XI. Les fichiers

1. Généralités
2. Fonctions générales sur les flots
3. Fichiers binaires
4. Fichiers textes

1. Généralités

Propriétés :

- stockage de masse
- longue durée de vie
- mais temps d'accès plus important que pour la mémoire "vive"

Support : disque dur, disque SSD, clé USB, DVD, cloud …

Signes distinctifs :

- porte un nom connu du système d'exploitation
- extension précisant le type de codage

Type de stockage :

- données compressées (.zip, .jpg, .mov, .mp3, ...)
- données binaires brutes sous forme standard (.png, .raw, ...)
- données brutes sous forme spécifique (.xls, .dat, ...)
- données de type texte (.txt, .doc, .c, ...)

Qu'est-ce qu'un fichier ?

Sorte de « cahier » où l'on peut effectuer 5 opérations de bases

1. **ouverture** d'un fichier (*désigner le fichier de travail*)
2. **aller à** (*se positionner à tel endroit du fichier*)
3. **écriture** (*mémoire centrale* **->** *fichier*)
4. **lecture** (*fichier* **->** *mémoire centrale*)
5. **fermeture** d'un fichier.

Un fichier en C :

- ***flot*** = *suite d'octets*
- terminée par **`EOF`** (***End Of File***)

Structuré de 2 manières :

- **fichier texte** (suite d'octets "interprétable" par un éditeur de texte. Ex : *tp1.c*)
- **fichier binaire** (même codage qu'en mémoire centrale)
- distinction faite uniquement par les fcts de lecture/écriture

Équipé d'un **pointeur** (index de lecture/écriture)

–> où effectuer la prochaine lecture/écriture

2. Fonctions générales sur les flots

Définition d'un pointeur de fichier : **`FILE *`** *identificateur ;*

```
#include <stdio.h>
FILE  *f1, *f2;
```

Rôle du pointeur :

- associé à un fichier (à l'ouverture)
- désigne l'endroit de la prochaine opération de lecture/écriture.

Ouverture d'un flot

```
FILE *fopen(char nom[], char mode[])
```

On associe :

- un pointeur de fichier
- avec un nom de fichier (ex : `"jazz.mp3"`)

Exemple :

```
FILE *f;
f = fopen("jazz.mp3", "r");
```

Remarques :

- **`f`** vaut **`NULL`** si l'ouverture a échoué (*fichier absent,* ...)
- **`f`** pointe au début du fichier
- toutes les opérations seront faites en **`f`**

Mode d'ouverture : définit le type d'opérations autorisées

`"r"` Lecture dans un fichier **existant**

`"r+"` Idem **`"r"`** et écriture autorisée

`"w"` **Création** d'un fichier **vide** et écriture
Remarque : si le fichier existe déjà, il est réinitialisé à vide

`"w+"` Idem **`"w"`** et lecture autorisée

`"a"` Ouverture en écriture en conservant l'existant.
Le pointeur est en fin de fichier sans possibilité d'accès à la partie initiale.

`"a+"` Idem **`"a"`** avec lecture autorisée sur la partie ajoutée (sans possibilité d'accès à la partie initiale)

Fermeture d'un flot :

```
int fclose(FILE *flot)
```

```
fclose(f); /* libère f pour une autre utilisation */
```

Schéma général de traitement d'un fichier :

```
1. FILE *f ;
2. f = fopen("bidule.dat", "r+") ;
3. traiter(f) ; /* action +ou- complexe */
4. fclose(f) ;
```

3. Fichiers binaires

Même codage que dans la mémoire centrale.

Lecture/écriture : transfert "brut" de paquets d'octets via des variables

- de type de base (**`short`** (2 octets), **`float`** (4 octets), …)
- tableaux
- structures

On définit généralement un type de structure spécifique d'un fichier

- Fichier : une suite de structures (ou d'enregistrements)

Exemple pour un fichier répertoire téléphonique :

```
typedef struct
{ char nom[20] ;
  unsigned long tel ;
} typeEnreg ;
```

