Généralités

sur les fichiers 2

1. Introduction

Avant l'introduction de l'outil informatique, les fichiers étaient traités manuellement. A ce moment-là, on effectuait déjà, certain traitement a savoir: la création, la suppression, la réunion, éclatement, extraction, la mise a jour de fichiers. Ces traitement sont qualifies de traitement fonctionnels.

Avec l'utilisation de l'ordinateur pour la manipulation des fichiers, en plus des traitements fonctionnels, d'autre traitements ont vu le jour, des traitement de servitude. Il s'agit de duplication de fichiers, de mémorisation temporaire…etc. Leur raison d' être tient essentiellement de contraintes technologiques telle que: Espace mémoire, le cout de stockage…

2.Operations fondamentales

2.1 La Création

Créer un fichier revient a :

- Créer sa structure.
- Saisir les articles du fichier et les stocker sur un support magnétique (optique).

Exemple :

Pour créer le fichier Etudiants, on définit sa structure et la taille de ses articles comme suit :

Nom du champ	*Longueur du champ*
Numéro étudiant	6 caractères
Nom étudiant	10 caractères
Prénom étudiant	10 caractères
Date naissance	8 caractères
Adresse étudiant	20 caractères

Puis, on saisit les informations relatives à chaque étudiant et on sauvegarde le fichier sur le disque sous le nom : *Etudiants.*

2.2 La Suppression

Supprimer un fichier revient à annuler son stockage, c'est-à-dire, à effacer tous les enregistrements qui le constituent, ainsi que sa structure. On distingue deux types de suppression : suppression logique et suppression physique.

- **La *suppression logique consiste à marquer le fichier de manière à* le rendre *transparent, en réalité, il existe toujours sur le support.***

- **La *suppression physique efface le fichier définitivement. L'espace* précédemment occupé par le fichier sera récupéré.**

3. la mise a jour

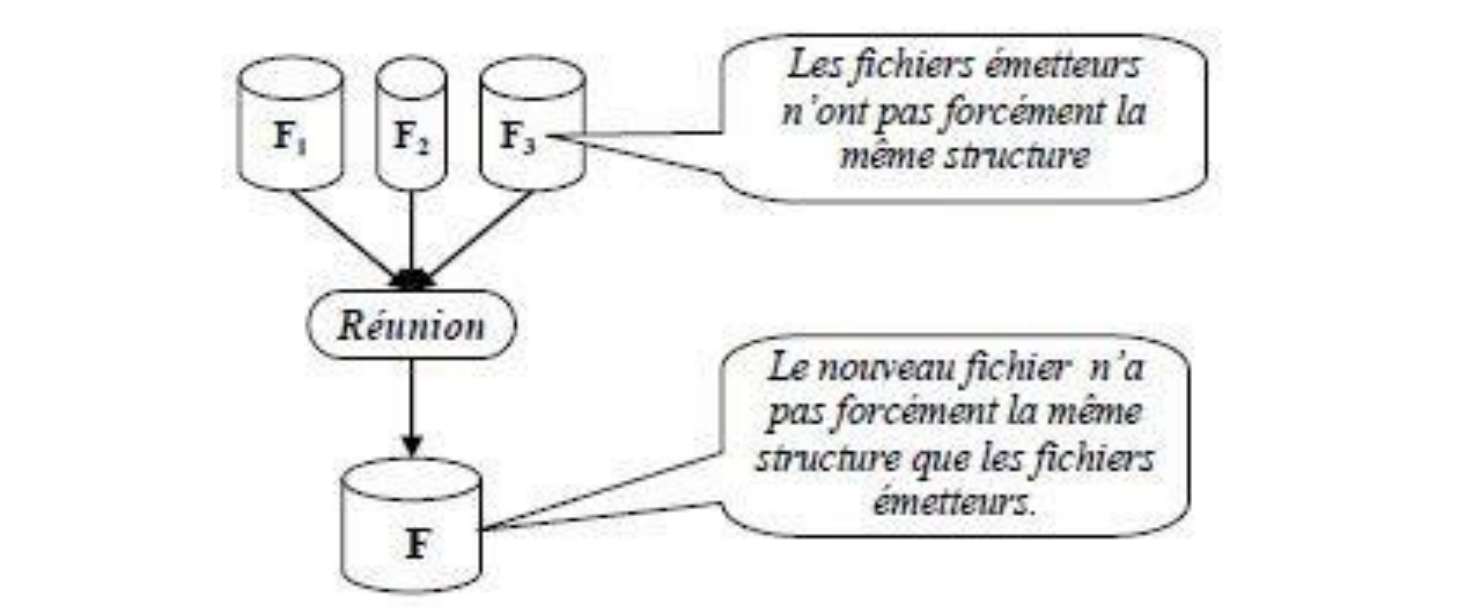
La mise à jour englobe les trois traitements suivants :

