Les fichiers en algorithmique

b.aghoutane@umi.ac.ma

Les fichiers

- Un **fichier** (*anglais : file)* est un **ensemble structuré de données de même type**.
- Il sert à **stocker** des informations de **manière permanente** afin de permettre une réutilisation ultérieure des informations qu'il contient.
- À l'encontre des **variables** qui sont stockés dans des **mémoires vives**, les **fichiers**, eux sont stockés sur des **mémoires non volatiles** (*disque dur, flash disque, DVD, ...*).
- Un **fichier** peut contenir des caractères (*fichier textes*), des valeurs (*numériques*), des enregistrements (*nom, prénom, numéro et l'email*) , des programmes (*python, C, ...*), …
- Un fichier se caractérise par quelques attributs: **nom**, **date de création, date dernière de modification**, **propriétaire**, et sa **catégorie** (*binaire, texte*).
- Les fichiers possèdent un **format** qui indique comment sont organisées les informations. L'extension suffixe du nom du fichier indique le format (par exemple **.txt, .pdf, .py**).

Les catégories des fichiers

Il existe **deux catégories** de fichiers :

1. **Fichiers textes :**
   - Les fichiers textes stockent des données sont écrites sous forme de texte.
   - On peut facilement créer des fichiers texte à l'aide des éditeurs de texte (*Bloc notes, MS word,…*).
   - Ils peuvent contenir le même genre d'information sur chaque ligne. Ces lignes sont appelées des **enregistrements**.
   - Lorsqu'on ouvre ces fichiers, on peut voir tout le contenu du fichier sous forme de texte brut.
   - On peut facilement modifier ou supprimer le contenu.
2. **Fichiers binaires :**
   - Les fichiers binaires stockent les données sous forme binaire (*0 et 1*).
   - Ils peuvent contenir une grande quantité de données,
   - ils ne sont pas lisibles facilement et offrent une meilleure sécurité que les fichiers textes.
   - Ce sont des fichiers qui ne possèdent pas de structure de lignes (d'enregistrement).

   Exemple : fichier son, image, vidéo, programme exécutable, etc.

29

Organisation : Fichiers texte (exemple)

Un **fichier** peut être structuré en enregistrement (par exemple : *Fiches des étudiants*).

- Ce fichier est destiné à mémoriser les **coordonnées d'un certain nombre d'étudiants**.
- Pour **chaque étudiant**, il faudra noter le **nom**, **prénom**, **numéro** et **l'email**.
- Il paraît plus simple de stocker un étudiant par ligne du fichier (*par enregistrement*).
- Autrement dit, une ligne ne contient que les informations concernant un étudiant.
- Un fichier ainsi codé sous forme d'enregistrements est appelé un **fichier texte**.
- Entre chaque enregistrement, sont stockés les octets relatifs aux caractères signifiant un retour au début de la ligne suivante.

30

Structures des enregistrements fichiers

Les **fichiers** peuvent être structurés en **enregistrements**.

Il y a deux grandes possibilités pour structurer ces enregistrements :

1. **La structure n°1 est dite délimitée :** Elle utilise un **caractère spécial**, appelé caractère de séparation ou délimitation, qui permet de repérer la fin d'un champ et le début d'un autre.

Exemple :

- "Hassnaoui"; "Mouad"; 0123; "h.mouad@yahoo.fr"
- "Idrissi"; "Jalal"; 0456; "i.jalal@gmail.com"

- **La structure n°2 est dite à champs de largeur fixe :** il n'y a pas de délimiteurs. Chaque **champ a une longueur prédéfinie** et occupe toute cette longueur, sinon il est complété par des espaces.

