Algorithmique et Programmation : Chap 7 : Les fichiers

E. M. Souidi

Faculté des Sciences - Rabat
SVI4 –STU4 2013-14

Introduction

Jusqu'à présent, nos programmes ont lu leurs données au clavier ou dans le programme lui même et ils ont sorti leurs résultats à l'écran.

Introduction

Jusqu'à présent, nos programmes ont lu leurs données au clavier ou dans le programme lui même et ils ont sorti leurs résultats à l'écran.
Pour le traitement des notes des étudiants, on ne peut ni inclure les notes dans le programme ni les saisir à chaque utilisation du programme.

La solution est de stocker les données à traiter dans des fichiers.

La solution est de stocker les données à traiter dans des fichiers.
Dans la plupart des travaux de programmation, on est obligé d'aller lire ou écrire dans des fichiers.

La solution est de stocker les données à traiter dans des fichiers.
Dans la plupart des travaux de programmation, on est obligé d'aller lire ou écrire dans des fichiers.
Nous allons voir dans ce chapitre, comment créer, lire et modifier des fichiers sur les périphériques disponibles.

Définition

Un fichier (en anglais file) est une collection d'octets (suite de 0 et de 1) représentant des informations. Ces octets sont organisés en structures hiérarchique afin de faciliter les opérations sur les fichiers : ouverture, écriture, lecture, ferméture.

Deux types de fichiers

Fichier texte : c'est un fichier dont le contenu est uniquement une suite de caractères informatiques : lettres, chiffres ou signes de ponctuation.

L'espace et le retour à la ligne sont considérés comme des caractères informatiques.

L'espace et le retour à la ligne sont considérés comme des caractères informatiques.
Les fichiers texte peuvent être édités avec des éditeurs de texte (edit, bloc Notes, emacs, vi etc) et affichés de manière lisible à l'écran.

Fichier binaire : c'est un fichier "non texte", dans le sens où certains octets contenus dans le fichier ne représentent pas des caractères. Par exempe : soulignement, tableau, son, image etc.

Un fichier binaire contient des données non textuelles. Il est constitué d'une suite d'octets auxquels seuls des programmes adaptés peuvent donner un sens. Exemples de fichiers binaires : fichier exécutable d'un programme, fichiers image, son, vidéo, etc

Les fichiers permettent de stocker (enregistrer) des informations sur une mémoire externe (disquette, disque dur, clef USB ...) distincte de la mémoire centrale ou mémoire vive (RAM).

Les fichiers permettent de stocker (enregistrer) des informations sur une mémoire externe (disquette, disque dur, clef USB ...) distincte de la mémoire centrale ou mémoire vive (RAM).
Le problème avec le stockage externe est la lenteur d'accès aux données.

Mémoire tampon

La mémoire tampon est une zone de la mémoire centrale de la machine réservée à un ou plusieurs enregistrements du fichier.

Mémoire tampon

La mémoire tampon est une zone de la mémoire centrale de la machine réservée à un ou plusieurs enregistrements du fichier. Pour des raisons d'efficacité, l'accès à un fichier se fait par l'intermédiaire d'une mémoire tampon (buffer en anglais).

L'utilisation de la mémoire tampon a pour effet de réduire le nombre d'accès à la périphérie d'une part et le nombre de mouvements de la tête de lecture/écriture d'autre part.

Ouverture de fichier

Avant de créer ou de lire un fichier, nous devons informer le système de cette intention pour qu'il puisse réserver une mémoire tompon nécessaire à l'accès à ce fichier.

Ouverture de fichier

Avant de créer ou de lire un fichier, nous devons informer le système de cette intention pour qu'il puisse réserver une mémoire tompon nécessaire à l'accès à ce fichier.
Cette opération s'appelle ouverture de fichier.

Fermeture de fichier

Après avoir terminé la manipulation (écriture, lecture, modification) du fichier, nous devons vider la mémoire tampon et libérer l'espace en mémoire que nous avons occupé pendant le traitement. Cette opération s'appelle fermeture du fichier.