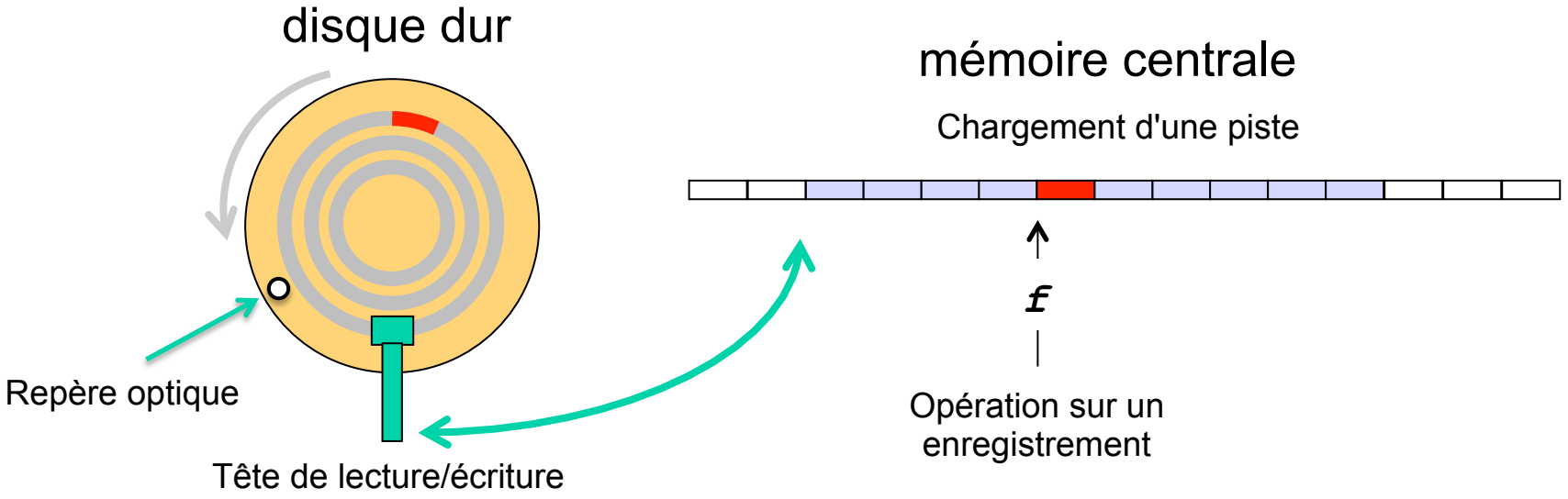
Fonction de lecture/écriture :

```
int fread(adr, taillePaquet, nombre, flot );
int fwrite(adr, taillePaquet, nombre, flot );
```

- rendent le nbre de paquets lus/écrits

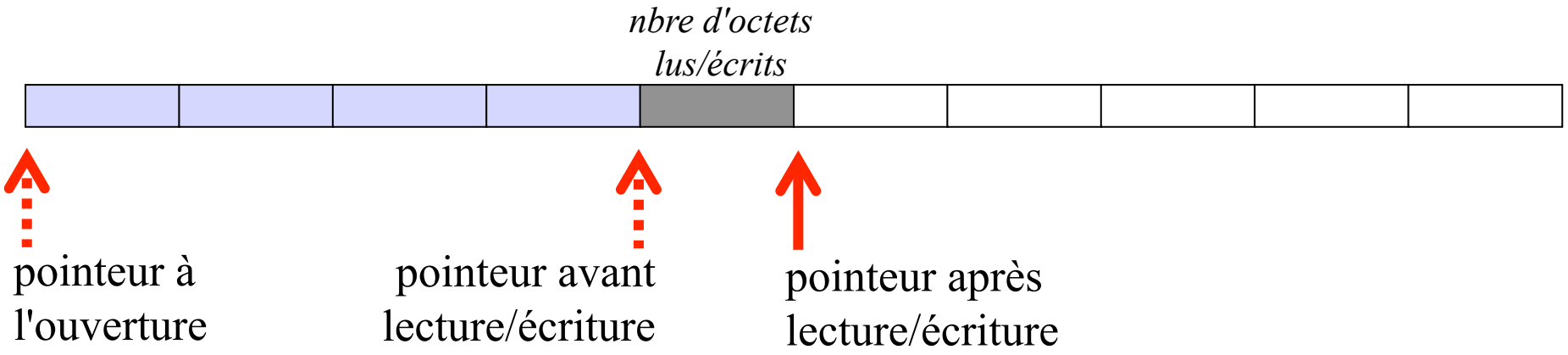
- Exemple d'appel :

```
typeEnreg e ;
fread(&e, sizeof(typeEnreg),1, f) ;
fwrite(&e, sizeof(typeEnreg),1, f) ;
```



Remarques :

- ces fct retournent le nombre de paquets effectivement transférés
- il y a une incrémentation automatique du pointeur de fichier



Exemples :

```
#define MAX 100
typeEnreg repertoire[MAX], e;
int nb;   // nb personnes dans le répertoire
FILE *f;
f = fopen("rep.dat", "r+");
```

- Lecture/écriture « en bloc » d'un petit fichier de taille connue

```
// s'assurer qu'on est au début du fichier
fread(repertoire, sizeof(typeEnreg), MAX, f);
//  le pointeur f est en fin de fichier
```

- Lecture/écriture « séquentielle » des enregistrements

```
// s'assurer qu'on est au début du fichier
int i;
for (i=0; i<MAX ; i++)
   fread(&repertoire[i], sizeof(typeEnreg),1, f);
//  le pointeur f est en fin de fichier
```

Exemples :

```
#define MAX 100
typeEnreg repertoire[MAX], e;
int nb;   // nb personnes dans le répertoire
FILE *f;
f = fopen("rep.dat", "r+");
```

- Lire un fichier de taille inconnue

```
nb = fread(repertoire, sizeof(typeEnreg), MAX, f);
```

- Lire séquentiellement un fichier de taille inconnue

```
nb = 0;
while(fread(&repertoire[nb],
            sizeof(typeEnreg),1, f) == 1)
{ nb++;
  faire un traitement sur repertoire[nb-1]
}
```

