Institut National des sciences Appliquées et des Technologies

Algorithmique et Structure de Données 1
Niveau MPI

Année universitaire
2019-2020

Dr. Aymen Sellaouti
Dr. Majdi Jribi

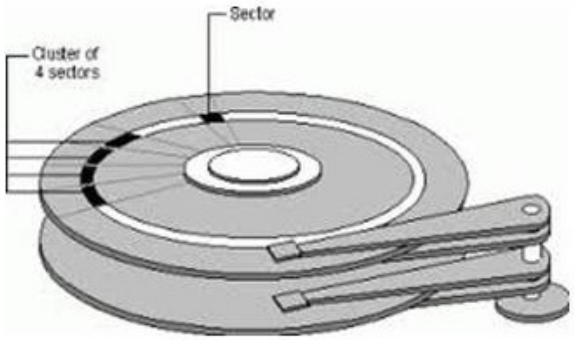
Chapitre 8
Les fichiers

- Partie 1: Introduction
- Partie 2: Ouverture et fermeture d'un fichier
- Partie 3: Lecture/Ecriture dans un fichier
- Partie 4: Exercices

- Partie 1: Introduction
- Partie 2: Ouverture et fermeture d'un fichier
- Partie 3: Lecture/Ecriture dans un fichier
- Partie 4: Exercices

Introduction

Disque Dur



Entrées/Sorties

Programme

- Ouvrir fichier X
- Lire D du fichier X
- Fermer fichier X
- R = Traitement (D)
- Ouvrir fichier Y
- Ecrire R dans fichier Y
- Fermer fichier Y

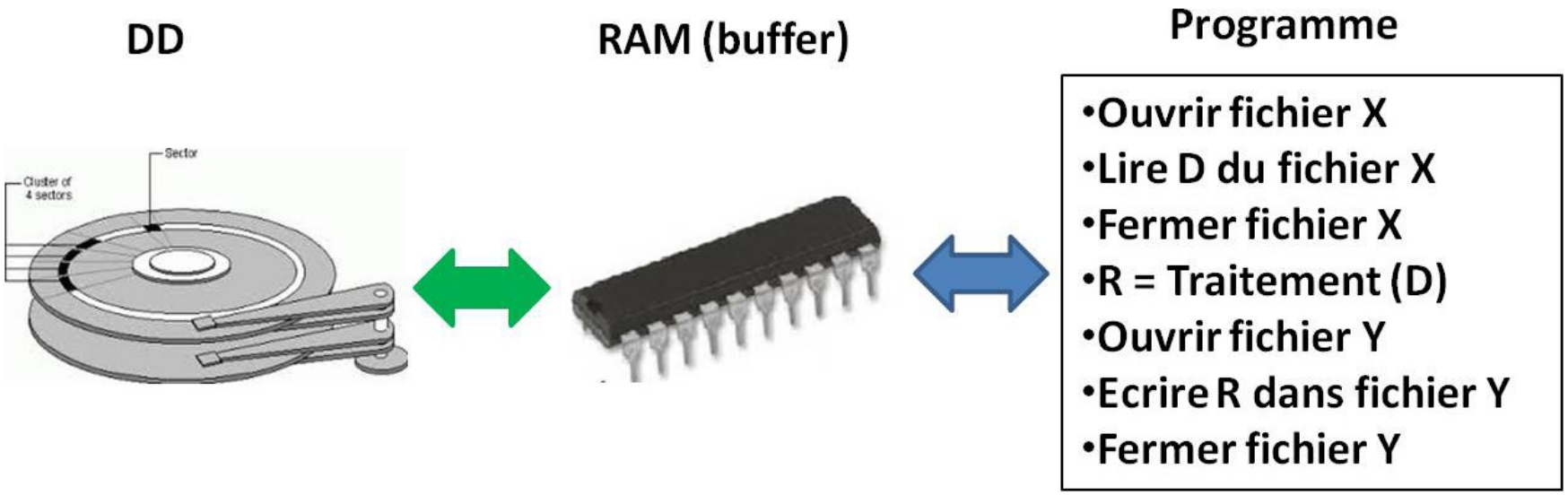
Introduction

Il existe deux types de Fichiers :

- **Les Fichiers textes** : sont les fichiers dont le contenu représente uniquement une suite de caractères imprimables, d'espaces et de retours à la ligne (.txt,...). Ils peuvent être lus directement par un éditeur de texte.