Types d'accès aux fichiers

On distingue : 1) L'accès séquentiel : avec ce mode d'accès, on ne peut accéder à une information sans lecture compète de toutes les informations depuis le début de fichier.

Types d'accès aux fichiers

On distingue : 1) L'accès séquentiel : avec ce mode d'accès, on ne peut accéder à une information sans lecture compète de toutes les informations depuis le début de fichier.
Les applications utilisant l'accès séquentiel au fichiers s'avèrent relativement lentes.

L'accès séquentiel est réservé aux applications pour lequelles il est indispensable de lire tout le fichier du début à la fin.

Par exemple : affichage d'une image.

Par exemple : affichage d'une image.
Cependant, la recherche d'une adresse dans un annuaire, ou un mot dans un dictionnaire électronique ne nécessiste pas la lecture du fichier du début à la fin.

Par exemple : affichage d'une image.
Cependant, la recherche d'une adresse dans un annuaire, ou un mot dans un dictionnaire électronique ne nécessiste pas la lecture du fichier du début à la fin.
Dans de tels cas, il faut un autre mode d'accès.

2) L'accès direct : Ce mode d'accès suppose une organisation particulière de l'information.

2) L'accès direct : Ce mode d'accès suppose une organisation particulière de l'information.
Un fichier à accès directe est structuré en enregistrements (lignes) et chaque enregistrement est composé de champs.

2) L'accès direct : Ce mode d'accès suppose une organisation particulière de l'information.
Un fichier à accès directe est structuré en enregistrements (lignes) et chaque enregistrement est composé de champs. De plus le même champs doit posséder dans chaque enregistrement la même longueur en octets.

Exemple : Nom, 20 car Prénom, 15, Ville, 12 <– Champs 1 2 3

Chaque enregistrement est repéré par son numéro d'enregistrement.

Chaque enregistrement est repéré par son numéro d'enregistrement.
Chaque champs est séparé du suivant par un séparateur de champs (espace, virgule etc..)

Chaque enregistrement est repéré par son numéro d'enregistrement.
Chaque champs est séparé du suivant par un séparateur de champs (espace, virgule etc..)
Chaque enregistrement est séparé du suivant par un séparateur d'enregistrement qui est généralement séparé de deux octets de codes ASCII 10 et 13.

Donc avec l'accès directe, on peut accéder directement à l'enregistrement de son choix, en précisant le numéro de cet enregistrement.

3) L'accès indexé : pour simplifier, il combine la rapidité de l'accès direct et la simplicité de l'accès séquentiel (en restant toutefois plus compliqué).

3) L'accès indexé : pour simplifier, il combine la rapidité de l'accès direct et la simplicité de l'accès séquentiel (en restant toutefois plus compliqué).
Il est particulièrement adapté au traitement de gros fichiers, comme les bases de données importantes.

Dans ce cours on se limitera au type de base : fichier texte en accès séquentiel.

Dans ce cours on se limitera au type de base : fichier texte en accès séquentiel.
Si l'on veut travailler sur un fichier, la première chose à faire est de l'ouvrir en précisant ce qu'on va en faire : lire, écrire ou ajouter.

Structure des enregistrements

Dans le cas d'un fichier texte structuré en enregistrement, il y a deux façons de structurer les données au sein d'un fichier texte :

Structure des enregistrements

Dans le cas d'un fichier texte structuré en enregistrement, il y a deux façons de structurer les données au sein d'un fichier texte :
la délimitation et les champs de largeur fixe.

Structure délimitée

Elle utilise un caractère spécial, appelé caractère de délimitation, qui permet de repérer quand finit un champ et quand commence le suivant.

Structure délimitée

Elle utilise un caractère spécial, appelé caractère de délimitation, qui permet de repérer quand finit un champ et quand commence le suivant.
Ce caractère de délimitation doit être strictement interdit à l'intérieur de chaque champ, faute de quoi la structure devient ambiguë.