Exemple :

• Hassnaoui	Mouad	0123	h.mouad@yahoo.fr
• Idrissi	Jalal	0456	i.jalal@gmail.com

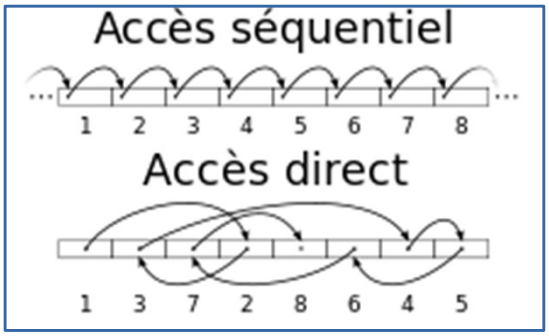
31

Types d'accès aux fichiers

Le **type d'accès** est la manière avec laquelle la machine **cherchera les informations** incluses dans le fichier. Le curseur indique la position de la prochaine lecture ou écriture.

- **L'accès séquentiel** : Commence au début du fichier. Pour lire une information particulière, il faut lire toutes les informations précédentes. Dans le cas d'un fichier texte, cela signifie qu'on lit le fichier ligne par ligne (*enregistrement par enregistrement*).
- **L'accès direct** : On peut accéder directement à l'information désirée (*enregistrement*), en précisant le numéro d'emplacement (*numéro d'ordre*) de cette information.

  ➔ **L'accès direct** est préférable à l'accès séquentiel lorsqu'une application doit écrire, lire ou modifier un seul élément d'un large ensemble.



Les **exemples** suivants concernent le type de base :

fichier texte en accès séquentiel

32

Les modes d'ouvertures des fichiers

- Si on veut travailler sur un fichier, la première chose à faire après la **déclaration** est de l'**ouvrir**.
- Lorsqu'on ouvre un fichier, il faut **exprimer** ce qu'on va en faire: **lire, écrire** ou **ajouter.**

Il existe **quatre modes d'ouverture** d'un fichier :

1. **Lecture (position du curseur en début de fichier) :**
   - On peut uniquement accéder et récupérer les informations qu'il contient,
   - On ne peut pas modifier son contenu.
2. **Ecriture (position du curseur en début de fichier) :**
   - On peut écrire toutes les informations que l'on veut.
   - Les informations précédentes, si elles existent, seront **intégralement écrasées** .
   - On ne pourra pas accéder aux informations qui existaient précédemment.
3. **Lecture/écriture (position du curseur en début de fichier) :**
   - Les données existantes sont préservées, et il est possible de modifier le contenu du fichier.
4. **Ajout (position du curseur en fin de fichier) :**
   - On ne peut ni lire, ni modifier les informations existantes.
   - Mais, on peut ajouter de nouvelles lignes (*nouveaux enregistrement)* à la fin du fichier.

33

Déclaration et ouvertures des fichiers séquentiels

La **structure fichier** se déclare comme une **variable** avec un type prédéfini :

```
Variable   nom_fichier : Fichier
```

Pour **ouvrir un fichier texte**, on écrira par exemple :

```
nom_fichier ← "Toto.txt"
Ouvrir (nom_fichier) en Lecture
```

ou bien

```
Ouvrir (nom_fichier, mode d'ouverture)
```

34

Lecture, écriture et fermeture des fichiers

Lire\ écrire dans un fichier ouvert :

Lire_Fichier (nom_fichier, liste des variables**)**

Ecrire_Fichier (nom_fichier, liste des variables**)**

Fermer le fichier :

Fermer (nom_fichier)

Tester la fin du fichier :

EOF (nom_fichier) # ***EOF****: End Of File*

35

Exemple d'ajout dans un fichier

```
Algorithme ExempleFichier
Variable
        Truc : chaîne Caractère
        Fich : Fichier
Début
        Fich ← "Exemple.txt"
        Truc ← "Fonfec"; "Sophie"; 0142156487; fonfec@yahoo.fr;
        Ouvrir(Fich) en Ajout
        Ecrire_Fichier (Fich, Truc)
        Fermer(Fich)
Fin
```

36

Exemple de lecture d'un fichier

```
Variable
        Truc : chaîne Caractère
        Fich : Fichier
Début
        Fich ← "Exemple.txt"
        Ouvrir(Fich) en Lecture
        Tantque Non EOF(Fich) Faire
                Lire_Fichier (Fich,Truc)
        FinTantQue
        Fermer(Fich)
Fin
```

37

Algorithmique – fichiers séquentiels

Exercice :

On souhaite mémoriser des noms des personnes dans un fichier nommé « **personne.txt** », créer les sous-programmes suivants :

1. Une procédure de création du fichier qui contient les noms des personnes.
2. Une procédure qui affiche les noms des personnes contenu dans le fichier.
3. Une fonction qui permet de chercher un nom passé en argument et qui renvoie vrai si ce dernier existe et faux sinon.