- La création de nouveaux enregistrements
- La suppression d'enregistrements existants
- La modification du contenu d'un enregistrement

La mise à jour est réalisée, généralement, sur une fichier permanent, via un fichier mouvement.

2.4 La réunion

Plusieurs fichiers émetteurs donnent naissance à un nouveau fichier.



Exemple :

Dans les fichiers émetteurs, nous avons l'information « Adresse employé » de type alphanumérique (par exemple : chorfa 10190 bouira).

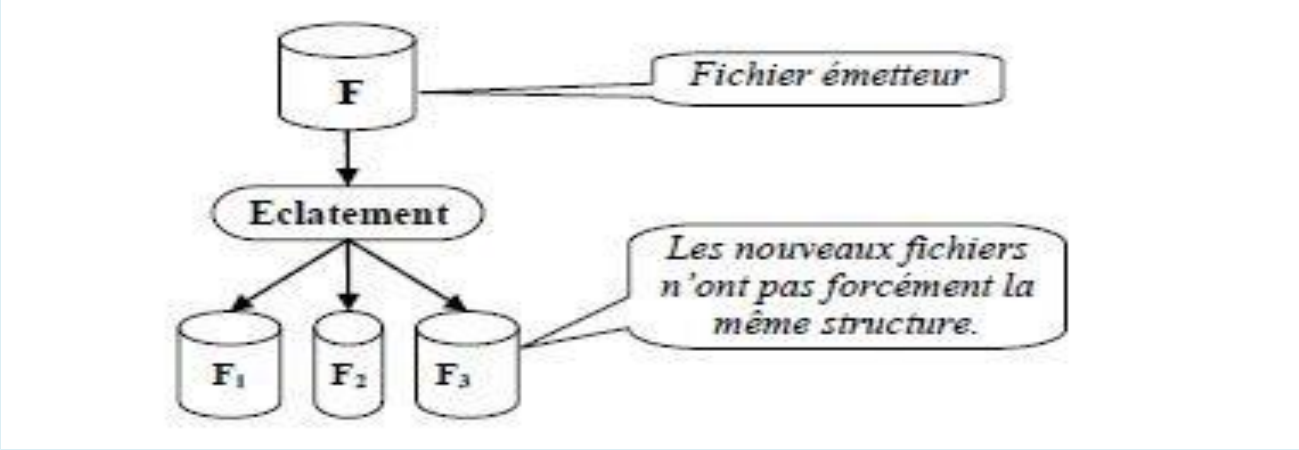
Dans le nouveau fichier, nous avons besoin d'effectuer des traitements qui dépendent de la wilaya. Dans ce cas, on va éclater le champ « Adresse » en sous champs (Ville, Code postal, Wilaya) comme suit : Ville :

- Chorfa
- Code postal : 10190
- Wilaya : Bouira

L'information reste la même, seule la manière de la représenter diffère.

2.5 L'éclatement

C'est l'opération inverse de la réunion. Un fichier émetteur donne naissance à plusieurs fichiers récepteurs. Là, encore, des modifications peuvent être apportées à la structure des fichiers mais pas au contenu



2.6 Le Tri

Trier un fichier revient à classer ses enregistrements dans un ordre croissant ou décroissant de la valeur d'un ou de plusieurs attributs appelés **arguments de tri.**

Exemple :

On considère le fichier Etudiants contenant les informations : (Numéro étudiant, nom étudiant, prénom étudiant, filière étudiant).

