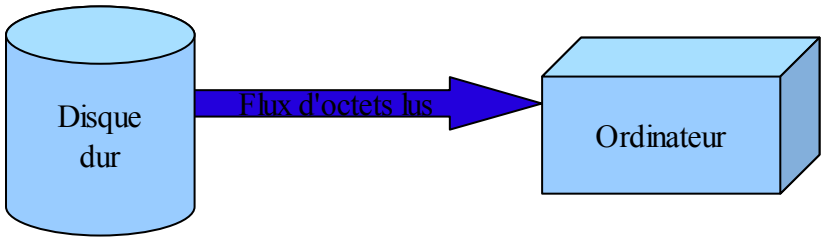
Point de vue logiciel sur les fichiers :

I) Introduction :

Les E/S fournissent ou reçoivent des données les unes à la suite des autres càd de manière séquentielle.

Le clavier envoie les numèros des touches tapées par l'utilisateur au fur et à mesure qu'il frappe sur le clavier.



II) Définition :

Un **fichier** est un **flux de données** transportant des données depuis ou vers un périphérique.

Un fichier est dit **séquentiel** lorsque les données qu'il contient, ne sont pas adressables. Pour accéder à la 100éme donnée il faut lire les 99 données la précèdant.

Un fichier est dit à accès direct ou **aléatoire** lorsque ses données sont adressables. On peut accèder directement à la centième position de donnée du fichier.

III) Système de fichier : (cf cours système)

Dans la plupart des OS, tout périphérique est vu comme un fichier et porte un nom de fichier.

IV) Utilisation d'un fichier : (rapport avec la POO)

Scénario d'utilisation	Opération sur le fichier	Information utile
• Trouver le fichier .	Créer une instance de l'objet flux d'accès fichier	Nom du fichier Eventuellement dossier contenant
• Définir la façon dont on va utiliser le fichier.	Ouvrir	Type d'utilisation : • en Lecture • en Ecriture • en Lecture/Ecriture Type d'accés : • Séquentiel. • Aléatoire.
Manipuler le fichier	Lire Ecrire	Où placer les données. Où prendre les données.
Fin d'utilisation	Fermer	

Analogie : quand on veut lire la définition d'un mot dans un dictionnaire.

- Trouver le fichier :

On commence par rechercher la position de la définition dans le livre dictionnaire. (par une méthode dichotomique car les mots sont classés suivant l'ordre alphabétique.)

- Ouvrir le fichier :
  Si on trouve le mot, on place le doigt dessus (ou la zone fovéal de notre oeil).
- Lire le fichier :
  On lit les lettres une par une de gauche à droite : il s'agit bien d'un flux de données.
- Fermeture :
  Une fois la définition lue, on referme le livre dictionnaire.

NB : Deux manières de procéder :

- Méthode synchrone : la fonction Lire bloque le programme.
- Méthode asynchrone : on demande au système de commencer la lecture du fichier et de nous informer quand la lecture sera achevée.
  Les E/S sont souvent lentes pour ne pas perdre de temps cette méthode permet à notre programme de travailler en attendant la fin de l'opération d'E/S.

V) Les fichiers en C : les concepts sont calqués sur la gestion des fichiers par le DOS.

#include<io.h>
int open(const char *path, int access [, unsigned mode]);

Créer une instance de flux d'accès à un fichier et ouvre ce fichier : l'instance est désigné par un entier unique appelé Handle. À Chaque opération effectué sur le fichier, il faudra fournir ce Handle.