Exemple : structure avec champs de largeur fixe.

Exemple : structure avec champs de largeur fixe.

Structure à champs de largeur fixe

Il n'y a pas de caractère de délimitation, mais on sait que les n premiers caractères de chaque ligne stockent le nom, les m suivants le prénom, etc. Cela impose de ne pas saisir un renseignement plus long que le champ prévu pour l'accueillir.

Avantage de la structure délimitée

est son faible encombrement en place mémoire ; il n'y a aucun espace perdu, et un fichier texte codé de cette manière occupe le minimum de place possible.

Mais a un inconvénient majeur :

Lenteur de la lecture. En effet, chaque fois que l'on récupère une ligne dans le fichier, il faut alors parcourir un par un tous les caractères pour repérer chaque occurrence du caractère de séparation avant de pouvoir découper cette ligne en différents champs.

Structure à champs de largeur fixe gaspille de la place mémoire, puisque le fichier est plein d'espaces (blancs). Mais la récupération des différents champs est très rapide. Lorsqu'on récupère une ligne, il suffit de la découper en différentes chaînes de longueur prédéfinie, et le tour est joué.

Quelle structure utiliser ?

A l'époque où la place mémoire coûtait cher et limitée, la structure délimitée était souvent privilégiée.

Quelle structure utiliser ?

A l'époque où la place mémoire coûtait cher et limitée, la structure délimitée était souvent privilégiée.
Mais depuis bien des années, la quasi-totalité des logiciels et des programmeurs optent pour la structure en champs de largeur fixe.

Ouverture de fichiers

Au besoin, un programme ouvre un fichier pour lire, enregistrer ou traiter les informations qui s'y trouvent.

Ouverture de fichiers

Au besoin, un programme ouvre un fichier pour lire, enregistrer ou traiter les informations qui s'y trouvent.
Le problème avec le stockage externe est la lenteur d'accès aux données.

Ouverture de fichiers

Au besoin, un programme ouvre un fichier pour lire, enregistrer ou traiter les informations qui s'y trouvent.
Le problème avec le stockage externe est la lenteur d'accès aux données.
Nous appelons structure de fichier l'organisation imposée à ce fichier afin de faciliter son traitement).

Types d'ouvertures

Si on ouvre un fichier pour lecture, on ne pourra que lire les informations qu'il contient, sans pouvoir les modifier.

Types d'ouvertures

Si on ouvre un fichier pour lecture, on ne pourra que lire les informations qu'il contient, sans pouvoir les modifier.
Si on ouvre un fichier pour écriture, on pourra écrire dedans tout ce que l'on veut.

Mais s'il y avait des informations, elles seront écrasées, sans pouvoir les réccupérer.

Mais s'il y avait des informations, elles seront écrasées, sans pouvoir les réccupérer.
Si on ouvre un fichier pour ajout, on ne peut ni lire, ni modifier les informations existantes. Mais on pourra, ajouter de nouvelles lignes (càd enregistrements).

En pseudo-code

Pour ouvrir un fichier

En pseudo-code

Pour ouvrir un fichier

Ouvrir (nomfichier , lecture)

En pseudo-code

Pour ouvrir un fichier
Ouvrir (nomfichier , lecture)
Ouvrir (nomfichier, écriture)

En pseudo-code

Pour ouvrir un fichier

Ouvrir (nomfichier , lecture)

Ouvrir (nomfichier, écriture)

Ouvrir (nomfichier, écriture)

En pseudo-code

Pour ouvrir un fichier
Ouvrir (nomfichier , lecture)
Ouvrir (nomfichier, écriture)
Ouvrir (nomfichier, écriture)
Et pour fermer un fichier

En pseudo-code

Pour ouvrir un fichier
Ouvrir (nomfichier , lecture)
Ouvrir (nomfichier, écriture)
Ouvrir (nomfichier, écriture)
Et pour fermer un fichier
Fermer (nomfichier)

En pseudo-code

Pour ouvrir un fichier
Ouvrir (nomfichier , lecture)
Ouvrir (nomfichier, écriture)
Ouvrir (nomfichier, écriture)
Et pour fermer un fichier
Fermer (nomfichier)

En pratique, il faut contrôler si l'ouverture d'un fichier a été accomplie avec succès avant de continuer les traitements.