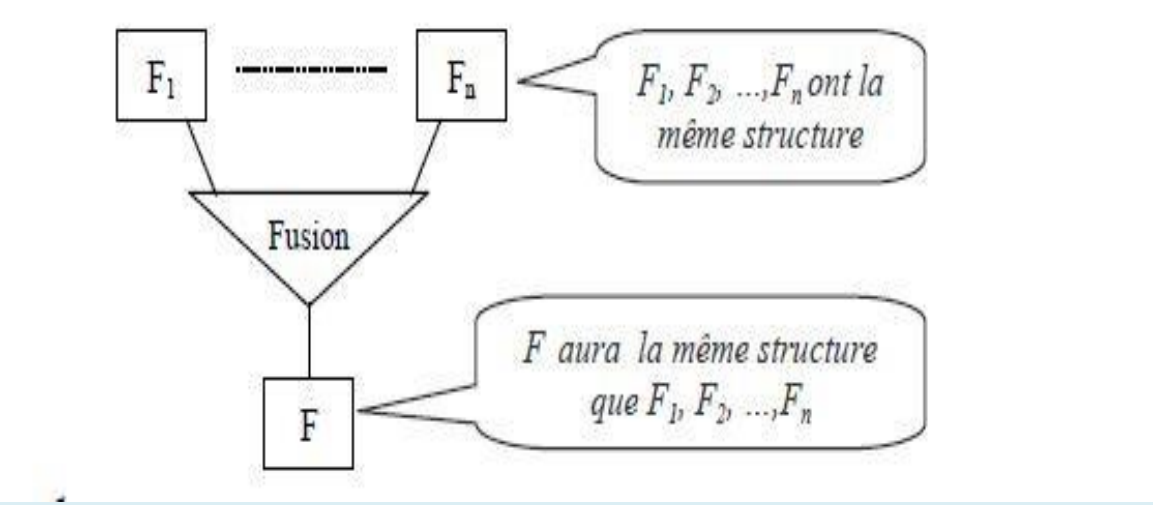
La manière la plus simple de trier ce fichier est de choisir la clé (numéro étudiant) comme argument de tri. Mais, cette solution n'est pas toujours conseillée. En effet, il faut prendre en compte l'utilisation future du fichier et les besoins de l'utilisateur.

2.7 La Fusion

Elle consiste à regrouper les enregistrements de deux ou plusieurs fichiers au sein d'un seul fichier.

Condition : Les fichiers à fusionner doivent avoir la même structure.

Conséquence : Le fichier résultant aura la même structure que les fichiers qui lui ont donné naissance.



Exemple :

On considère que dans un établissement scolaire, les étudiants sont gérés selon la filière étudiée à travers trois fichiers :

Le fichier *Etud_Inf* :

Numéro	*Nom*	*Prénom*	*Adresse*	*Filière étudiée*
21-101	Ben Nabi	Malek	Tizi Ouzou	Informatique
21-102	Ben Ziad	Tarek	Dellys	Informatique
21-103	Ben Bouali	Hassiba	Alger	Informatique

Le fichier *Etud_com* :

Numéro	*Nom*	*Prénom*	*Adresse*	*Filière étudiée*
21-201	Belkacem	Krim	Bouira	Comptabilité
21-202	Abane	Ramdane	M'sila	Comptabilité
21-203	N'soumer	Fathma	Tizi Ouzou	Comptabilité

Le fichier *Etud_Fis* :

Numéro	*Nom*	*Prénom*	*Adresse*	*Filière étudiée*
21-301	Mammeri	Mouloud	Tebessa	Fiscalité
21-302	Djaout	Tahar	Boumerdès	Fiscalité
21-303	Feraoun	Mouloud	Setif	Fiscalité

Exemple :

Le résultat de la fusion est le fichier *Etudiants* suivant :

Numéro	*Nom*	*Prénom*	*Adresse*	*Filière étudiée*
21-201	Belkacem	Krim	Bouira	Comptabilité
21-202	Abane	Ramdane	M'sila	Comptabilité
21-203	N'Soumer	Fathma	Tizi Ouzou	Comptabilité
21-301	Mammeri	Mouloud	Tebessa	Fiscalité
21-302	Djaout	Tahar	Boumerdès	Fiscalité
21-303	Feraoun	Mouloud	Setif	Fiscalité
21-101	Ben Nabi	Malek	Tizi Ouzou	Informatique
21-102	Ben Ziad	Tarek	Dellys	Informatique
21-103	Ben Bouali	Hassiba	Alger	Informatique

Vous remarquez que ce fichier a exactement la même structure que les fichiers :*Etud_Inf, Etud_com et Etud_Fis.*

Le fichier Etudiants est trié dans l'ordre alphabétique des filières.
(Argument de tri= filière).

