SI4 - Bases de la programmation

Les fichiers

Accès séquentiel

Accès direct

OSI

Introduction

* Les fichiers font partie des structures de stockage.
* L'organisation d'un fichier peut être séquentielle, directe, OSI.
* Les données d'un fichier peuvent être sous forme texte, ou sous forme d'une représentation interne (typage).
* Les opérations sur un fichier sont :
  * OUVRIR(canal)
  * LIRE(enregistrement, canal)
  * ECRIRE(enregistrement, canal)
  * FERMER(canal)

1. Structure de données

* Un fichier utilise des données structurées.
* La taille d'un fichier peut changer de manière dynamique.
* Les données sont rangées suivant une structure :

```
TYPE
STRUCTURE personne
    Age : entier
    Nom : chaîne de caractères[255]
FIN STRUCTURE
VAR
unePersonne : personne
…
unePersonne.age <- 12
UnePersonne.nom <- ''TOTO''
```

1. Structure de données

* En langage C :

```
typedef struct personne
{
        int age;
        char nom[255];
};
personne unePersonne;
unePersonne.age = 12;
strcpy(unePersonne.nom,”TOTO”);
```

2. Ouverture d'un fichier

* Pour ouvrir un fichier, il faut son mode d'accès, son emplacement, son nom.

```
TYPE
        STRUCTURE personne
VAR
        fichPersonne : fichier séquentiel de personne
        unePersonne : personne
…
OUVRIR fichPersonne en lecture // écriture, lecture/écriture
LIRE (fichPersonne, unePersonne)[5]
```

2. Ouverture d'un fichier

* En langage C, l'ouverture se fait simplement :

```
typedef struct personne
{… };
personne unePersonne;
FILE * fichPersonne;
fichPersonne = fopen(fileName, ”w”); // ‘’r”, ‘’a”
fread(&unePersonne,sizeof(personne),1,fichPersonne);
```

3. Gestion des enregistrements

* Un fichier contient en général plusieurs enregistrements.
* Pour cela, une boucle permet de parcourir les enregistrements jusqu'à la fin. Attention au TANT QUE !

```
…
LIRE(fichPersonne, unePersonne)
TANT QUE !EOF(fichPersonne)
        //exploitation des données
        LIRE(fichPersonne, unePersonne)
FIN TANT QUE
…
```

3. Gestion des enregistrements

```
fread(&unePersonne,sizeof(personne),1,fichPersonne);
while (!feof(fichPersonne))
{
        // exploitation des données
        fread(&unePersonne,sizeof(personne),
1,fichPersonne);
}
```

4. Gestion en mémoire centrale

* Attention, charger un fichier en mémoire pour l'exploiter.
* Utiliser des tableaux pour stocker les enregistrements.

```
VAR
i : entier
tabPersonne : tableau de [1..100] de personne
…
tabPersonne[i].age <- 15
tabPersonne[i].nom <- ''TITI''
```

4. Gestion en mémoire centrale

* En langage C :

```
int i;
personne tabPersonne[100];

tabPersonne[i].age = 15;
strcpy(tabPersonne[i].nom,"TITI");
```

5. Lecture

* Lire un fichier signifie prendre les données du disque dur et les mettre dans des variables en mémoire centrale.

* LIRE(fichPersonne, unePersonne)
* fread(&unePersonne,sizeof(personne),1,fichPersonne);

* La variable est ici, unePersonne

6. Ecriture

* Cette opération consiste à prendre des données de la mémoire centrale pour les enregistrer sur le disque dur.

* ECRIRE(fichPersonne, unePersonne)
* fwrite(&unePersonne,sizeof(personne), 1,fichPersonne);

* Aucun changement avec la lecture !

7. Fermeture

* Un fichier ouvert doit être fermé après utilisation.

* FERMER(fichPersonne)
* fclose(fichPersonne);
* fcloseall(); // tous les fichiers

8. Synthèse

* Les fichiers nécessitent :
  * Des structures pour déclarer des types.
  * Des variables pour faire des enregistrements en mémoire, associées aux structures.
  * Un descripteur de canal, associé à un fichier disque.
  * Des opérations de lecture et d'écriture.
  * Des tableaux pour accélérer les traitements.