui se compose de caractères pour le nom suivis et d'une extension.
- En C, un fichier est toujours référencé par un nom de variable qui doit être déclaré par la forme suivante :

```
FILE * nom_de_fichier;
```

- L'étoile * est requise (pointeur). **nom_de_fichier** est choisi par le programmeur et peut être n'importe quel nom de variable valide.
- La correspondance entre le nom de fichier de l'ordinateur et le nom de la variable est crée à l'aide de la fonction **fopen ()**.

Création et utilisation de fichiers de données

- Deux arguments sont nécessaires pour la fonction **fopen ()** :
  1. Le premier argument est le nom du fichier à ouvrir.
  2. Le deuxième argument est le mode dans lequel le fichier doit être utilisé. Les modes les plus utilisés sont "**r**", "**w**" et "**a**", qui représentent respectivement la lecture, l'écriture ou l'ajout à un fichier.
- L'instruction :

```
fichier1 = fopen ("Cours1.dat", "w");
```

  ouvre le fichier **Cours1.dat** et l'assigne a la varible **fichier1**. Le nom **fichier1** est un nom sélectionné par le programmeur et déclaré avec **FILE**.
- Un fichier ouvert en écriture crée un nouveau fichier et le rend utilisable dans la fonction ouvrant le fichier. Si un fichier existe avec le même nom, l'ancien contenu de ce fichier sera écrasé.

Ajout de données à un fichier

- L'ouvertre pour l'ajout rend un fichier disponible pour ajouter des données à la fin d'un fichier.
- Si le fichier ouvert pour l'ajout n'existe pas, un nouveau fichier avec le nom désigné est créé et mis à disposition pour recevoir des données.
- L'instruction :

```
fichier2 = fopen("fich_ajout.dat","a");
```

  ouvre un fichier nommé **fich_ajout.dat** et le rend disponible pour ajouter des données.
- La seule différence entre un fichier ouvert en écriture et un fichier ouvert en mode ajout est l'endroit où les données seront placées dans le fichier.
- En mode écriture, les données sont écrites à partir du début du fichier, en mode ajout les données sont écrites à partir de la fin du fichier. Pour un nouveau fichier, les deux modes sont identiques.

Lecture de données d'un fichier

- Un fichier ouvert en lecture récupère le contenu d'un fichier existant et le met à disposition en entrée du programme.

- L'instruction :

```
fichier3 = fopen ("test.dat", "r");
```

  ouvre le fichier nommé **test.dat** et lit les données contenu dans ce fichier.

- L'appel de la fonction **fopen ()** peut être effectué n'importe où dans une fonction. Le prototype de cet fonction est contenu dans le fichier entête **stdio.h**.

Lecture de données d'un fichier

- Lorsqu'un appel à **fopen ()** est rencontré, l'ordinateur vérifie l'existance du fichier sur le système.
- Si un fichier portant le nom indiqué existe, le le fichier est ouvert.
- Si le fichier n'existe pas et que le fichier a été ouvert en mode écriture ou en mode ajout ("w", "a"), un fichier vierge portant le nom indiqué sera créé.
- Si un fichier ouvert en lecture n'existe pas, la fonction **fopen ()** renvoie le constante système **NULL**.

Exemple 1 : ouverture en lecture

```c
#include <stdio.h>
#include <stdlib.h>
main ( )
{
FILE *in_file;
in_file = fopen ("test.dat", "r") ; /* open the file */
if (in_file == NULL) /* if fopen() returns NULL to in_file */
{
printf("\nLe fichier ne peut pas etre ouvert.");
printf("\nMerci de verifier si ce fichier existe.");
}
else
printf("\nLe ficher a ete ouvert avec succes.");
}
```

Quand le programme s'exécute sur un système qui contient le fichier **test.dat**, il affiche le message :
Le fichier a été ouvert avec succès pour la lecture.
Si le fichier n'existe pas, il affiche le message :
Le fichier ne peut pas être ouvert.
de verifier si ce fichier existe.

Exemple 2 : écrire dans un fichier

- Si un fichier est ouvert en mode écriture, les données peuvent être écrites en utilisant presque fonctions d'écriture de données sur un écran d'affichage.
- Les fonctions permettant d'écrire dans un fichier sont les suivantes :
  1. fputc(c,nom_fichier) : écrit un seul caractère.
  2. fputs(string,nom_fichier) : écrit une chaine de caractères.
  3. fprintf(nom_fichier, "format",args) : écrit les valeurs des arguments dans le fichier selon le format.
- La fonction **fprintf ()** est utilisée de la même manière que la fonction **printf ()**, avec l'ajout du nom de fichier comme argument. Le nom du fichier dirige la sortie vers un fichier spécifique plutôt que vers le dispositif d'affichage (écran).