2.8 Extraction

Ce traitement consiste à extraire ou à recopier des enregistrements ou des parties d'enregistrements sur un autre support selon un critère donné.

Exemple :

Imprimer la liste des étudiants admis à partir d'un fichier ***Etudiants*** contenant les informations : numéro étudiant – nom étudiant – prénom étudiant – adresse étudiant – spécialité étudiant – résultats étudiant.

La liste imprimée contiendra uniquement les informations : numéro étudiant – nom étudiant – prénom étudiant – résultats étudiant.

2.9 La copie

Copier un fichier (traitement de servitude) revient à dupliquer son contenu sur un support. Ce traitement peut être justifié par différente raison :

- Permettre un temps d'accès plus rapide.
- Garantir une fiabilité plus grande pour éviter les pertes d'informations.

3. Différences entre MC (Mémoire Centrale) et MS (Mémoire Secondaire)

Toute donnée doit être présente en mémoire centrale avant d'être manipulée par un programme. La mémoire centrale est une mémoire de travail.

Par contre la mémoire secondaire n'est utilisée que pour sauvegarder les informations avant d'être chargées en mémoire centrale pour être traitées. C'est une zone de stockage.

La mémoire secondaire (MS) est généralement de grande taille mais le temps d'accès est très grand relativement à la mémoire centrale (MC). De plus, la MS est généralement non volatile et le coût à l'unité de stockage est moindre que celui de la MC. Voici quelques caractéristiques des deux types de mémoires :

- **Mémoire Centrale** : circuit électronique (VLSI)
- **Mémoire Secondaire** : généralement dispositif mécanique (disques, bandes, ...)
- **Temps d'accès** : MC 100 000 à 1000 000 fois plus rapide que le disque magnétique
- **Coût par octet (prix)** : MC plus chère que MS
- **Persistance** : MC volatile, MS persistante

4. Fichiers physique et fichier logique

Le lien qui existe entre ces deux notions est pratiquement le même que celui qui existe entre le plan d'une maison et la maison elle-même avant et après sa construction.

1 fichier logique :

- Le fichier logique est décrit par sa structure (la structure des enregistrements qu'il contient).
- Le fichier logique ne dépend pas du support physique qui va être utilisé pour le stockage du fichier.

2. fichier physique :

- Le fichier physique est le résultat de stockage d'un fichier logique sur un support physique.
- Le fichier physique est défini par son contenu et son support physique.

5. Enregistrement physique et enregistrement logique

5.1 Enregistrement Logique :

Les enregistrements d'un fichier logique sont dits enregistrements logiques ou articles.

La taille d'un enregistrement est mesurée en octets ou en caractères; Elle peut être fixe, variable ou indéfinie.

On distingue 3 types d'enregistrements :

Enregistrement de longueur fixe: la longueur d'un enregistrement est définie par le nombre de caractères de sa structure.

EXEMPLE : **Fichier étudiant**

8c	NCE	enregistrement de longueur fixe = 50 caractères
20c	Nom	
20c	prénom	
2c	section	

Enregistrement de longueur variable :

EXEMPLE :

Fichier employé: enregistrement de longueur variable:
CNSS, Nom, Prénom, Grade, Fonction, Enfant**(s),** État civil
Car la zone Enfant(s) est variable : Les employés n'ont pas le même nombre d'enfants.

Solution : crée un fichier pour les enfants ayant la structure suivante :
Enfant (CNSS, Prénom, DN)

1	**ALI**	**01/01/1985**
1	**HEDIA**	**31/12/1987**
1	**MOSTAPHA**	**03/05/2000**
2	**JIHENE**	**04/06/2001**

5. Enregistrement physique et enregistrement logique

5.2 Enregistrement physique :

- Les enregistrements d'un fichier physique sont dits enregistrements physiques.
- L'enregistrement physique représente la quantité d'informations échangée entre la mémoire centrale et l'unité de stockage.

