

Structures de Fichiers et Structures de Données

Chapitre 1

Généralités sur les fichiers

PLAN

Chapitre 1: Généralités sur les fichiers

1. Introduction
2. Les différents types de mémoires
3. Les notions de fichiers
4. La machine abstraite pour les structures de fichiers
5. Les fichiers en langage C
6. Conclusion

1. Introduction

Un fichier est le concept à travers lequel, un programme ou une application stocke des données en mémoire externe.

- Donnée VS Information:

Une donnée est un élément brut.

Une information est une donnée transformée ayant une sémantique (signification, un sens,...)

1. Introduction

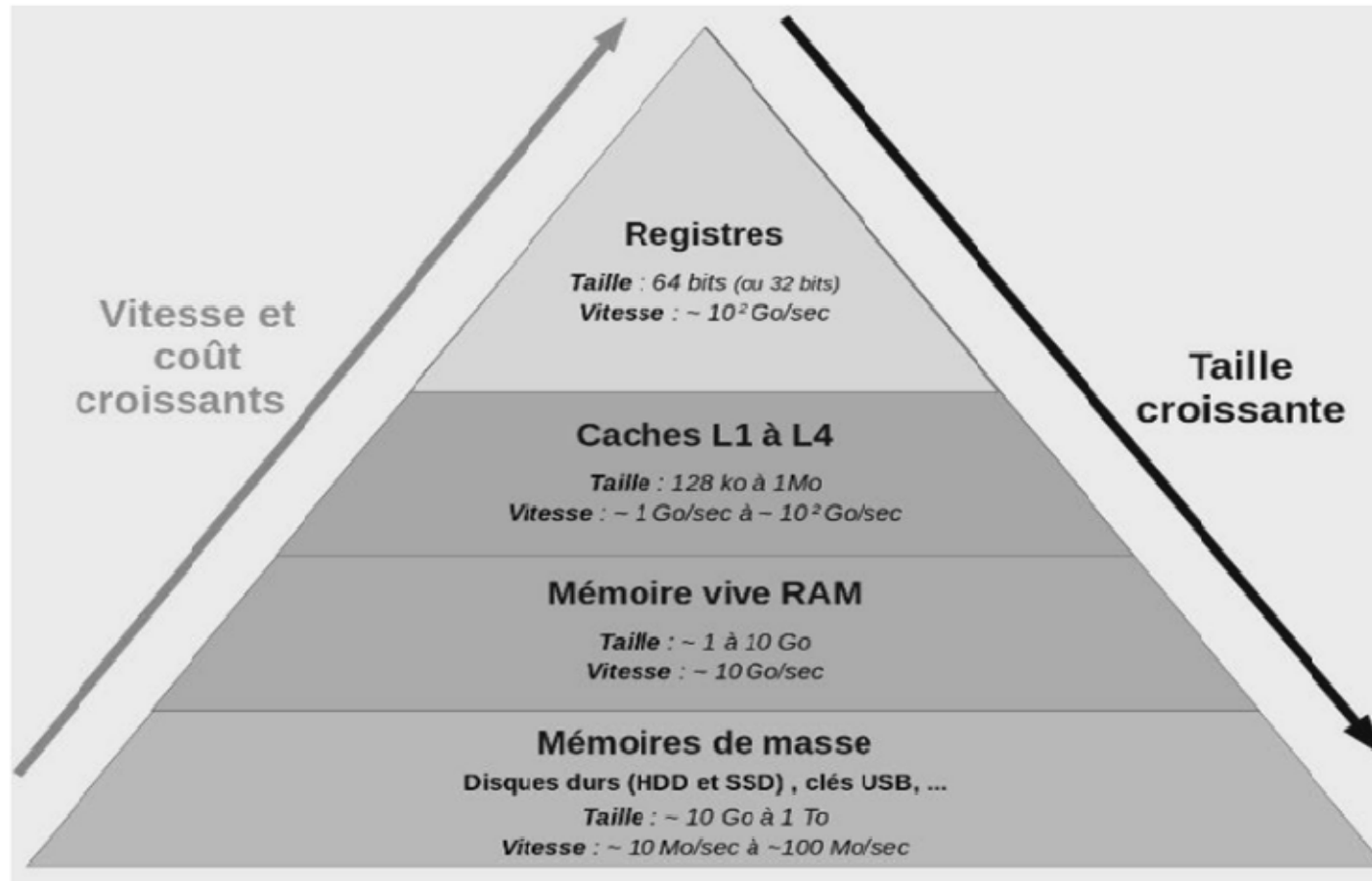
Les ensembles de données sont représentés par des structures de données externes et internes.

Leurs manipulations induit des transferts de données entre **mémoire centrale et mémoire externe (secondaire)**.

2. Les différents types de mémoire

- Les registres du processeur **très rapide** (qlq ns), **volatile**, taille de **quelques mots** mémoires
- La mémoire cache entre processeur et mémoire centrale (MC) plus rapide que la MC (qlq dizaines de ns), **volatile**, **très petite taille**
- La mémoire centrale (MC) **rapide** (qlq centaines de ns), **volatile**, relativement de **petite taille**
- Une zone tampon (jouant le rôle de cache) entre MC et MS c'est une petite partie de la mémoire centrale réservée par le système « **buffer cache** »
- Les différents types de mémoires secondaires (MS) principalement : disques magnétiques (ou disque durs HDD) et flash-disk (SSD...) **plus lents** que MC, **non volatiles**, **grande taille** et **moins coûteux** que MC
- Les mémoires d'archivage (tertiaire), généralement des bandes magnétiques (ou plus rarement des disques optiques), **très lentes**, **non volatile**, **grande taille** et généralement à **très faible coût**