- **Les Fichiers binaires** : sont les fichiers qui ne sont pas assimilables à des fichiers textes (.exe, .mp3, .png,...). Ils ne peuvent pas être lus directement par un éditeur de texte .

Introduction



Dans mon programme , le système d'exploitation fait :

- Ouvrir un fichier.
  $\Rightarrow$ créer un *buffer (b)* dans la RAM.
- Lire/écrire dans le fichier ouvert.
  $\Rightarrow$ lire/écrire dans *b*.
- Fermer le fichier.
  $\Rightarrow$ "flusher" le contenu de *b*, libérer *b*,...

Introduction

- En langage C, les informations nécessaires à maintenir l'association **programme** ⇔ **buffer** ⇔ **disque dur** sont décrites dans une structure `FILE` (`stdio.h`).
- Parmi les informations stockées dans la structure `FILE` , on trouve :
  - *le N° du fichier à ouvrir,*
  - *le type d'ouverture (lecture/écriture),*
  - *l'adresse du* ***buffer*** *associé,*
  - *la position du curseur de lecture,*
  - *la position du curseur d'écriture,*
  - ...
- Pour utiliser un fichier, il faut donc commencer par déclarer une variable de type `FILE`, ou plus exactement un pointeur sur `FILE` (`FILE *`), qu'on appel aussi *flux de données* :

```
FILE * nomPointeurFichier;
```

- Partie 1: Introduction
- Partie 2: Ouverture et fermeture d'un fichier
- Partie 3: Lecture/Ecriture dans un fichier
- Partie 4: Exercices

Ouverture et fermeture d'un fichier

Le language C offre deux fonctions pour l'ouverture et la fermeture d'un fichier :

- La fonction `fopen` : permet d'ouvrir un fichier, suivant un *mode*, et retourne un flux (pointeur sur `FILE`).

```
FILE * fopen(char* nomFichier, char* mode)
```

  La fonction retourne `NULL` si l'ouverture n'est pas possible

- La fonction `fclose` : permet de fermer un fichier (un flux) ouvert.

```
void fclose(FILE * pf)
```

Modes d'ouverture d'un fichier

Les différents modes d'ouvertures d'un fichier sont :

Mode	Signification
"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
"rb"	ouverture d'un fichier binaire en lecture
"wb"	ouverture d'un fichier binaire en écriture
"ab"	ouverture d'un fichier binaire en écriture à la fin
"r+"	ouverture d'un fichier texte en lecture/écriture
"w+"	ouverture d'un fichier texte en lecture/écriture
"a+"	ouverture d'un fichier texte en lecture/écriture à la fin
"r+b"	ouverture d'un fichier binaire en lecture/écriture
"w+b"	ouverture d'un fichier binaire en lecture/écriture
"a+b"	ouverture d'un fichier binaire en lecture/écriture à la fin

Exemples


```
// Déclaration du flux
FILE * fp ;

// Ouvrir le fichier   test.text en écriture
// et association au flux
if ((fp=fopen("test.text","w"))==NULL){
     printf("Impossible d'ouvrir le fichier \n");

}

...

// Fermeture du flux (du fichier)
fclose(fp);
```

- Partie 1: Introduction
- Partie 2: Ouverture et fermeture d'un fichier
- Partie 3: Lecture/Ecriture dans un fichier
- Partie 4: Exercices

Lecture/Ecriture dans un fichier

- Une fois le fichier ouvert, le langage C permet plusieurs types d'accès à un fichier :
  - Par caractère
  - Par ligne
  - Par données formatées
  - Par enregistrement
  - direct

Accès par caractère

Plusieurs fonctions pour la lecture/écriture depuis/dans les fichiers textes existent :

- `int getc(FILE * pf)` : retourne le caractère suivant du flux `pf`. Elle retourne la constante `EOF` si elle rencontre la fin du fichier ou en cas d'erreur.
- `int putc(int c, FILE * pf)` : écrit le caractère c dans le fichier associé à`pf`. Retourne le caractère écrit ou `EOF` en cas d'erreur.

Remarques :

- `getchar()` ⇔ `getc(stdin)`
- `putchar(c)` ⇔ `putc(c,stdout)`

Accès par ligne

On peut accéder au contenu du fichier ligne par ligne ; en considérant les lignes comme chaîne de caractères dans les fichiers texte. Ainsi la fonction :

char * fgets(char *S, int max, FILE *f)

permet de lire une chaîne de caractères en s'arrêtant au caractère '\n' ou bien à max-1 caractères. Le résultat est stocké dans la chaîne de caractères S.