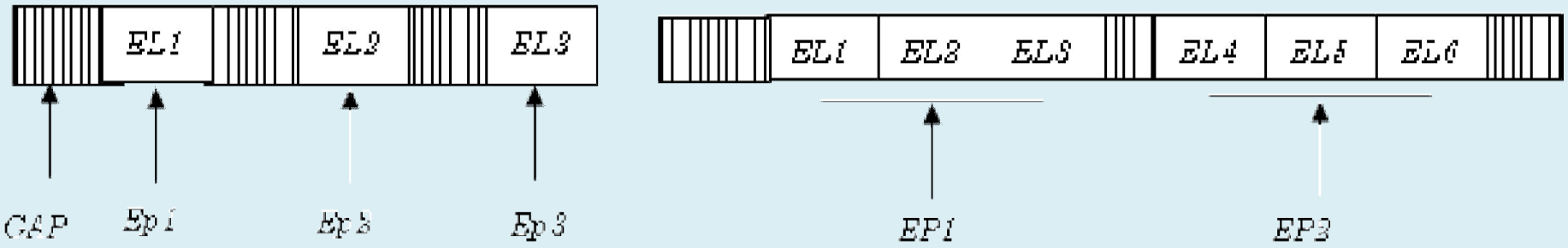
Un enregistrement physique ou Bloc est la plus petite entité de données qui peut être lue ou écrite en une seule opération. L'enregistrement physique contient un ou plusieurs enregistrements et éventuellement certaines informations de contrôle et d'organisation.

6. Le Facteur de blocage, son intérêt

Le nombre d'enregistrements logiques contenus dans un enregistrement physique correspond au facteur de Blocage. Le blocage de certain nombre d'enregistrements logiques en enregistrements physiques permet de réaliser essentiellement un gain de temps dont l'exécution des opérations d'E/S.

le facteur de blocage est calculée de manière à occuper au mieux les blocs physiques.

EXEMPLE : Cas d'une bande Magnétique.

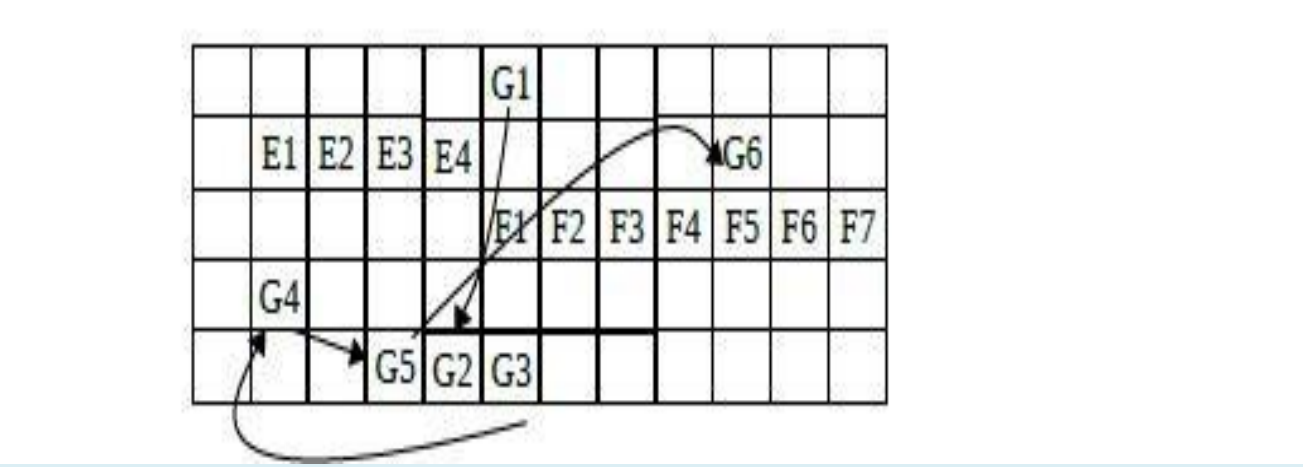


GAP : Espace pour accélérer ou ralentir la vitesse de lecture de l'enregistrement.
Soit un fichier de 1000 enregistrements logiques avec facteur du blocage égal à 1, on a donc besoin de 1000 E/S pour lire la totalité du fichier par contre on a uniquement besoin de 500 opérations E/S si le facteur de blocage est égal à 2.