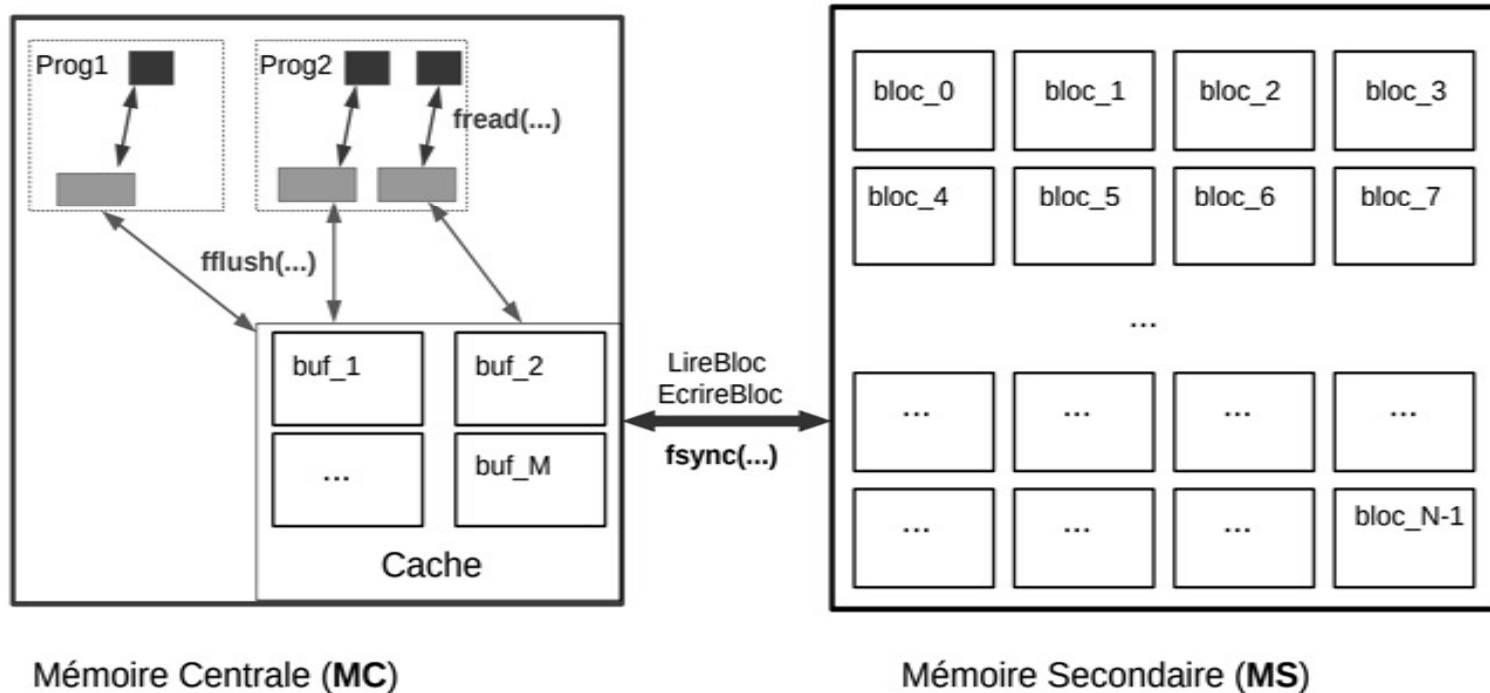
2. Les différents types de mémoire



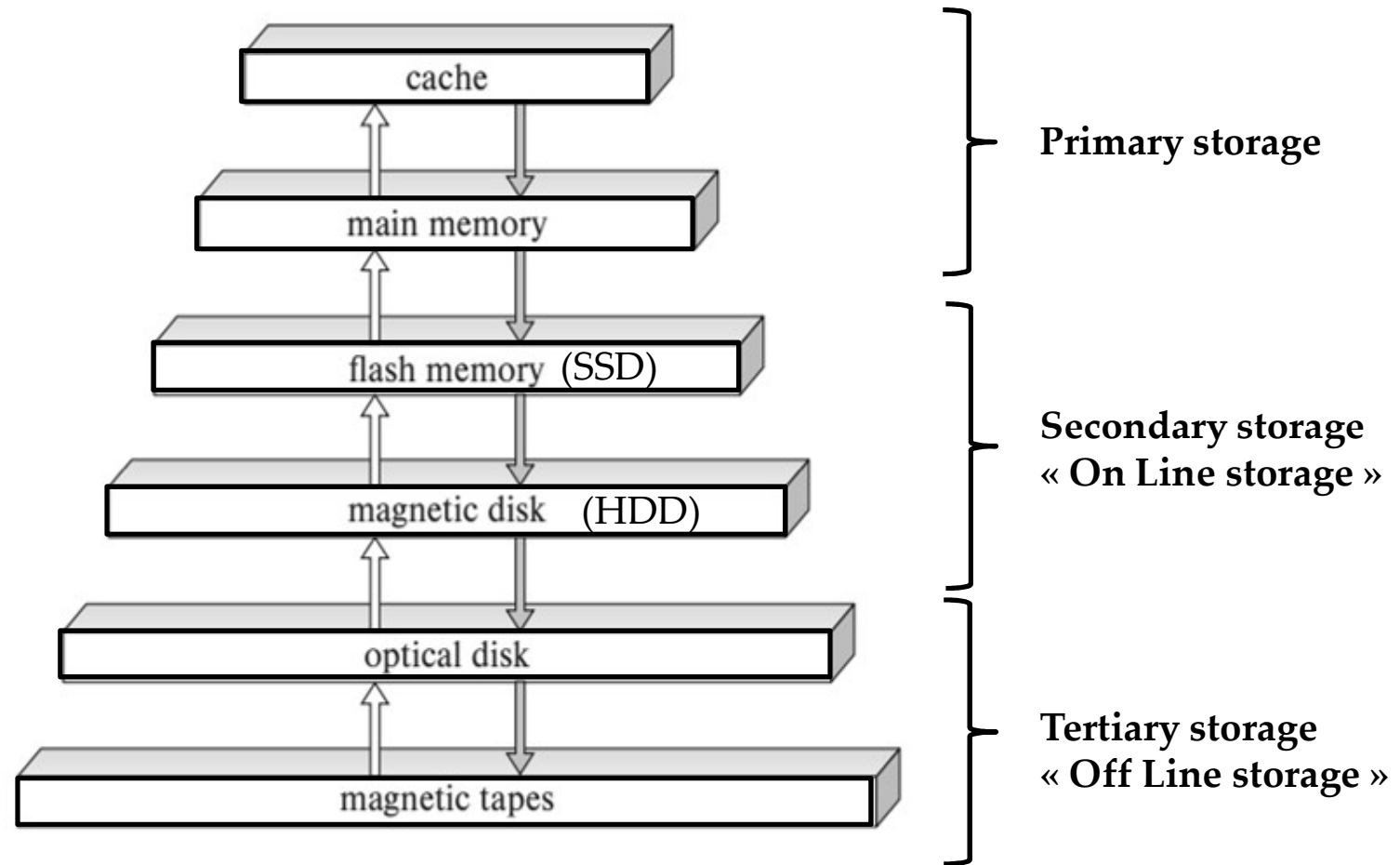
2. Les différents types de mémoire

➔ Zone tampon

Les buffers du cache (entre MS et MC) / Les buffers de l'application



2. Les différents types de mémoire



2. Les différents types de mémoire

Il existe plusieurs types de mémoires dans un ordinateur.

Chaque type de mémoire est caractérisé par:

- la vitesse d'accès,
- la taille,
- la nature de l'accès permis (lecture, écriture, effacement, ...),
- le coût
- et la volatilité (pertes d'information en cas de crash système ou redémarrage).

2. Les différents types de mémoire

Pourquoi doit-on utiliser un espace secondaire?

- La MC est de taille toujours limitée
- La MC est plus coûteuse que la MS
- La MC est volatile (perte de données, ...)