Les fichiers en Python

La fonction prédéfinie open() permet de créer un "objet-fichier", pour pouvoir lire et écrire dans un fichier.

Les fichiers en Python

La fonction prédéfinie open() permet de créer un "objet-fichier", pour pouvoir lire et écrire dans un fichier.
"objet-fichier" est un intermédiaire qui permet d'accéder aux fichiers.

Les fichiers en Python

La fonction prédéfinie open() permet de créer un "objet-fichier", pour pouvoir lire et écrire dans un fichier.
"objet-fichier" est un intermédiaire qui permet d'accéder aux fichiers.
La fonction open() prend deux arguments, le premier est le nom de fichier à ouvrir, et le deuxième argument est le mode d'ouverture.

Il y a trois modes :

Il y a trois modes :

1. a=append : ajout

Il y a trois modes :

1. a=append : ajout
2. r=read : lire

Il y a trois modes :

1. a=append : ajout
2. r=read : lire
3. w=write : écrire

Il y a trois modes :

1. a=append : ajout
2. r=read : lire
3. w=write : écrire

Pour créer "l'objet fichier" en mode 'ajout' (a=append)

Pour créer "l'objet fichier" en mode 'ajout' (a=append)

```
>>> objet=open('monfichier','a')
```

Pour créer "l'objet fichier" en mode 'ajout' (a=append)

```
>>> objet=open('monfichier','a')
```

Ecriture séquentielle dans un fichier

La méthode write() permet d'écrire dans le fichier

Ecriture séquentielle dans un fichier

La méthode write() permet d'écrire dans le fichier

```
>>> objet.write('me voila écrire dans monfichier !')
```

Ecriture séquentielle dans un fichier

La méthode write() permet d'écrire dans le fichier

```
>>> objet.write('me voila écrire dans monfichier !')
>>> objet.write('et je continu à écrire')
```

Ecriture séquentielle dans un fichier

La méthode write() permet d'écrire dans le fichier

```
>>> objet.write('me voila écrire dans monfichier !')
>>> objet.write('et je continu à écrire')
```

Chaque nouvel appel de la fonction write() continue l'écriture à la fin du fichier.

La méthode close() ferme le fichier.

La méthode close() ferme le fichier.

```
>>> objet.close()
```

La méthode close() ferme le fichier.

```
>>> objet.close()
```

Remarque : Ne pas confondre le nom du fichier "monfichier" et "objet" qui est le nom de "l'objet-fichier" qui donne accès à "monfichier". Pour ouvrir un fichier

Remarque : Ne pas confondre le nom du fichier "monfichier" et "objet" qui est le nom de "l'objet-fichier" qui donne accès à "monfichier". Pour ouvrir un fichier

```
>>> Fichier1 = open("fichier.dat","w")
```

Remarque : Ne pas confondre le nom du fichier "monfichier" et "objet" qui est le nom de "l'objet-fichier" qui donne accès à "monfichier". Pour ouvrir un fichier

```
>>> Fichier1 = open("fichier.dat","w")
```

w indique "write"

Remarque : Ne pas confondre le nom du fichier "monfichier" et "objet" qui est le nom de "l'objet-fichier" qui donne accès à "monfichier". Pour ouvrir un fichier

```
>>> Fichier1 = open("fichier.dat","w")
```

w indique "write"

```
>>> Fichier1.write("Ceci est un test \n")
```

Remarque : Ne pas confondre le nom du fichier "monfichier" et "objet" qui est le nom de "l'objet-fichier" qui donne accès à "monfichier". Pour ouvrir un fichier

```
>>> Fichier1 = open("fichier.dat","w")
```

w indique "write"

```
>>> Fichier1.write("Ceci est un test \n")
```

Notez le "\n" qui indique un retour à la ligne.