Ecrire un algorithme principal faisant appel aux différents sous-programmes.

38

1. Une procédure de création du fichier qui contient les noms des personnes.

```
Procedure creation(fn : Fichier)
        Variable    nom : chaine,   rep : caractère
Début
        fn ← Ouvrir ("personne.txt", ECRITURE)
        Rep← 'O'
        Tantque (rep = 'O') Faire
                Ecrire("Entrer un Nom : "),
                Lire(nom)
                Ecrire_Fichier(fn, nom)
                Ecrire("Voulez-vous ajouter un autre nom (O/N) : ")
                Lire(rep)
        FinTantque
        Fermer(fn)
FinProcedure
```

39

2. Une procédure d'affichage des noms de personnes.

```
Procedure  Affichage(fn : Fichier)
        Variable nom : chaîne
Début
        fn← Ouvrir("personne.txt", LECTURE)
        Tantque NON(EOF(fn)) Faire
                Lire_Fichier(fn, nom)
                Ecrire(nom)
        FinTantque
        Fermer(fn)
FinProcedure
```

40

3. Une fonction qui permet de chercher un nom passé en argument et qui renvoie Vrai si ce dernier existe et Faux sinon.

```
Fonction  Recherche(x : chaine ; fn : Fichier) : Booleen
        Variable nom : chaine, Trouve : Booleen
Début
        fn← Ouvrir("personne.txt", LECTURE)
        Trouve=Faux
        Tantque ((Trouve=Faux) ET (NON(EOF(fn))) Faire
                Lire_Fichier (fn, nom)
                Trouve ← nom = x
        FinTantque
        Si (EOF(fn)) Alors
                Retourner Faux
        Sinon
                Retourner Vrai
        FinSi
        Fermer(fn)
FinFonction
```

41

4. Ecrire un algorithme principal faisant appel aux différents sous-programmes.

```
Algorithme personne
        Variable F1:Fichier
Début
        Creation(F1)
        Affichage(F1)
        Si Recherche("Mohamed Ali", F1) Alors
                Ecrire("Mohamed Ali est existant dans le fichier")
        Sinon
                Ecrire("Mohamed Ali est non existant dans le fichier")
        Fin Si
Fin
```

42

Les fichiers en Python

b.aghoutane@umi.ac.ma

Généralité sur les fichiers

Remarques

- Lorsque l'on utilise un programme :
  - on saisit des informations,
  - on ≪ calcule ≫ des nouvelle informations
- À l'arrêt du programme ces informations sont perdues

- L'objectif des fichiers est de conserver ces informations d'une exécution à l'autre du programme
- Cela amène à **enregistrer** de l'information et à **relire** de l'information

Généralité sur les fichiers

Définition

- Un fichier désigne un ensemble d'informations stocké sur un support physique : disque dur, CD, DVD, clé USB.
- Un fichier se caractérise par :
  1. un nom qui permet de l'identifier de manière unique (unicité) ;
  2. un chemin d'accès qui indique son emplacement dans le disque dur ;
  3. la taille ;
  4. le type.

Types de fichiers

- Généralement, en programmation on distingue deux types de fichiers : :
  1. Fichier texte : les informations sont représentées dans le disque dur selon un code donné(UTF-8, ISO 8859, cp 1250, etc). Ces fichiers sont listables (on peut lire le contenu).
  2. Fichier binaire : les informations (n'importe quel type) sont codées telles qu'elles sont dans la mémoire (brut). Ces fichiers ne sont pas listables.

45

Ouverture d'un fichier : open(nomFichier,mode,encodage)

Principe

- La fonction **open** permet l'ouvreture d'un fichier.
- L'action d'ouverture signifie deux choses, soit on crée le fichier s'il n'existe pas ou soit on ouvre le fichier pour lire son contenu s'il existe déjà.