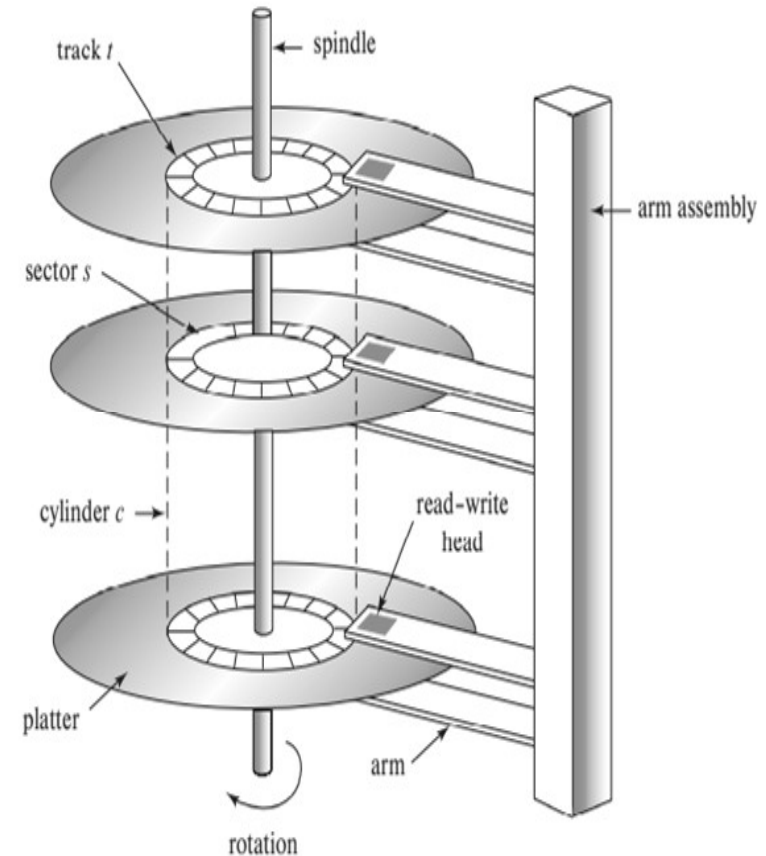
Problème avec le stockage sur les MS?

- Le temps d'accès est très lent en MS (10^{-3} secondes), par contre en MC très rapide (10^{-9} secondes).

2. Les différents types de mémoire

Disque Magnétique: HDD (Hard Disk Drive)

- Disque = ensemble de **plateaux** tournant avec une vitesse de rotation déterminée
- Plateau = ensemble de **pistes** concentriques numérotées 0, 1, 2, ... sur chaque face (**surface**)
- **Cylindre** = ensemble de pistes ayant un même diamètre sur les différents plateaux
- Piste = ensemble de **secteurs** de même taille
- Le **bras du disque** contient les têtes de lecture/écriture (une pour chaque face d'un plateau).



2. Les différents types de mémoire

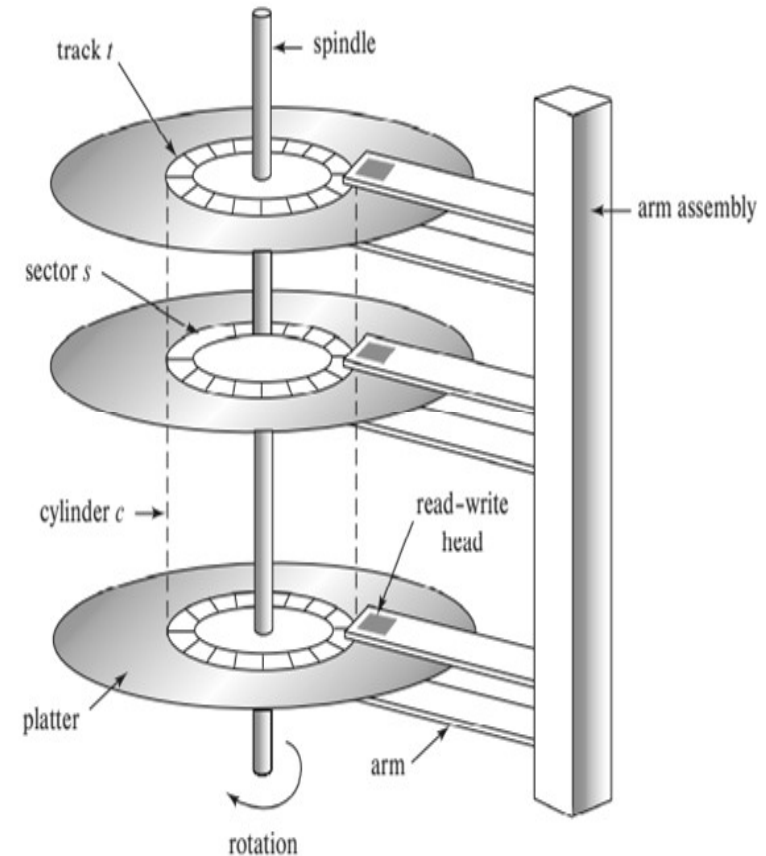
Disque Magnétique: HDD (Hard Disk Drive)

Exemple d'un DD de 1 To:

- 8 plateaux (donc 16 surfaces)
- 2^{16} , ou alors 65536, pistes par surface
- (en moyenne) $2^8 = 256$ secteurs par piste
- $2^{12} = 4096$ octets par secteur.

La capacité du disque est le produit :

16 surfaces (2^4) * 65536 pistes (2^{16}) * 256 secteurs (2^8) * 4096 octets (2^{12}) $\Rightarrow 2^{40}$ octets (1 To)

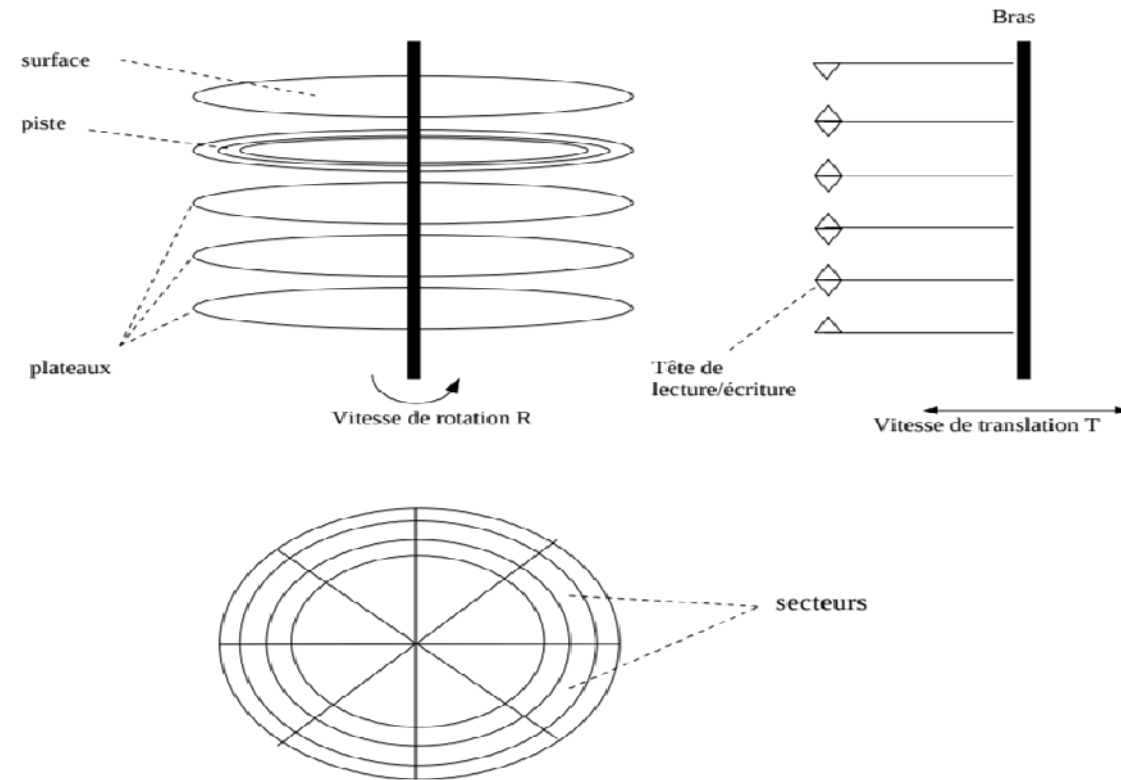


2. Les différents types de mémoire

Disque Magnétique: HDD (Hard Disk Drive)

Temps d'accès = temps du déplacement du bras + le temps d'attente de passage du secteur sous la tête de lecture/écriture, incluant le transfert de données)

Le temps de déplacement du bras (temps de translation) est plus grand que le temps de rotation.