7. Le fichier statique et le fichier dynamique

on modélise la MS par une zone contiguë de blocs numérotés séquentiellement (ces numéros représentent les adresses de blocs).

Les blocs sont des zones contiguës d'octets de même taille, renfermant, entre autre, les données (enregistrements) des fichiers. Dans le schéma suivant, la MS contient 3 fichiers E, F et G



Le fichier E est formé de 4 blocs contigus, le fichier F est formé de 7 blocs contigus et le fichier G est une liste de blocs.

7. le fichier statique et le fichier dynamique

Pour écrire des algorithmes sur les structures de fichiers on utilisera la machine abstraite définie par le modèle suivant:
{ouvrir, fermer, lireDir, ecrireDir, lireSeq, ecrireSeq, aff_entete, entete, allocbloc }

Un fichier est donc un ensemble de blocs **numérotés logiquement (1, 2, 3, ... n)**

Déclaration d'un fichier ' f ' et de ses zones tampons 'buf1', 'buf2' (deux buffers) : var f : FICHIER de TypeBloc BUFFER buf1, buf2 ENTETE (type1, type2, ...typem); Dans cette ligne de déclaration, il y a la définition de :

- Une variable f de type FICHIER

- Deux variables buf1 et buf2, de type 'TypeBloc' qui servent comme zones tampons pour lire et écrire les données du fichier.

- la structure de l'entête du fichier, formée par m caractéristiques dont les types sont spécifiés

Les opérations du modèle sont :

- **Ouvrir (f, nomfichier, mode)**

pour ouvrir ou créer un fichier de nom 'nomfichier' selon la valeur de 'mode'
« a » : ouvrir un ancien fichier en lecture/écriture et charge ses caractéristiques en MC
« n » : créer un nouveau fichier en lecture/écriture et alloue une zone en MC pour ses Caractéristiques

- **Fermer (f)**

ferme le fichier f, en sauvegardant ses caractéristiques sur disque en cas de modifications.

- **LireDir (F, i, buf)**

lire dans le buffer 'buf' le contenu du bloc numéro 'i' du fichier 'f' (numéro logique de blocs).

- **EcrireDir (F, i, buf)**

écrit le contenu de la variable 'buf' dans le bloc numéro 'i' du fichier (numéro logique)

- **LireSeq(F, buf)**

lire dans 'buf' le contenu du bloc courant du fichier 'f'. (lecture séquentielle, le bloc courant pointera le suivant dans le fichier)

- **EcrireSeq(F, buf)**

écrit le contenu de la variable 'buf' dans le bloc courant du fichier (écriture séquentielle, le bloc courant pointera le suivant dans le fichier)

- **Entete(f, i)**

retourne la valeur de la 'i'ème caractéristique du fichier 'f' (ne nécessite pas d'accès disque car après l'ouverture du fichier, ses caractéristiques se trouvent déjà chargés en MC)

- **Aff_entete(f, i, val)**

affecte la valeur 'val' dans la 'i'ème caractéristique du fichier 'f' (ne nécessite pas d'accès disque car après l'ouverture du fichier, ses caractéristiques se trouvent déjà chargés en MC)

- **AllocBloc(f)**

alloue un nouveau bloc au fichier et retourne son numéro. Dans un fichier vu comme tableau de blocs, cette opération n'est pas nécessaire, il suffit juste d'écrire après la fin du fichier)

Les fichiers sont très important pour la gestion de n'importe quelle application (gestion de stock, gestion personnel, gestion de scolarité) puisque il permettent de regrouper toutes les informations indispensable pour une bonne gestion; c'est-à-dire faciliter leurs mises à jour, leur contrôle, leur sécurité, leur confidentialité, partage et archivage.

Conclusion :

Les fichiers sont gérés par le **système gestion de fichiers SGF** qui est inclus dans le système d'exploitation; le SGF est en charge de :

- création/suppression de fichiers.
- contrôle d' accès aux fichier.
- partage des fichiers entre plusieurs utilisateurs.
- protection des fichiers contre la destruction.