Syntaxe de la fonction

- La fonction **open**(NomFichier,mode,encodage) possède **trois** arguments :
  1. NomFichier : c'est la chaîne de caractères qui indique le nom de fichier à ouvrir. En général ce nom, pourra comporter une information (chemin, répertoire...) permettant de préciser l'endroit où se trouve le fichier.
  2. Le mode : c'est la chaîne de caractères qui indique le mode d'ouverture du fichier. Les différents modes les plus utilisés sont les suivants :
     * "r" : lecture seule ;
     * "w" : écriture seule (destruction de l'ancienne version si elle existe) ;
     * "a" : ouvrir le fichier en mode ajout ;
  3. Encodage :cet argument peut être ajouté pour indiquer l'encodage utilisé lors du transfert des données de la RAM vers le disque dur.

46

La fermeture d'un fichier : close

- La fonction **close**() réalise ce que l'on nomme une fermeture de fichier.
- Cette méthode force l'écriture sur disque dur du tampon associé au fichier

Création d'un fichier

- *Exemple*1
```
>>>f1=open("INFO2.txt","w")
>>> f1.close()
```
- *Exemple*2
```
>>> NomFich=input("Donnez le nom de fichier à creer :")
>>> f1=open(NomFich,"w")
>>> f1.close()
```

Ouvrir un fichier

- *Exemple*3
```
>>>f1=open("INFO2.txt","r")
>>> f1.close()
```
- *Exemple*4
```
>>> NomFich=input("Donnez le nom de fichier à ouvrir :")
>>> f1=open(NomFich,"r")
>>> f1.close()
```

Remarques

1. L'utilisation du mode "r" pour l'ouverture d'un fichier en lecture est optionnel ;
2. La précision de l'encodage lors de l'ouvreture est aussi optionnel ;

47

La fonction d'ecriture : write

Principe

Pour écrire dans un fichier texte, il faut tout d'abord ouvrir ce fichier en mode écriture ou ajout (write 'w' ou append 'a'). L'action de l'écriture consiste à transferer la valeur d'une variable (dans la RAM) de type str vers un fichier existant dans le disque dur.

Syntaxe

La fonction write permet d'écrire des données dans un fichier. Sa syntaxe est : f.write("chaîne de caractères"), f est le descripteur de fichier.

Exemple 1

- Ce code consiste à créer le fichier "test.txt"
```
>>>f=open("test.txt","w")
>>>f.write("salut tout le monde\n")
>>>f.write("Cours INFO2\n")
>>>f.close()
```

.

Résulat 1

- Sur le disque dur "test.txt"

---

salut tout le monde
Cours INFO2

.

48

La fonction d'ecriture : write

Exemple 2

- Cette fonction permet de sauvegarder la table de multiplication d'un entier n dans un fichier source

```
def TablMulti(source,n) :
    f=open(source,"w")
    for i in range(1,11) :
        f.write(str(i))
        f.write("*")
        f.write(str(n))
        f.write("=")
        f.write(str(i*n))
        f.write("\n")
    f.close()
```

.

Résulat 2

- L'appel de la fonction TablMulti("Table5.txt",5) permet de créer sur le disque dur "Table5.txt"

```
1*5=5
2*5=10
3*5=15
4*5=20
5*5=25
6*5=30
7*5=35
8*5=40
9*5=45
10*5=50
```

.

- Remarque : on peut écrire d="", d=str(i)+"*"+str(n)+"="+str(i*n)+"\n" et f.write(d)

La fonction d'ecriture : write

Ecrire à la fin de fichier

- Il peut être intéressant d'ajouter des lignes à un fichier. Il faut alors l'ouvrir en écriture en mode 'a' et non 'w'.

```
>>> f_out = open('test.txt', 'a', encoding = 'utf-8')
```

- Pour rajouter deux lignes à la fin du fichier test.txt on fait :

```
>>> f_out = open('test.txt', 'a', encoding = 'utf-8')
>>> f_out.write("2014\n")
>>> f_out.write("Pyhton est gratuit\n")
>>> f_out.close()
```

- Le contenu du fichier "test.txt" devient :

```
salut tout le monde
Cours INFO2
2014
Pyhton est gratuit
```

- Il n'est pas possible de supprimer des lignes dans un fichier. Il faut créer un nouveau fichier et détruire l'ancien !