2. Les différents types de mémoire

Flash Memory

NAND flash → SSD (Solid State Drive)

- * C'est un dispositif de stockage externe (MS) à base de mémoire flash NAND, robuste et offrant de meilleures performances que le disque magnétique.
- * Les données sont stockées dans des micropuces.
- * La mémoire est divisée en blocs (ex: 256 Ko) et chaque bloc est composé d'un certain nombre de pages (ex: 4 Ko).



2. Les différents types de mémoire

Flash Memory

NAND flash → SSD (Solid State Drive)

Il y a 3 opérations possibles :

- Lire une page ⇒ très rapide (20 microsecondes)
- Ecrire une page, à condition qu'elle soit dans un état effacé ⇒ rapide (100 à 200 microsecondes)
- Effacer toute les pages d'un bloc ⇒ lente (quelques millisecondes)

2. Les différents types de mémoire

Flash Memory

NAND flash → SSD (Solid State Drive)

Inconvénients:

- Le nombre de fois qu'une page flash peut être effacée est limité (environ 100 000 à 1 000 000 fois)

Une fois cette limite atteinte, des erreurs dans le stockage des bits sont susceptibles de se produire.

3. Les notions de fichiers

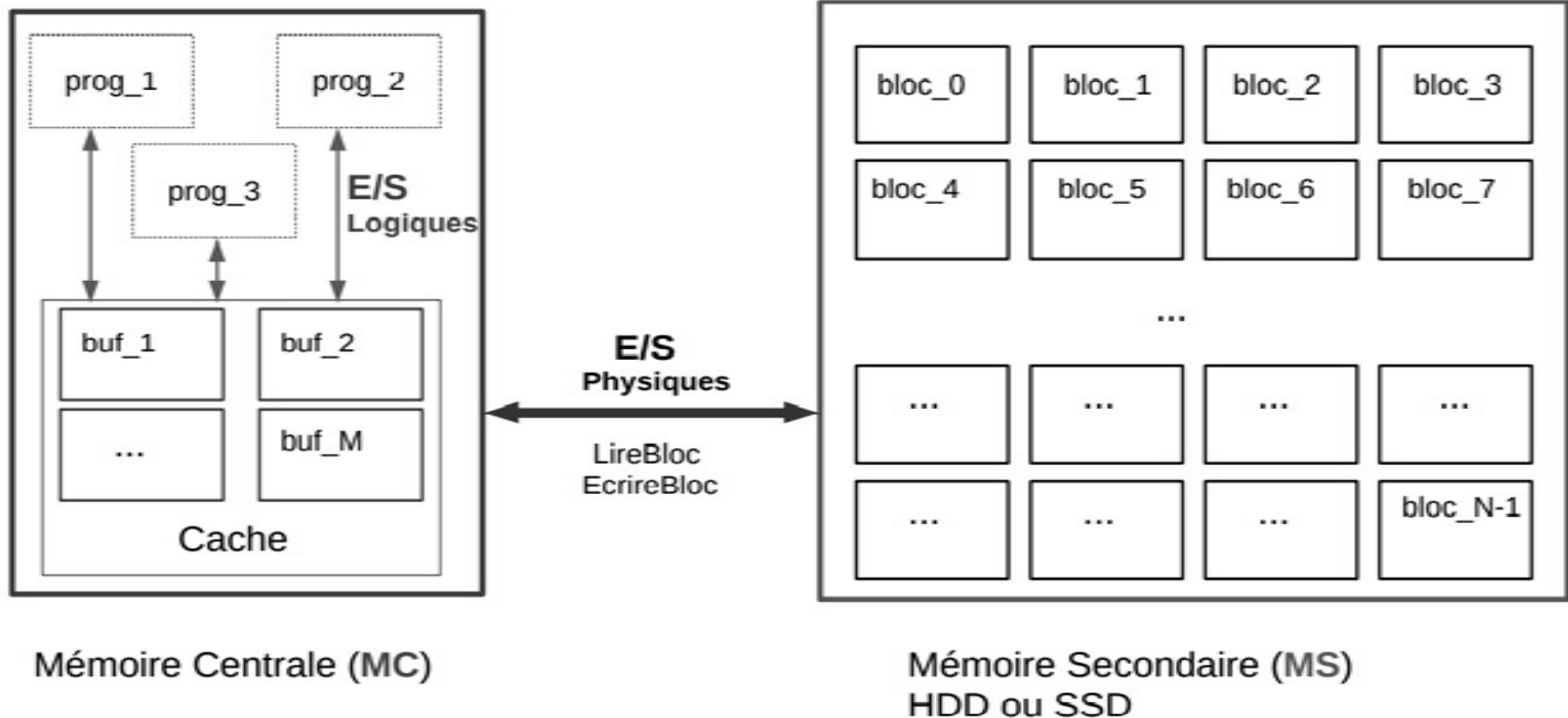
On considère la mémoire secondaire MS comme une grande zone de stockage formée de **blocs physiques** de même **taille fixe**.

Selon le type de la MS, le **bloc physique**, au niveau du système d'exploitation, peut être composé d'un ou de plusieurs secteurs d'un HDD ou alors d'une ou de plusieurs pages d'un SSD.

Chaque **bloc physique** représente donc un **tableau d'octets** pouvant être lu ou écrit en une seule opération d'accès (ou d'E/S physique) :

LireBloc(num, Buf) et EcrireBloc(num, Buf)

3. Les notions de fichiers



3. Les notions de fichiers

Définition

Un fichier est le concept à travers lequel, un programme ou une application stocke des données en mémoire externe.

Les fichiers sont utilisés à différents niveaux d'abstraction avec des sémantiques différentes:

- Au niveau applicatif (haut niveau)
- Au niveau interne (système ou niveau physique)

3. Les notions de fichiers

Définition

- Au niveau applicatif -

←
Fichier **Typé**
(suite enregistrements)

→
Fichier **Non Typé**
(flux d'octets)

Un fichier est une suite d'enregistrements (articles) en MS

Ex : $F = \langle e_1, e_2, e_3, \dots e_n \rangle$

Chaque enregistrement est formé d'un certain nombre de champs (attributs)

Ex : $e = \{ \text{nom : 'aaa' , age : 19 , adr : 'cccccc' } \}$

Un fichier est une suite d'octets (caractères) en MS

Ex : $F = \langle a,b,c,d,e,f,g,h,i,j,k,l,m, \dots \rangle$

3. Les notions de fichiers

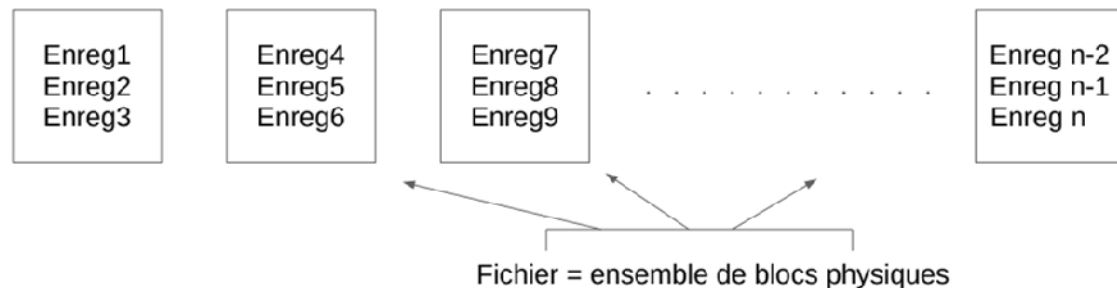
Définition

- Au niveau interne -

Un fichier est un ensemble de blocs physiques en MS, renfermant une suite d'octets non interprétés.