Les fichiers en langage C

Pour déclarer une variable fichier (de type flux):

FILE *f;

Pour ouvrir un fichier:

f = fopen (nomfichier, mode);

nomfichier une chaîne de caractères indiquant le nom du fichier et mode une chaîne de caractères indiquant le mode d'ouverture.

Pour un fichier texte, le mode peut être :

« r » ouverture en lecture. Vous pourrez lire le contenu du fichier, mais pas y écrire

« w » création d'un nouveau fichier (écrasement de l'ancien s'il existe déjà)

« a » ouverture en mode « ajout » Vous écrirez dans le fichier, en partant de la fin du fichier.

« r+ » ouverture en lecture/écriture

« a+ » ouverture en mode « ajout » en lecture/écriture Vous écrivez et lisez du texte à partir de la fin du fichier.

Les fichiers en langage C

Pour un fichier binaire, le mode peut être :

« **rb** » ouverture en lecture

« **wb** » création d'un nouveau fichier (écrasement de l'ancien s'il existe déjà)

« **ab** » ouverture en mode « ajout »

« **rb+** » / « **r+b** » ouverture en lecture/écriture

« **ab+** » / « **a+b** » ouverture en mode « ajout » en lecture/écriture

Les fichiers en langage C

Pour fermer un fichier:
fclose(f);

Pour savoir si on a dépassé la fin de fichier (en mode lecture) :
feof(f)

Retourne vrai (entier différent de 0) si on a tenté de lire au delà de la fin de fichier. Donc le schéma général pour lire le contenu d'un fichier est le suivant :

```
FILE *f ;
f = fopen(...) ;
<< opération_de_lecture>>
while ( ! feof( f ) ) {
// Traitement des données lus
... ;
<< opération_de_lecture>>
}
fclose( f )
```

Les fichiers en langage C

a) 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);

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

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

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

fseek(f, deplacement, 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

b) Lecture / Ecriture en mode texte

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

c = fgetc(f); lit un caractere ;

```
int caractereActuel = 0;
 caractereActuel = fgetc(fichier); // On lit le caractère
  printf("%c", caractereActuel); // On l'affiche
while (caractereActuel != EOF); // On continue tant que fgetc n'a pas retourné EOF (fin de fichier)
```

fputc(c, f); écrit un caractère dans le fichier (UN SEUL caractère à la fois) ;

```
FILE* fichier;
fputc('A', fichier);
```

fgets(buf, n, f); lit une chaîne ;

```
 fgets(chaine, TAILLE_MAX, fichier);
  printf("%s", chaine);
```

Les fichiers en langage C

Pour écrire une chaîne de caractères dans un fichier texte, on peut utiliser :
fputs(buf, f);

```
fputs("Quel age avez-vous ? "    ;fichier);
```

Pour effectuer une lecture formatée dans un fichier texte, on peut utiliser :

fscanf(f, format, &var1, &var2, …)

```
fscanf(fichier, "%d %d %d", &score[0], &score[1], &score[2]);

printf("Les meilleurs scores sont : %d, %d et %d", score[0], score[1], score[2]);
```

Pour effectuer une écriture formatée dans un fichier texte, on peut utiliser :
fprintf(f, format, exp1, exp2, …)

```
    printf("Quel age avez-vous ? ");

    scanf("%d", &age);

    fprintf(fichier, "%d ", age);
```

c) Exemples de programmes C

Dans le programme suivant, on construit un fichier binaire contenant un certain nombre d'enregistrements pour une application simple de gestion d'agenda téléphonique :

```
#include <stdio.h> //fichier1.c
struct parm { char nom[20];
    char tel[20]; };
int main() {
    struct parm p;
    FILE *output;
    char namefile[21];
    char rep = '\0';
     printf("Entrez le nom de fichier a creer: ");
    scanf("%21s", namefile);
   output = fopen(namefile, "wb");
    printf("Saisie des articles\n");
```

```
//fichier1.c (la suite)
do {
        printf("Entrez le nom: ");
        scanf("%s", p.nom);
        printf("Entrez le tel : ");
        scanf("%s", p.tel);
        fwrite(&p, 1, sizeof(p), output);
        printf("Voulez-vous saisir un autre article (O/N) ? ");
        // Read the newline character left in the input buffer
       while (getchar() != '\n');
        rep = getchar(); //  wait for new input
        printf("\n");
         } while (rep == 'O' || rep == 'o');
    fclose(output);
    return 0; }
```