Exemple 2 : écrire dans un fichier

```c
#include <stdio.h>
#include <stdlib.h>

main( )
{
FILE *fich_sortie; /* FILE declaration */
float poids =165.0 , pente = 7.5, fact = 2.0625;
fich_sortie = fopen("test1.txt","w");
fprintf(fich_sortie,"%f",poids) ;
fprintf (fich_sortie, "\n%f     %f", pente, fact);
}
```

- L'écriture dans un fichier est éxactement la même chose que l'écriture sur un périphérique de sortie standard, à l'exception de la désignation explicite du nom du fichier et l'utilisation de **fprintf ()** à la place de **printf ()**. Cela signifie que toutes les techniques que nous avons appris pour créer des affichages de sortie s'appliquent au fichier.

Exemple 3 : écrire des données tableau

L'exemple suivant illustre le stockage de données à partir d'un tableau dans un fichier :

```c
#include <stdio.h>
#include <stdlib.h>
main( )
{
FILE *nouveau_fich;
int i;
float result[5] = {16.25, 17.0, 15.75, 18.0, 19.5};
nouveau_fich = fopen("exper.dat","w"); /* Ouvre le fichier */
for (i = 0; i < 5; ++i)
fprintf(nouveau_fich,"\n%ld %9.6f", i, result[i]);
fclose(nouveau_fich);
}
```

- Le programme comprend un appel à la fonction **fclose ()**. Cette fonction est utilisé pour rompre le lien établi par l'appel de la fonction **fopen ()** et la varible **nouveau_fichier** dans le programme.

Exemple 3 : écrire des données tableau

- Tous les ordinateurs ont une limite sur le nombre maximum de fichiers qui peuvent être ouvert en même temps, donc la fermeture des fichiers est utile.

- Lorsqu'un fichier est fermé avec **fclose()**, un marqueur de fin de fichier spécial **EOF** (End-Of-file) est automatiquement placé par le système d'exploitation comme dernier caractère du fichier.

- Le caractère **EOF** a un code numérique unique qui n'a pas de représentation équivalente parmi les caractères imprimables. Il garantit que le caractère **EOF** ne peut jamais être confondu avec un caractère valide contenu dans le fichier.

- Le caractère **EOF** peut être utilisé comme un sentinelle lors de la lecture des données d'un fichier.

Exemple 4 : Lire les données d'un fichier

- La lecture de données à partir d'un fichier est presque identique à la lecture de données à partir du clavier.
- Les fonctions premettent la lecture à partir d'un fichier sont les suivantes :
  1. fgetc(nom_fichier) : lit un caractère d'un fichier.
  2. fgets(nom_chaine_ caractères,n, nom_fichier) : lit $n-1$ caractères et les stocke dans **nom_chaine_ caractères**. Par exemple :fgets (& line [0], 81, in_fichier);
  3. fscanf(nom_fichier,"format",args) : Lit les valeurs des arguments à partir du fichier selon le format donné.
- Toutes ces fonctions détectent correctement le marqueur de fin de fichier **EOF**.
- Les fonctions **fgetc ()** et **fscanf ()**, renvoient la constante nommée EOF lorsque le marqueur est détecté.
- La fonction **fgets ()** renvoie un **NULL** () lorsqu'elle détecte la fin d'un fichier.

Exemple 4 : Lire les données d'un fichier

L'exemple suivant utilise la fonction **fscanf ()** pour lire cinq lignes du fichier **exper.dat** écrit dans l'exemple 3. A chaque lecture du fichier, un entier et une valeur réelle sont saisis dans le programme et affichés.

```
#include <stdio.h>
main( )
{
FILE *nouveau_fich;
int i, n;
float val;
nouveau_fich = fopen ("exper.dat" , "r") ;
for (i = 1; i <=5; ++i)
{
fscanf(nouveau_fich,"%d %f", &n, &val);
printf("\n%d %f",n,val);
}
fclose(nouveau_fich);
}
```

Exemple 4 : Lire les données d'un fichier

Le programme suivant illustre la lecture du fichier **exper.dat** en utilisant le marqueur **EOF** dans une boucle **while**, qui est retourné par **fscanf ()** lorsque la fin de fichier est rencontrée.

```
#include <stdio.h>
main( )
{
FILE *nouveau_fich;
int i, n;
float val;
nouveau_fich = fopen ("exper.dat" , "r");
while (fscanf(nouveau_fich,"%d %f", &n, &val) != EOF)
{
printf("\n%d %f",n,val);
}
fclose(nouveau_fich);
}
```

Table des matières

Accès aléatoire aux fichiers

- L'organisation des fichiers fait référence à la manière dont les données sont stockées dans un fichier.
- Une organisation séquentielle signifie que les caractères du fichier sont stockés les uns après les autres.
- En C, il est possible d'accéder à un fichier de manière aléatoire.
- Les fonctions de la bibliothèque **rewind ()**, **fseek ()** et **ftell ()** peuvent être utilisé pour fournir un accès aléatoire à un fichier.
- En accès aléatoire, n'importe quel caractère du fichier peut être lu immédiatement, sans avoir à lire au préalable tous les caractères stockés avant cela.