Les données (par exemple les enregistrements) du fichier sont stockés à l'intérieur des blocs selon une certaine organisation.

L'accès aux contenus des blocs se fait via les opérations d'E/S bufférisées (**LireBloc** et **EcrireBloc**).



3. Les notions de fichiers

Fichiers **BINAIRES**

Un fichier binaire est simplement une suite d'octets sans aucune interprétation particulière faite par le système. Ces suites d'octets représentent les données telle qu'elles étaient représentées en MC avant leur transfert sur le fichier. De ce fait les données d'un fichier binaire sont très dépendantes du système où elles ont été produites (donc généralement peu portables).

Fichiers **TEXTES**

Un fichier texte est formé d'un ensemble de lignes terminées chacune par une marque de fin de ligne.

Dans les systèmes de type Unix (comme Linux par exemple), cette marque de fin de ligne est le caractère '**\n**' (code ascii 10). Dans d'autres systèmes, la marque de fin de ligne peut être composée de deux caractères (comme '**\r**' et '**\n**').

Les fonctions de lecture et d'écriture sur les fichiers textes, permettent de s'abstraire de ces différences entre systèmes.

3. Les notions de fichiers

Définition - Bloc d'entête -

Un ensemble d'informations sont nécessaires pour l'exploitation du fichier, c'est ce qu'on appelle les **caractéristiques** du fichiers. Ces informations sont sauvegardés soit dans des fichiers spéciaux (les répertoires) gérés par le systèmes d'exploitation soit au début de chaque fichier (**bloc d'entête**) des fois gérés par l'application elle-même.

Il existe deux types d'information :

- informations statiques, telles que le nom du fichier, la date de création, etc.
- informations dynamique telles que le nombre courant d'articles, l'adresse du dernier bloc, etc.

3. Les notions de fichiers

Définition

- Méthode d'accès -

- une manière d'organiser les blocs du fichier sur MS
- le placement des enregistrements à l'intérieur des blocs
- les caractéristiques et informations nécessaires pour manipuler le fichier
- le nombre de buffers à réserver en MC pour optimiser les accès
- l'implémentation des opérations d'accès (recherche, insertion, suppression, ...)

Le but des structures de fichiers est **d'optimiser les performances d'accès** (temps d'exécution et occupation mémoire)

3. Les notions de fichiers

Définition

- Méthode d'accès -

Le principal critère de performance se résume au nombre d'Entrée/Sortie physiques effectuées. La complexité algorithmique est alors directement liée au **nombre d'Entrée/Sortie**.

Un autre critère de performance concerne l'occupation de **l'espace mémoire**

- L'encombrement mémoire :
⇒ Taille des structures de données utilisées / Taille des données stockées \geq 100 % (plus le rapport est petit, plus c'est intéressant)
- Le facteur de chargement :
⇒ Nombre de données insérées / Nombre de place disponibles dans le fichier
] 0 % , 100%] (plus le rapport est grand, plus c'est intéressant)

4. La machine abstraite pour les structures de fichier

Pour écrire des algorithmes sur les structures de fichiers, on utilisera la machine abstraite définie par le modèle suivant:

{Ouvrir, Fermer, LireDir, EcrireDir, Aff_Entete, Entete, Allocbloc }

Dans ce modèle, on manipule des numéros de blocs relatifs au début de chaque fichier (c'est donc des numéros logiques).

L'utilisation des adresses physiques n'est pas d'une utilité particulière à ce niveau de la présentation.

Dans ce modèle, un fichier est donc un ensemble de blocs numérotés logiquement (1, 2, 3, ... n).

4. La machine abstraite pour les structures de fichier

Déclaration d'un fichier *f* et de ses zones tampons *buf1*, *buf2* (deux buffers) :

Const *b*=...

Type Tenreg = Structure // le contenu des enregistrements

champs1: Typeqq

champs2: Typeqq

 ...

fin

Type TBloc = Structure // le contenu des blocs

tab : Tableau [*b*] de Tenreg

NB : entier

fin

f : **FICHER** de TBloc **BUFFER** *buf1*, *buf2* **ENTETE** (*type*₁, *type*₂, ...*type*_{*m*});

4. La machine abstraite pour les structures de fichier

OUVRIR(F , nomfichier , mode)	Ouvre ou crée un fichier Mode = 'A' veut dire ouvrir en lecture/écriture un fichier qui existe déjà. Les caractéristiques seront lues en MC lors de l'ouverture. Mode = 'N' veut dire créer un nouveau fichier en lecture/écriture. Les caractéristiques seront allouées en MC lors de la création
FERMER(F)	Ferme le fichier. Les caractéristiques seront sauvegardées en MS
LIREDIR(F , i , buf)	Lecture du bloc numéro i de F dans la variable buf
ECRIREDIR(F , i , buf)	Ecriture de buf dans le bloc numéro i de F
ENTETE(F , i)	Retourne la valeur de la caractéristique numéro i
AFF_ENTETE(F , i , v)	Affecte v à la caractéristique numéro i
ALLOC_BLOC(F)	Retourne le numéro d'un nouveau bloc alloué à F

4. La machine abstraite pour les structures de fichier

Exemple:

Const b= 4

Type Tetudiant = Structure // le contenu des enregistrements

 matricule: entier;

 nom: chaine[30];

fin

Type TBloc = Structure // le contenu des blocs

 tab : Tableau [b] de Tetudiant;

 nb : entier;

fin

f : **FICHER** de TBloc **BUFFER** buf **ENTETE** (entier, entier);

// caractéristiques: -1- nombre de blocs

 -2- nombre d'enregistrements

4. La machine abstraite pour les structures de fichier

//Affichage du contenu d'un fichier

Début

OUVRIR (F , « fichier.dat » , 'A'); // ouverture en mode ancien

Nbre_Bloc \leftarrow **ENTETE** (F , 1);

i \leftarrow 1;

TANTQUE i \leq Nbre_Bloc FAIRE // parcourir le fichier bloc par bloc

LIREDIR (F , i , Buf); //Lire le bloc i dans Buf

 j \leftarrow 1;

TANTQUE j \leq Buf.nb FAIRE // parcourir à l'intérieur du buffer

 Val \leftarrow Buf.tab[j];

 écrire (Val); j++

FTQ

 i++

FTQ

FERMER (F);

Fin

5. Les fichiers en langage C

- Déclaration d'une variable fichier f (de type flux): `FILE *f`
- Ouverture d'un fichier:
`FILE * fopen(const char * filename, const char * mode)`

Le mode peut être:

Mode du Fichier texte	Mode du Fichier binaire	
"r"	"rb"	ouverture en lecture
"w"	"wb"	création d'un nouveau fichier (écrasement de l'ancien s'il existe déjà)
"a"	"ab"	ouverture en mode « ajout »
"r+"	"rb+"	ouverture en lecture/écriture
"w+"	"wb+"	création d'un nouveau fichier en lecture/écriture
"a+"	"ab+"	ouverture en mode « ajout » en lecture/écriture

- Fermeture d'un fichier: `fclose(f)`

5. Les fichiers en langage C

Lecture / Ecriture en mode texte

- Pour lire un caractère dans un fichier texte, on peut utiliser :

c = fgetc(f);

Lit et retourne (type int) le prochain caractère de f ou bien la constante EOF si la fin de fichier a été dépassée.