Méthodes pour la lecture d'un fichier texte

Principe

Pour lire le contenu d'un fichier texte, il faut tout d'abord ouvrir ce fichier en mode lecture (read 'r'). L'action de lecture consiste à transferer le contenu du fichier (dans le disque dur) vers une variable (dans la RAM). Pour ce faire, on utilise la fonction read :

−− >Les variantes de la fonction read

- read() : la méthode read sans argument lit toutes les données présentes dans le fichier et les transfère dans une variable de type chaîne de caractères (string)

Exemple 1

- Ce code consiste à lire le fichier "test.txt"

```
>>>f=open("test.txt","r")
>>>ch=f.read()
>>>print(ch)
>>>f.close()
```

.

Résulat 1

- Sur l'écran on obtient :

```
salut tout le monde
Cours INFO2
2014
Pyhton est gratuit
```

.

Méthodes pour la lecture d'un fichier texte

- read(size) : la fonction read peut également être utilisée avec un argument size. Celui-ci indiquera combien de caractères doivent être lus, à partir de la position déjà atteinte dans le fichier.

Exemple 2

- Ce code consiste à lire le fichier "test.txt"

```
>>>f=open("test.txt","r")
>>>t=f.read(5)
>>>print(t)
>>>f.close()
```

.

Résulat 2

- Sur l'écran on obtient :

```
salut
```

.

52

Méthodes pour la lecture d'un fichier texte

- readline() : cette méthode ne lit qu'une seule ligne à la fois à partir du fichier (en incluant le caractère de fin de ligne). Pour lire le contenu entier de fichier, il suffit d'itérer la lecture jusqu'à ce que la ligne soit vide.

Exemple 2

```
>>>f1=open("Table5.txt","r")
>>>ch=f1.readline()
>>>print(ch)
1*5=5
>>>ch=f1.readline()
>>>print(ch)
2*5=10
>>>ch=f1.readline()
>>>print(ch)
3*5=15
>>>f1.close()
```

Parcourir un fichier avec for

```
>>>f=open("test.txt","r")
>>>f
<¡ o.TextIOWrapper name='test.txt'
encoding =' UTF − 8' >
>>> for i in f :
    print(i)
salut tout le monde
Cours INFO2
2014
Pyhton est gratuit
>>>f.close()
```

53

Méthodes pour la lecture d'un fichier texte

Exemple 4 (solution 1)

- Une fonction qui lit un fichier source caractère par caractère

```
def LireFichier(source) :
    f=open(source,"r")
    c=f.read(1)
    while c !="" :
        print(c)
        c=f.read(1)
    f.close()
```

Exemple 4 (solution 2)

- Une fonction qui lit un fichier source ligne par ligne

```
def LireFichier(source) :
    f=open(source,"r")
    while 1 :
        lg=f.readline()
        if lg !="" :
          print(lg)
        else :
          break
```

Résulat 4 : L'appel LireFichier("test.txt") affiche sur l'écran

```
salut tout le monde
Cours INFO2
2014
Pyhton est gratuit
```

54

Méthodes pour la lecture d'un fichier texte

- readlines() : cette méthode transfère toutes les lignes d'un fichier dans une liste de chaînes de caractères (chaque élément de la liste est une ligne de fichier).

Exemple 5

```
>>>f=open("test.txt","r")
>>>L=f.readlines()
>>>print(L)
['salut tout le mond\n', 'Cours INFO2\n', '2014\n', 'Python est gratuit\n']
>>>f.close()
```

Remarques

- La méthode readline() est une méthode qui renvoie une chaîne de caractères, alors que la méthode readlines() renvoie une liste ;
- À la fin du fichier, readline() renvoie une chaîne vide, tandis que readlines() renvoie une liste vide.