S : chaîne de caractères où sera stockée la ligne lue

max : nombre maximum de caractères à lire, en général cette variable le nombre maximum de caractères pouvant être lus, c'est à dire la taille de la zone de stockage: sizeof(s);

f : descripteur du fichier

Accès par ligne

La fonction ajoute à la chaîne S le caractère '\0' après le dernier caractère qu'elle a stocké dans le tableau.
La fonction retourne le pointeur S reçu en paramètre. Autrement dit elle retourne la ligne de texte lue à partir du fichier et stockée dans la chaîne S. Si la fin du fichier est atteinte, la fonction retourne le pointeur NULL.
Pour écrire dans un fichier une ligne de texte :

char fputs(char *S, FILE *f)

Cette fonction écrit la chaîne de caractères S dans le fichier de descripteur f, elle retourne le dernier caractère écrit.

Accès par données formatées

On peut aussi lire et écrire des variables de types quelconques, en utilisant ***fprintf()*** et ***fscanf()*** qui permettent de réaliser le même travail que *printf()* et *scanf()* sur des fichiers ouverts en mode texte:

fprintf(FILE *f, char *format, argument) ;

fscanf(FILE *f, char *format, &argument) ;

Exemples

Que font les instructions suivantes ?

```
char chaîne[80]
FILE *f ;
f=fopen(" …. ", "r") ;
if( !f)
        printf(« impossible d'ouvrir le fichier ») ;
else
{               while( fgets(chaîne, sizeof(chaîne), f) !=NULL)
                        puts(chaîne) ;
                fclose(f) ;
}
```

Accès par enregistrement

Permet de lire ou écrire les objets de type structure. Le fichier doit être ouvert en mode **binaire**. Les données échangées ne sont pas traitées comme du texte.

int fread (void *bloc, int taille, int nb, FILE *f)

int fwrite (void *bloc, int taille, int nb, FILE *f)

Les paramètres sont décrits comme suit :

bloc : Adresse de l'espace mémoire à partir duquel on fait l'échange avec le fichier qui reçoit ou fournit l'enregistrement. Cet espace mémoire :

- reçoit les enregistrements lus (dans le cas de fread).
- fournit les données à écrire (dans le cas de fwrite).

Il faut que cet espace soit de taille suffisante pour supporter le transfert des données

taille : taille de l'enregistrement en nombre d'octets (sizeof(enregistrement))

nb : nombre d'enregistrements à lire ou à écrire

f : descripteur du fichier

Les 2 fonctions retournent le nombre d'enregistrements lus/écrits.

Exemple


```c
#include <stdio.h>
#include <stdlib.h>
void main( ) {
    const int NB=50;
    FILE *in, *out;
    int tab1[NB], tab2[NB];
    int i;

    for (i = 0 ; i < NB; i++)
        tab1[i] = i;

    // écriture du tableau dans sortie.bin
    if ((out = fopen("sortie.bin", "wb")) == NULL)
        printf("\nImpossible d'écrire");
    else
    {
    fwrite(tab1, NB * sizeof(int), 1, out);
    fclose(out);  }
```

Exemple


```
// lecture dans sortie.bin
if ((in = fopen("sortie.bin", "rb")) == NULL)
      printf("\nImpossible de lire");

else

{
fread(tab2, NB * sizeof(int), 1, in);
fclose(in);

for (i = 0 ; i < NB; i++)
     printf("%d\t",tab2[i]);

printf("\n"); }


}
```

Accès direct

Le langage C permet un accès direct au données d'un fichier.

- `int fseek (FILE * pf, long int offset, int origine)` : affecte l'indicateur de position associé à `pf`, par la position `offset + origine`.
  - `pf` : pointeur sur `FILE` identifiant le flux.
  - `offset` : nombre de bytes à partir de `origine`.
  - `origine` : position à partir de laquelle `offset` est ajouté. Peut-être spécifié par l'une des constantes suivantes,
    - SEEK_SET Début du fichier
    - SEEK_CUR Position courante du pointeur
    - SEEK_END Fin du fichier.
- `long int ftell ( FILE * pf )` : retourne la valeur actuelle de l'indicateur de position.