En cas d'erreur EOF est aussi retournée.

- Pour écrire un caractère dans un fichier texte, on peut utiliser :

fputc(c, f);

Ecrit le caractère c à la position courante du fichier f. En cas d'erreur, la fonction retourne (type int) la constante EOF.

5. Les fichiers en langage C

Lecture / Ecriture en mode texte

- Pour lire une ligne dans un fichier texte, on peut utiliser :

fgets(buf, n, f);

Lit dans la variable buf, tous les caractères à partir de la position courante de f, jusqu'à trouver une marque de fin de ligne '\n' (qui est aussi lue dans buf) ou alors jusqu'à ce que n-1 caractères soient lus. Un caractère de fin de chaîne '\0' est rajouté à la fin de buf. En cas d'erreur ou de dépassement de la fin de fichier, la fonction fgets retourne NULL.

- Pour écrire une chaîne de caractères dans un fichier texte, on peut utiliser :

fputs(buf, f);

Ecrit tous les caractères contenus de buf (sauf le '\0') dans le fichier f, à partir de la position courante. Pour écrire une ligne, il faut prévoir un caractère '\n' à la fin de buf (et avant le caractère de fin de chaîne '\0').

En cas d'erreur, la fonction retourne (type int) EOF.

5. Les fichiers en langage C

Lecture / Ecriture en mode texte

- Pour effectuer une lecture formatée depuis un fichier texte, on peut utiliser :
fscanf(f, format, &var1, &var2, ...)
Comme scanf, sauf que les données proviennent du fichier texte f (au lieu de la console)
- Pour effectuer une écriture formatée dans un fichier texte, on peut utiliser :
fprintf(f, format, exp1, exp2, ...)
Comme printf, sauf que les données iront dans le fichier texte f (au lieu de la console)

5. Les fichiers en langage C

Lecture / Ecriture en mode binaire

- Pour lire des données d'un fichier binaire, on utilise généralement 'fread' :

`nbelt_lus = fread(buf, taille_elt, nb_elt, f);`

Demande de lecture d'un nombre d'éléments consécutifs égal à `nb_elt`, chacun de taille `taille_elt` octets depuis le fichier `f`.

Le résultat de la lecture sera placé dans la zone pointée par `buf` et de taille au moins égale à `taille_elt * nb_elt` octets.

La fonction retourne le nombre d'éléments effectivement lus, qui peut donc être inférieur au nombre demandé (en cas de fin de fichier ou d'erreur de lecture)

5. Les fichiers en langage C

Lecture / Ecriture en mode binaire

- Pour écrire des données dans un fichier binaire, on utilise généralement 'fwrite' :

nbelt_ecr = fwrite(buf, taille_elt, nb_elt, f);

Demande d'écriture d'un nombre d'éléments égal à nb_elt, chacun de taille taille_elt octets dans le fichier f.

Les octets à écrire (au nombre de taille_elt*nb_elt) sont à récupérer depuis la zone pointée par buf.

La fonction retourne le nombre d'éléments effectivement écrits, qui peut donc être inférieur au nombre demandé (en cas d'erreur d'écriture).

5. Les fichiers en langage C

Lecture / Ecriture en mode binaire

- Pour modifier la position courante dans un fichier, on peut utiliser 'fseek' :

fseek(f, déplacement, origine);

Déplace la position courante d'un nombre d'octets égal à déplacement, relativement à :

- début du fichier, si origine vaut SEEK_SET
- position courante, si origine vaut SEEK_CUR
- fin du fichier, si origine vaut SEEK_END

Voir exemples sur CodeBolcks



Management

Projects Files FSy

Workspace

Exemple1

Sources

main.c

Exemple2

Exemple3

Exemple4

Exemple5

main.c X main.c X main.c X main.c X main.c X

```

1  /*
2  * Construction d'un fichier binaire avec n enregistrements : < nom , telephone >
3  */
4
5  #include <stdio.h>
6  int main()
7  {
8      // variable enregistrement
9      struct Tenreg {
10         char nom[20];
11         char tel[15];
12     } e;
13
14     // variable fichier
15     FILE *f;
16     char nomf[30];
17     printf( "\nConstruction d un fichier agenda telefonique\n\n");
18     printf( "Donnez le nom du fichier a construire : ");
19
20     // un espace avant %s permet de sauter tous les caractères blancs avant de lire le nom de fichier
21     scanf( " %s", nomf );
22
23     // creation du nouveau fichier en mode binaire
24     f = fopen( nomf, "wb" );
25     if ( f == NULL ) {
26         printf( "erreur lors de l ouverture du fichier %s en mode wb\n", nomf );
27         return 0;
28     }
29
30     // lecture d'un enregistrement depuis la console
31     printf( "donnez un nom et un tel (ou 0 0 pour terminer le programme) : " );
32     // un espace avant %s permet de sauter tous les caractères blancs avant de lire les données e.nom et e.tel
33     scanf( " %s %s", e.nom, e.tel );
34     while ( e.nom[0] != '0' ) {
35         // écriture dans le fichier
36         fwrite( &e, sizeof(e), 1, f );
37         // lecture d'un enregistrement depuis la console
38         printf( "donnez un nom et un tel (ou 0 0 pour terminer le programme) : " );
39         // un espace avant %s permet de sauter tous les caractères blancs avant de lire les données e.nom et e.tel
40         scanf( " %s %s", e.nom, e.tel );
41     }
42
43     // fermeture du fichier
44     fclose( f );
45     return 0;
46 }
47

```

```
"C:\DONNEES\ESI-SFSD\EL Allia\Exemples\Exemple1\bin\Debug\Exemple1.exe"

Construction d un fichier agenda telefonique

Donnez le nom du fichier a construire : Exemple1
donnez un nom et un tel (ou 0 0 pour terminer le programme) : Nadia
1111111111
donnez un nom et un tel (ou 0 0 pour terminer le programme) : Papa
2222222222
donnez un nom et un tel (ou 0 0 pour terminer le programme) : Manan
3333333333
donnez un nom et un tel (ou 0 0 pour terminer le programme) : Chouchou
4444444444
donnez un nom et un tel (ou 0 0 pour terminer le programme) : 0
0

Process returned 0 (0x0)   execution time : 95.724 s
Press any key to continue.
```

```
42
43 // fermeture du fichier
44 fclose( f );
45 return 0;
46 }
47
```