c) Exemples de programmes C

Voici un exemple de programme qui parcours séquentiellement le fichier de données et affiche son contenu :

```
#include<stdio.h>  //fichier2.c
struct parm{ char nom[20]; char tel [20];  } ;
int main(){
          struct parm p;
          FILE *f;
          char namefile[21];
          int x,n,i;
          printf("entrez le nom de fichier a lister :");
          scanf("%21s",namefile);
          f=fopen(namefile,"rb");
while ( ! feof(f) ) {
   n = fread(&p, sizeof(p), 1, f);
   if ( n == 1 )
   printf("nom : %s \t tel : %s\n",  p.nom, p.tel);
}
fclose(f);
return 0 ; }
```

c) Exemples de programmes C

Le programme suivant montre un exemple d'accès direct (avec fseek) à un enregistrement de position donnée :

```
#include <stdio.h> //fichier3.c
struct parm { char nom[20]; char tel[20]; };
int main() {
    struct parm p;
    FILE *f;
    char namefile[21];
    int x, n, i;
    printf("Entrez le nom de fichier a manipuler : ");
    scanf("%21s", namefile);
    f = fopen(namefile, "rb");
    printf("Donnez le numéro de l'enregistrement a lire : ");
    scanf("%d", &i);
    fseek(f, (i - 1) * sizeof(p), SEEK_SET);
    n = fread(&p, sizeof(p), 1, f);
    if (n == 1)
    printf("Enreg num: %3d \t Nom : %s \t Tel : %s\n", i, p.nom, p.tel);
    else  printf("Erreur lors de la lecture de l'enregistrement.\n");
    fclose(f); return 0; }
```

c) Exemples de programmes C

Si on avait opté pour un fichier texte, on aurait eu le programme suivant :

```
#include <stdio.h> //fichier4.c
int main() {
struct Tenreg { char nom[20]; char tel[15]; } e;
FILE *f; // variable fichier
char nomf[30];

printf("\n Construction d'un fichier agenda telephonique\n\n");
printf("Donnez le nom du fichier à construire : ");
scanf(" %s", nomf);
f = fopen( nomf, "w" ); // creation du nouveau fichier en mode texte
if ( f == NULL ) {
printf("erreur lors de l'ouverture du fichier %s en mode w\n", nomf);
return 0;
}
```

```
//fichier4.c (la suite)
// insertion des enregistrements lus à la console
printf("donnez un nom et un tel (ou 0 0 pour terminer le programme) : ");
scanf(" %s %s", e.nom, e.tel);
while ( e.nom[0] != '0' ) {
// écriture formatée dans le fichier
fprintf(f, " %s , %s\n", e.nom, e.tel ); // un enregistrement par ligne
printf("donnez un nom et un tel (ou 0 0 pour terminer le programme) : ");
scanf(" %s %s", e.nom, e.tel);
}
fclose( f );
return 0;
}
```

c) Exemples de programmes C

Le programme suivant permet de lire séquentiellement le fichier ainsi construit et affiche les enregistrements à l'écran :

```
#include <stdio.h> //fichier5.c
int main() {
struct Tenreg {char nom[20]; char tel[15]; }e;
    FILE *f;
    char nomf[30];
    int n, i;
    printf("\n Affichage du contenu d'un fichier agenda telephonique \n");
    printf("Donnez le nom du fichier à lister : ");
    scanf(" %s", nomf);
    f = fopen(nomf, "r"); // ouverture du fichier texte
    if (f == NULL)
{ printf("Erreur lors de l'ouverture du fichier %s en mode r\n", nomf);
return 0; }
```

```
//fichier5.c (la suite )
// lecture des enregistrements depuis le fichier

// premiere version
   while ( ! feof(f) ) {
   fscanf(f, "%s , %s", e.nom, e.tel);
   printf("\t nom : %s \t tel : %s\n", e.nom, e.tel);  }

 // deuxième version
   // while (fscanf(f, "%s , %s", e.nom, e.tel) == 2) {
    // printf("\t Nom : %s \t Tel : %s\n", e.nom, e.tel);   }

fclose(f);

    return 0;
}
```