Les fonctions d'accès aléatoire

- La fonction rewind () réinitialise la position actuelle au début du fichier, elle requiert le nom du fichier comme seul argument. Par exemple, l'instruction :

```
rewind(fichier);
```

réinitialise le fichier afin que le prochain caractère accédé soit le premier caractère du fichier.

- La fonction **fseek ()** permet de se déplacer à n'importe quelle position dans le fichier. Afin de bien utiliser cette fonction, il faut savoir comment les les données sont référencées dans le fichier.
- Chaque caractère dans un fichier est localisé avec une position, le premier caractère du fichier est situé à la position 0, le caractère suivant à la position 1, et ainsi de suite.
- La position d'un caractère est également référencé avec son décalage par rapport au début du fichier.
- La fonction **fseek ()** nécessite trois arguments: le nom du fichier; le décalage et la position à partir de laquelle le décalage doit être calculé.

```
fseek(fichier, décalage, origine)
```

Les fonctions d'accès aléatoire

- Les valeurs de l'argument **origine** peuvent être 0, 1 ou 2.
  1. Une origine à 0 signifie que le décalage est relatif au début du fichier.
  2. Une origine de 1 signifie que le décalage est relatif à la position actuelle dans le fichier.
  3. Une origine de 2 signifie que le décalage est relatif à la fin du fichier.
- Un offset positif signifie avancer dans le fichier et un décalage négatif signifie un reculer dans le fichier.

```
fseek(fichier,4L,0); /* aller au 5ème caractère dans le fichier */
fseek(fichier,4L,1); /* avance de cinq caractères */
fseek(fichier,-4L,1); /* recule de cinq caractères */
fseek(fichier,0L,0); /* aller au début de fichier - pareil que rewind() */
fseek(fichier,0L,2); /* aller à la fin du fichier */
fseek(ficheir,-10L,2);/* aller au 10 ème caractères avant la fin du fichier */
```

- Notez que le décalage passé à **fseek ()** doit être un entier long; L indique au compilateur de considérer le décalage comme tel.

Les fonctions d'accès aléatoire

- La fonction **ftell ()**, renvoie simplement la valeur de décalage du prochain caractère qui sera lu ou écrit.

- Par exemple, si 10 caractères ont déjà été lut à partir d'un fichier nommé fichier, l'appel de fonction :

```
ftell (fichier);
```

renvoie l'entier long 10. Cela signifie que le prochain caractère à lire est un décallage de 10 à partir du début du fichier, c.à.d le 11 caractères du fichier.

Exemple 5

- L'exemple suivant illustre l'utilisation de **fseek ()** et **ftell ()** pour lire et afficher un fichier dans un ordre inversé, du dernier caractère au premier.

```c
#include <stdio.h>
#include <stdio.h>
main( )
{
int ch, n;
long int decallage, last, ftell();
FILE *in_file;
in_file = fopen("exper.dat","r");
fseek(in_file,0L,2); // Aller jusqu'à la fin du fichier //
last = ftell(in_file); // enrgistrer le décallage (offset) du dernier caractère //
for (decallage = 0; decallage <= last; ++decallage)
{
    fseek(in_file, -decallage, 2); // position du caractère suivant //
    ch = getc(in_file);// obtient le caractère //
    switch(ch)
    {
        case '\n': printf ("LF : ");
        break;
        case EOF : printf("EOF: ");
        break;
        default : printf("%c: ",ch);
        break;
        }
}
```

Exemple 5

- Le programme va initialement au dernier caractère du fichier. Le décalage de ce caractère, qui est le caractère de fin, est enregistré dans la variable **last**.
- La fonction **ftell ()** renvoie un entier long.
- À partir de la fin du fichier, **fseek ()** est utilisée pour positionner le prochain caractère à lire, référencé à l'arrière du fichier.
- Lors de la lecture de chaque caractère, le caractère est affiché et le décalage est ajusté pour accéder au caractère suivant.

Table des matières

Exemple 5

- Il existe des fonctions prédéfinies en C autres que **printf()** et **scanf()**.

Fonction	Description
getchar()	Lire un caractère
gets()	Lire une chaine de caractère
putchar()	Afficher un carctère
puts()	Afficher une chaine de caractère

- Toutes ces fonctions sont définies dans le fichier en-tête <stdio.h>.

Références