Management

Projects Files FSy

Workspace

- Exemple1
- Exemple2
 - Sources
 - main.c
- Exemple3
- Exemple4
- Exemple5

```

1  /*
2  * Liste le contenu d'un fichier binaire avec enregistrements : < nom , telephone >
3  */
4  #include <stdio.h>
5  int main()
6  {
7      // variable enregistrement
8      struct Tenreg {
9          char nom[20];
10         char tel[15];
11     } e;
12
13     // variable fichier
14     FILE *f;
15     char nomf[30];
16     int n, i;
17     printf( "\nAffichage du contenu d'un fichier agenda telefonique\n\n" );
18     printf( "Donnez le nom du fichier à lister : " );
19
20     // un espace avant %s permet de sauter tous les caractères blancs avant de lire le nom de fichier
21     scanf( " %s", nomf );
22
23     // ouverture du fichier
24     f = fopen( nomf, "rb" );
25     if ( f == NULL ) {
26         printf( "erreur lors de l'ouverture du fichier %s en mode rb\n", nomf );
27         return 0;
28     }
29
30     // lecture des enregistrements depuis le fichier
31     i = 1;
32     n = fread( &e, sizeof(e), 1, f );
33     if ( n == 1 ) // si le nombre d'anreg lus = 1
34         printf( "%3d nom : %s \t tel : %s\n", i++, e.nom, e.tel );
35
36     while ( ! feof(f) ) {
37         n = fread( &e, sizeof(e), 1, f );
38         if ( n == 1 ) // si le nombre d'anreg lus = 1
39             printf( "%3d nom : %s \t tel : %s\n", i++, e.nom, e.tel );
40     }
41
42     // fermeture du fichier
43     fclose( f );
44     return 0;
45 }
46

```


main.c [Exemple2] - Code::Blocks 20.03

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global> main(): int

Management

Projects Files FSy
Workspace
Exemple2
Sources
main.c

```
main.c x
9      char nom[20];
10
11      // Affichage du contenu d'un fichier agenda telephonique
12
13      // Donner le nom du fichier à lister : Exemple2
14      1 nom : Nadia      tel : 1111111111
15      2 nom : Papa      tel : 2222222222
16      3 nom : Maman     tel : 3333333333
17      4 nom : Chouchou  tel : 4444444444
18
19      Process returned 0 (0x0)  execution time : 18.872 s
20      Press any key to continue.
21
22      // Fermeture du fichier
23      fclose( f );
24      return 0;
25  }
```

Logs & others

Code::Blocks x Search results x Cccc x Build log x Build messages x CppCheck/Vera++ x CppCheck/Vera++ messages x Cscope x Debugger x DoxyBlocks x Fortran info x Close

C:\DONNEES\ESI-SFSD\EL Allia\Exemples\Exemple2\main.c

C/C++

Windows (CR+LF)