4. Positionnement

Rappel : chaque lecture/écriture incrémente le pointeur de fichier

On peut déplacer le pointeur de fichier :

- **`rewind(`***`f`***`)`** *(positionne au début)*
- **`fseek(`***`f, sautEnOctets, origine`***`)`**

  avec pour origine :
  - **`SEEK_CUR`** (position courante)
  - **`SEEK_END`** (fin)
  - **`SEEK_SET`** (début)

Remarque : dans le cas de mode "a", le pointeur ne peut accéder à la zone des données initiales (protection de l'existant)

Accès "aléatoire" sur de gros fichiers :

On dispose d'un fichier d'étudiants structuré en enregistrements suivant le modèle ci-dessous :

```
typedef struct
{ long numEtudiant;
  char nom[50];
  float notes[10];
} typeEtud;
```

On souhaite écrire le programme qui permet de saisir les notes d'un étudiant déjà inscrit.

Algorithme :

- lire le numéro étudiant
- rechercher sa position dans le fichier
- charger sa fiche (*lire*) en mémoire centrale
- mettre à jour la fiche (saisie ses 10 notes au clavier)
- recopier sa fiche ainsi modifiée au même emplacement dans le fichier

Programme :

```
typedef struct
{ long numEtudiant; char nom[50]; float notes[10];
} typeEtud;

#define MAX 100000          // 70.000 étudiants à AMU
int nbEtud, table[MAX];     // position <-> n° étudiant

FILE *f;

/* initialiser la table d'acces */
void initTableAcces(void);

/* rechercher la position d'un étudiant dans le fichier (-1 si abs) */
int acces(int numEtud);

/* remplir le champ notes d'une structure Etudiant */
void saisirNotes(int numEtud);

void main(void)
{ f = fopen("etud_AMU.dat", "r+");
  initTableAcces();
  ...
  fclose(f);
}
```

```
int nbEtud;
int table[1000]; /* position <-> n° étudiant */

void initTableAcces(void);
{ typeEtud e;
  rewind(f);
  nbEtud = 0;
  while(fread(&e, sizeof(e), 1, f)==1)
  { table[nbEtud] = e.numEtudiant;
    nbEtud++;
  }
}

int acces(int numero)
{ int i=0;
  while (i<nbEtud)
  { if (table[i]==numero) return i;
    i++;
  }
  return -1; /* numero inconnu */
}
```

```
void saisirNotes(int numEtud)
{ int position, i ;
  typeEtud e ;
  position = acces(numEtud) ;
  if (position<0)
  { printf("etudiant inexistant\n") ; return }
  /* lecture des données de l'étudiant */
  fseek(f, position*sizeof(typeEtud), SEEK_SET) ;
  fread(&e, sizeof(typeEtud), 1, f) ;
  /* saisie des 10 notes au clavier */
  for (i=0 ; i<10 ; i++)
  { printf("note[%d] ?\n", i);
    scanf("%f", &(e.note[i]));
  }
  /* sauvegarde de la modification */
  fseek(f, -sizeof(typeEtud), SEEK_CUR) ;
  fwrite(&e, sizeof(typeEtud), 1, f) ;
}
```

Rappel sur les fichiers

Ouverture/fermeture d'un fichier

```
FILE *f;
f = fopen("jazz.mp3", "r+");
fclose(f) ;
```

3 modes : `r, w, a`

Fichiers binaires

```
typeEnreg e ;
fread(&e, sizeof(typeEnreg), 1, f) ;
fwrite(&e, sizeof(typeEnreg), 1, f) ;
```

Déplacer le pointeur

- Incrémentation automatique lors des lectures/écritures
- Repositionnement

```
rewind(f) ;  // repositionnement au début
fseek(f, nbre d'octets, SEEK_CUR) ;
   SEEK_SET, SEEK_CUR, SEEK_END
```

5. Fichiers textes

- L'information dans un fichier texte est « lisible » (i.e. éditable).
- **`stdin`** et **`stdout`** sont deux "fichiers textes" bien connus…

- Fcts de lecture/écriture : Binaire **<->** forme lisible (suite de caract.)
- Elles sont proches des fcts d'entrées/sorties :

`int fscanf(`*`flot, format, adr1,..., adrn`***`)`**
  - retourne le nombre de variables effectivement lues
  - ou **`EOF`** si fin de fichier

```
fscanf(f,"%d%c", &i, &c);
```

`int fprintf(`*`flot, format, exp1,..., exp_n`***`)`**
  - écrit dans le flot les résultats des n expressions

```
fprintf(f,"%d%c", i+1, c);
```

`int fgetc(FILE *flot)` lecture du caractère suivant

- rend **`EOF`** si fin de fichier

`Ex : c = fgetc(f);`

`int fputc(int caractere, FILE *flot)`

- rend **`EOF`** si une erreur se produit

`Ex : fputc(c, f);`

`char *fgets(char *s, int n, FILE *flot)`

`Ex : fgets(s,80,stdin);` *rend NULL si erreur*