Consultation des données : lecture d'un fichier (Read)

Méthode traditionnelle pour le traitement de fichiers de petite taille. La consultation des données nécessite d'ouvrir une "communication" entre le programme et les fichiers. Ce canal de communication permet de recevoir un flux de données.

Pour établir la communication, il faut connaître : le "chemin d'accès" aux données (disque dur local) l'"adresse" des données (lorsqu'il s'agit de données stockées sur un serveur distant)

L'opération d'ouverture de fichier initialise un descripteur de fichier, qui sert à désigner (dans le programme) le fichier sur lequel on travaille, et d'accéder au contenu du flux.

Ouverture simple:

Python

```
f = open('monfichier.dat','r')
```

Le deuxième argument représente le mode d'ouverture, ici 'r' représente une ouverture en mode lecture.

Java

```
FileReader d = new FileReader ("monfichier.dat");
BufferedReader f = new BufferedReader(d);
```

Nous utilisons ici comme descripteur un objet de type BufferedReader signifiant explicitement que les données sont chargées en mémoire tampon (le "buffer"). Les objets FileReader et BufferedReader sont définis dans la librairie `java.io` :

```
import java.io.*;
```

Ouverture avec test :

Il est important de vérifier que cette opération d'ouverture s'effectue correctement avant de poursuivre le programme (nombreuses possibilités d'erreur : fichier effacé, erreur de nom, pas de droits de lecture,...). On utilise une instruction de test spécifique pour vérifier que l'ouverture du fichier s'est correctement effectuée, de type `try...catch...` (essaye ... sinon ...) permettant de prévoir une action de secours lorsqu'une une opération "risquée" échoue.

Python

```
try :
  f = open('monfichier.dat','r')
```

```
except IOError:
  print "Erreur d'ouverture!"
```

Java

```
try{
  FileReader d = new FileReader ("monfichier.dat");
  BufferedReader f = new BufferedReader(d);
}
catch (Exception e){
  System.out.println("Problème d'ouverture!");
}
```

Lorsque l'opération d'ouverture est réalisée avec succès, le flux de données devient accessible en lecture (les premières lignes du fichier sont chargées en mémoire et une tête de lecture se positionne sur le premier caractère de la première ligne). Il ne reste plus qu'à lire les données.

La consultation des données s'effectue séquentiellement à l'aide de l'opérateur de lecture `readline`. Chaque appel à cet opérateur charge les nouvelles données et déplace la tête de lecture vers les données suivantes. Cette opération peut être effectuée plusieurs fois jusqu'à atteindre la fin de fichier.

Si on suppose que les données sont rangées sous la forme d'une série de tuples, chaque opération de lecture consiste à consulter un tuple, et à positionner la tête de lecture sur le tuple suivant.

Exemples : lecture de données texte : chaque opération de lecture lit tous les caractères jusqu'au caractère "fin de ligne".

1. Lecture d'une ligne unique :

Python

```
s = f.readline()
```

Java

```
s = f.readLine()
```

2. Lecture de toutes les lignes (la lecture s'effectue dans une boucle) + affichage de la ligne:

Python

```
for s in f :
  print s
```

Java

```
String s;
while ((s = f.readLine()) != null){
    System.out.println(s);
    }
```

Previous : 2.1.5 Fichiers et répertoires Next : Enregistrement des données (Write)