WINDOWS-1252

Line 45, Col 2, Pos 1267

Insert

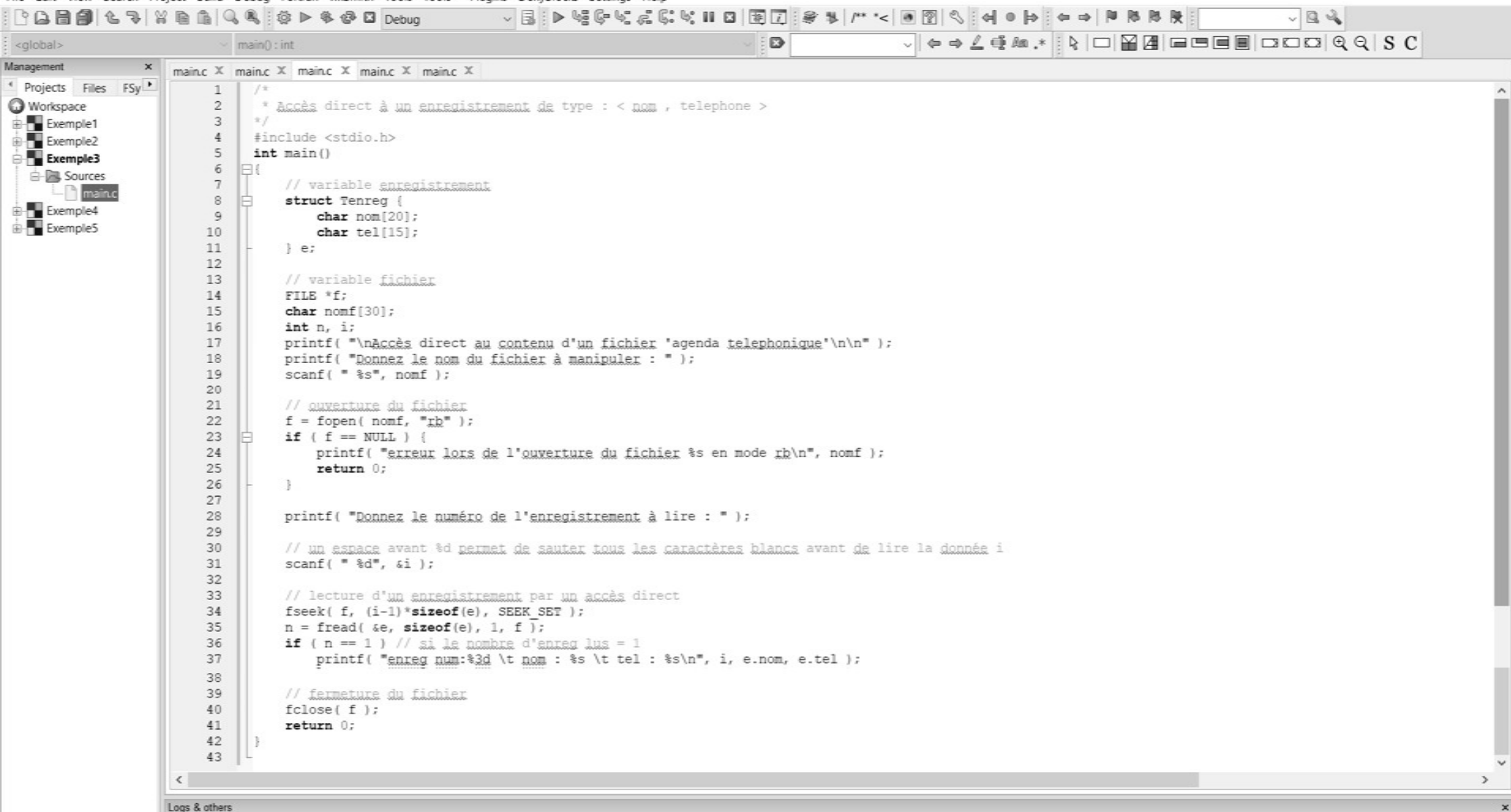
Read/Write

default

Taper ici pour rechercher

Windows taskbar icons

System tray icons: 21°C, 10:16 PM 10/10/2021



```
1  /*
2  * Accès direct à un enregistrement de type : < nom , telephone >
3  */
4  #include <stdio.h>
5  int main()
6  {
7      // variable enregistrement
8      struct Tenreg {
9          char nom[20];
10         char tel[15];
11     } e;
12
13     // variable fichier
14     FILE *f;
15     char nomf[30];
16     int n, i;
17     printf( "\nAccès direct au contenu d'un fichier 'agenda telephonique'\n\n" );
18     printf( "Donnez le nom du fichier à manipuler : " );
19     scanf( " %s", nomf );
20
21     // ouverture du fichier
22     f = fopen( nomf, "rb" );
23     if ( f == NULL ) {
24         printf( "erreur lors de l'ouverture du fichier %s en mode rb\n", nomf );
25         return 0;
26     }
27
28     printf( "Donnez le numéro de l'enregistrement à lire : " );
29
30     // un espace avant %d permet de sauter tous les caractères blancs avant de lire la donnée i
31     scanf( " %d", &i );
32
33     // lecture d'un enregistrement par un accès direct
34     fseek( f, (i-1)*sizeof(e), SEEK_SET );
35     n = fread( &e, sizeof(e), 1, f );
36     if ( n == 1 ) // si le nombre d'enreg lus = 1
37         printf( "enreg num:%3d \t nom : %s \t tel : %s\n", i, e.nom, e.tel );
38
39     // fermeture du fichier
40     fclose( f );
41     return 0;
42 }
43
```

"C:\DONNEES\ESI-SFSD\EL Allia\Exemples\Exemple3\bin\Debug\Exemple3.exe"

Accès direct au contenu d'un fichier 'agenda téléphonique'

Donnez le nom du fichier à manipuler : Exemple3

Donnez le numéro de l'enregistrement à lire : 4

enreg num: 4 nom : Chouchou tel : 4444444444

Process returned 0 (0x0) execution time : 7.295 s

Press any key to continue.

35 n = fread(&e, sizeof(e), 1, f);

36 if (n == 1) // si le nombre d'enreg lus = 1

37 printf("enreg num:%3d \t nom : %s \t tel : %s\n", i, e.nom, e.tel);

38

39

40

```
1  /*
2  * Construction d'un fichier texte avec n enregistrements <nom,telephone>
3  */
4  #include <stdio.h>
5  int main()
6  {
7      // variable enregistrement
8      struct Tenreg {
9          char nom[20];
10         char tel[15];
11     } e;
12
13     // variable fichier
14     FILE *f;
15     char nomf[30];
16     printf("\nConstruction d un fichier agenda telephonique\n\n");
17     printf("Donnez le nom du fichier a construire : ");
18
19     // un espace avant %s permet de sauter tous les caractères blancs avant de lire le nom de fichier
20     scanf(" %s", nomf);
21
22     // creation du nouveau fichier en mode texte
23     f = fopen( nomf, "w" );
24     if ( f == NULL ) {
25         printf( "erreur lors de l'ouverture du fichier %s en mode w\n", nomf );
26         return 0;
27     }
28
29     // insertion des enregistrements lus à la console
30     printf( "donnez un nom et un tel (ou 0 0 pour terminer le programme) : " );
31
32     // un espace avant %s permet de sauter tous les caractères blancs avant de lire les données e.nom et e.tel
33     scanf( " %s %s", e.nom, e.tel );
34
35     while ( e.nom[0] != '0' ) {
36         // écriture formatée dans le fichier
37         fprintf( f, " %s , %s\n", e.nom, e.tel ); // un enregistrement par ligne
38         printf( "donnez un nom et un tel (ou 0 0 pour terminer le programme) : " );
39         scanf( " %s %s", e.nom, e.tel );
40     }
41
42     // fermeture du fichier
43     fclose( f );
44     return 0;
45 }
46
```

```
9      char nom[20];
10     char tel[15];
11     } e;
12
```

Construction d'un fichier agenda téléphonique

Donnez le nom du fichier à construire : Exemple4.txt

donnez un nom et un tel (ou 0 0 pour terminer le programme) : Mohamed

777

donnez un nom et un tel (ou 0 0 pour terminer le programme) : Abdeldjalil

333

donnez un nom et un tel (ou 0 0 pour terminer le programme) : Noor

111

donnez un nom et un tel (ou 0 0 pour terminer le programme) : Maria

555

donnez un nom et un tel (ou 0 0 pour terminer le programme) : 0

0

Process returned 0 (0x0) execution time : 61.955 s

Press any key to continue.

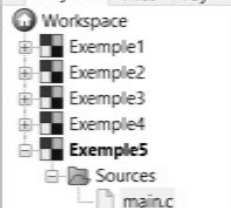
et e.tel

```
45 }
46
```



Management

Projects Files FSy



```
1  /*
2  * Liste le contenu d'un fichier texte (enregistrements de type <nom,telephone>)
3  */
4  #include <stdio.h>
5  int main()
6  {
7      // variable enregistrement
8      struct Tenreg {
9          char nom[20];
10         char tel[15];
11     } e;
12
13     // variable fichier
14     FILE *f;
15     char nomf[30];
16     int n, i;
17     printf( "\nAffichage du contenu d'un fichier agenda telephonique\n\n" );
18     printf( "Donnez le nom du fichier à lister : " );
19     scanf( " %s", nomf );
20
21     // ouverture du fichier texte en mode lecture
22     f = fopen( nomf, "r" );
23     if ( f == NULL ) {
24         printf( "erreur lors de l'ouverture du fichier %s en mode r\n", nomf );
25         return 0;
26     }
27
28     // lecture des enregistrements depuis le fichier
29     i = 1;
30     n = fscanf( f, "%s , %s", e.nom, e.tel );
31     if ( n == 2 ) // si le nombre d'elements lus = 2 (le nom et le tel)
32         printf( "%3d \t nom : %s \t tel : %s\n", i++, e.nom, e.tel );
33
34     while ( ! feof(f) ) {
35         n = fscanf( f, "%s , %s", e.nom, e.tel );
36         if ( n == 2 ) // si le nombre d'elements lus = 2 (le nom et le tel)
37             printf( "%3d \t nom : %s \t tel : %s\n", i++, e.nom, e.tel );
38     }
39
40     // fermeture du fichier
41     fclose( f );
42     return 0;
43 }
44
```

```
7 // variable enregistrement
8 struct Tenreg {
```

"C:\DONNEES\ESI-SFSD\EL Allia\Exemples\Exemple5\bin\Debug\Exemple5.exe"

Affichage du contenu d'un fichier agenda telefonique

Donnez le nom du fichier à lister : Exemple5.txt

```
1 nom : Mohamed tel : 777
2 nom : Abdeldjalil tel : 333
3 nom : Noor tel : 111
4 nom : Maria tel : 555
```

Process returned 0 (0x0) execution time : 15.460 s
Press any key to continue.

```
41 fclose( f );
42 return 0;
43 }
44
```

6. Conclusion

Hypothèse de travail:

Durant le cours de SFSD, on considérera des fichiers de données formés par des enregistrements pour présenter les différentes structures de fichiers étudiées.

Les enregistrements peuvent être composés d'un ou de plusieurs champs (ou attributs). L'un de ces champs sera considéré comme clé de recherche (**Search Key**). C'est le champ utilisé dans les conditions de recherche d'enregistrements.

Les enregistrements du fichier de données sont répartis à travers les blocs physiques alloués au fichier.

6. Conclusion

Les prochains chapitres concernent les différents types d'organisations de fichiers et l'étude de leur impacts sur les performances. Ce sont les "**méthodes d'accès**" aux fichiers ou alors "structures de fichiers".