La fonction fseek retourne une valeur 0 s'il n'y a pas de problème sinon une valeur différente de zéro en cas de problème

Exemple


```c
#include <stdio.h>

void main () {
   FILE * fp;
   long size;

   if ((fp = fopen ("monFichier.txt","rb"))==NULL) {
      printf("\nImpossible d'ouvrir le fichier");

   } else {
      fseek(fp, 0, SEEK_END);
      size = ftell(fp);
      fclose (fp);
      printf ("La taille de monFichier.txt: %ld bytes.\n",
                  size);
   }

}
```

Fin de fichier

int feof(FILE *f) détecte la fin du fichier dans le flux f.

• Cette fonction retourne une valeur différente de zéro si la fin de fichier est détectée sinon elle retourne zéro.

- Partie 1: Introduction
- Partie 2: Ouverture et fermeture d'un fichier
- Partie 3: Lecture/Ecriture dans un fichier
- Partie 4: Exercices

Exercices

Exercice 1

Ecrire un programme C qui calcule et affiche le nombre d'occurrence d'un caractère saisie au clavier dans un fichier texte dont on saisie le nom.

Exercices

Solution

```
#include<stdio.h>
Void main()
{char c, nom_fich[20];
int Nb;
FILE *f;
Printf("donnez le nom du fichier");
gets(nom_fich);
f=fopen(nom_fich,"r");
        If(f!=NULL)
                {
                printf("Taper le caractère");
                scanf("%c",&c);
                Nb=0;
                while(!feof(f))
                if(getc(f)==c)
                Nb++;
        Printf("le caractère %c se  trouve %d fois dans le fichier %s", c,nb,nom_fich);
                }
fclose(f);
}
```

Exercices

Exercice 2

Ecrire un programme C qui crée un fichier binaire de nom reels.dat, puis enregistre N réels saisis au clavier dans ce fichier.

Exercices

Solution

```
#include<stdio.h>
void main()
{
float x; int N,I;
FILE *f;
f=fopen(“reels.dat,”wb”);
printf(“Taper le nombre de réels a sauvgarder”);
scanf(“%d”,&N);
      for (i=1;i<=N;i++)
      {printf(“Taper un réel:”);
      Scanf(“%f”,&x);
      fwrite(&x,sizeof(float),1,f);
      }
fclose(f);
}
```

Exercices

Exercice 3

Ecrire un programme C qui calcule et affiche la moyenne des nombres réels stockés dans le fichier reels.dat crée dans l'exercice précédent.

Exercices

Solution

```
#include<stdio.h>
void main()
{
float x,s=0;
FILE *f;
f=fopen(“reels.dat,”rb”);
While(!feof(f))
{fread(&x,sizeof(float),1,f);
s=s+x;
}
      printf(“La somme des réels du fichier reels.dat est : %f”,s);
fclose(f);
}
```

Exercices

Exercice 4

On définit des étudiants par un nom, un prénom et un code (deux étudiants différents ne peuvent pas avoir le même code). Ecrire en C les fonctions suivantes :

CreeFichier :

qui permet de saisir le nom d'un fichier, un entier N ainsi que les noms, prénoms et codes des N étudiants pour construire le fichier.

AfficheFichier :

qui liste le contenu d'un fichier dont le nom est donne en paramètre.

Exercices

```
void CreeFichier  (char *nom_fichier){
ETUDIANT  e;
int n;
FILE *fp;
gets(nom_fichier);
fp = fopen(nom_fichier, “wb”);
If (! fp) printf(“impossible de creer le fichier”);
else
{ printf(“donner le nombre des etudiants”);
   scanf(“%d”,&n);
   for (i=0;i < n;i++)
            {
            scanf(“%d”,&e.code);
            gets(e.nom);
            gets(e.prenom);
            fwrite (&e,sizeof(ETUDIANT),1,fp);
            }
   fclose(fp);
 }
}
```

```
Typedef struct {
            int code;
            char nom[20];
            char prenom[20];
            }ETUDIANT;
```

Exercices

```
void AfficheFichier  (char *nom_fichier){
ETUDIANT  e;
int n;
FILE *fp;
gets(nom_fichier)
fp = fopen(nom_fichier, “rb”);
If (! fp) printf(“impossible d’ouvrir le fichier”);
else
{while (!feof(fp))
            {
            if (fread(&e,sizeof(ETUDIANT),1,fp)==1)
            {
            printf(“%d”, e.code);
            puts(e.nom);
            puts(e.prenom);
            }

fclose(fp);
 }
}
```