On peut aussi écrire plusieurs lignes à la fois >>> liste = ['Première ligne de texte \n', "Deuxième ligne de texte \n"]

On peut aussi écrire plusieurs lignes à la fois >>> liste = ['Première ligne de texte \n', "Deuxième ligne de texte \n"]
Notez l'utilisation des guillemets ou des apostrophes

On peut aussi écrire plusieurs lignes à la fois >>> liste = ['Première ligne de texte \n', "Deuxième ligne de texte \n"]
Notez l'utilisation des guillemets ou des apostrophes
>>> Fichier1.writelines(liste)

On peut aussi écrire plusieurs lignes à la fois >>> liste = ['Première ligne de texte \n', "Deuxième ligne de texte \n"]
Notez l'utilisation des guillemets ou des apostrophes
>>> Fichier1.writelines(liste)
Finalement, on ferme le fichier

On peut aussi écrire plusieurs lignes à la fois >>> liste = ['Première ligne de texte \n', "Deuxième ligne de texte \n"]
Notez l'utilisation des guillemets ou des apostrophes
>>> Fichier1.writelines(liste)
Finalement, on ferme le fichier
>>> Fichier1.close()

Lecture d'un fichier

On ouvre d'abord le fichier

Lecture d'un fichier

On ouvre d'abord le fichier

```
>>> Fichier1 = open("fichier.dat","r")
```

Lecture d'un fichier

On ouvre d'abord le fichier

```
>>> Fichier1 = open("fichier.dat","r")
>>> A=Fichier1.read()
```

Lecture d'un fichier

On ouvre d'abord le fichier

```
>>> Fichier1 = open("fichier.dat","r")
>>> A=Fichier1.read()
```

read() permet de lire la totalité du fichier

Lecture d'un fichier

On ouvre d'abord le fichier

```
>>> Fichier1 = open("fichier.dat","r")
>>> A=Fichier1.read()
```

read() permet de lire la totalité du fichier

```
>>> print A
```

Lecture d'un fichier

On ouvre d'abord le fichier

```
>>> Fichier1 = open("fichier.dat","r")
>>> A=Fichier1.read()
```

read() permet de lire la totalité du fichier

```
>>> print A
```

read(*n*) permet de lire les *n* caractères à partir de la position déjà atteinte dans le fichier On ferme le fichier à la fin.

read(*n*) permet de lire les *n* caractères à partir de la position déjà atteinte dans le fichier On ferme le fichier à la fin.

```
>>> Fichier1.close()
```

read(*n*) permet de lire les *n* caractères à partir de la position déjà atteinte dans le fichier On ferme le fichier à la fin.

```
>>> Fichier1.close()
```

La méthode readlines() permet de lire l'intégralité d'un fichier en une seule instruction.

La méthode readlines() permet de lire l'intégralité d'un fichier en une seule instruction. C'est valable pour les petits fichiers, les gros risquent de saturer la mémoire de l'ordinateur.

La méthode readlines() permet de lire l'intégralité d'un fichier en une seule instruction. C'est valable pour les petits fichiers, les gros risquent de saturer la mémoire de l'ordinateur.
Dans ce dernier cas on utilise la fonction readline() dans une boucle.

La méthode readlines() permet de lire l'intégralité d'un fichier en une seule instruction. C'est valable pour les petits fichiers, les gros risquent de saturer la mémoire de l'ordinateur.
Dans ce dernier cas on utilise la fonction readline() dans une boucle.
readlines() renvoie une liste. readline() renvoie une chaine de caractères.