int write(int handle, void *buf, unsigned len); Ecrit

int read(int handle, void *buf, unsigned len); Lit

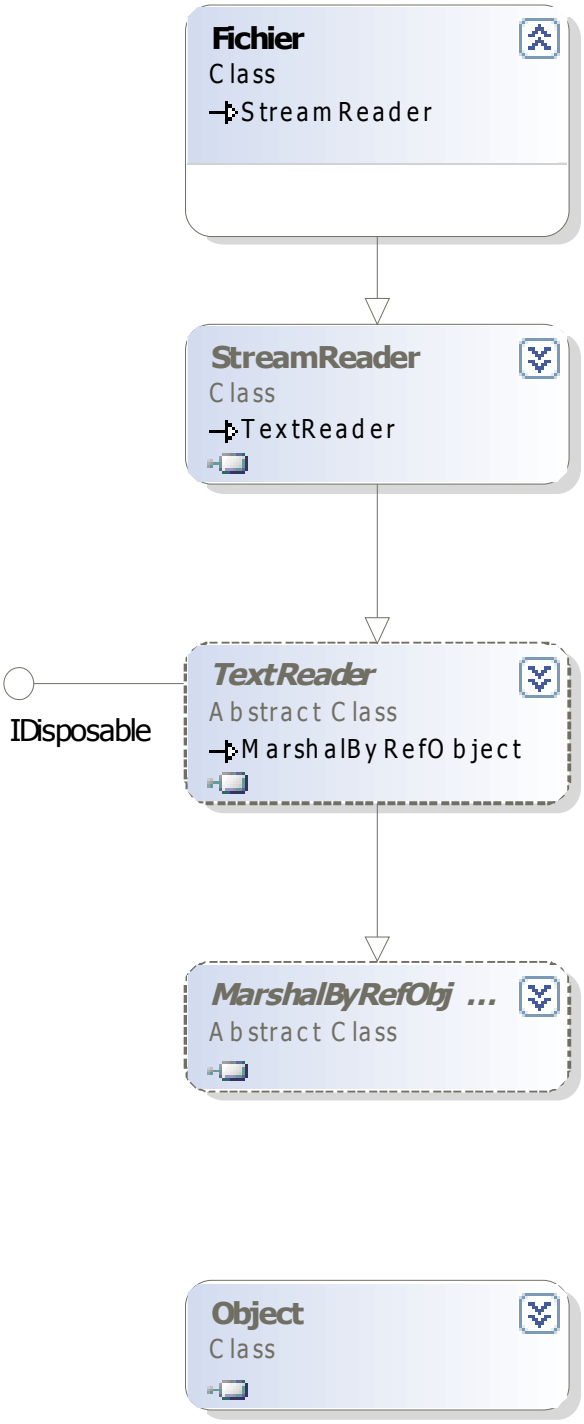
int close(int handle); Ferme le fichier correspondant au Handle

Les fichiers standards : Fichiers qui sont toujours ouverts.

Entrée standard (clavier) : cin
Sortie standard (console) : cout
Sortie d'erreur (console) : cerr

// Le 14/11/08 TSI

VI) Les fichiers avec C# :



Objet flux :

Stream, StreamReader, StreamWriter

Objet fichier :
cf Fichier et flux de données de l'aide C#
E/S de Fichier de base

Fichiers Standard pour les applications « Console » » :

Console :

- Entrée.
- Sortie.
- Erreur.

1. Utilisation de StreamReader :

- Lecture sans traitement d'exception :

```
StreamWriter sw = new StreamWriter("fichier.txt");
sw.WriteLine("Coucou");
sw.Close();
```

- Lecture avec traitement d'exception :

```
StreamReader sr;

try
{
   sr = new StreamReader("fichier1.txt");
   labelResultat.Text = sr.ReadLine();
   sr.Close();
}
catch (FileNotFoundException )
{
   MessageBox.Show(" fichier1.txt non trouvé !");
   // pbm avec le close
}
finally
{


}
```

- Utilisation de “using” :

  Utiliser de l'instruction C# using pour gérer correctement les objets systèmes tel que les flux (Fichiers). Ainsi, dans tout les cas, vous êtes sure que les ressources utilisées sont libérées même si une exception se produit. Les objets utilisés doivent implémenter l'interface IDisposable.

Exemple :

```
try {
      using (StreamReader fichier = new StreamReader("fichier.txt"))
          {
             while (!fichier.EndOfStream) // s'arrêter avant la fin du fichier
             {
               //... traitement utilisant le fichier

             }
          }
      }
   catch (FileNotFoundException e)
       {
          MessageBox.Show(" fichier.txt non trouvé !");
       }
```

ici “Using” garantie l'appel de la méthode “Dispose()” des “StreamReader” qui appelle “close” et libère les ressources.

2. Utilisation de StreamWriter : voir example

3. Utilisation des outils de sérialisation de .net :

Un certain nombre de classes de .net facilite la transformation d'objet de nos classes en fichier au format XML.

Exemple :

```
// définition de la classe à sérialiser
public class Personne
{
    public string Nom, Prenom;
    public int Age;

    public Personne()  // pour sérialiser il faut un constructeur sans paramètre
    {
    }
    public Personne(string aN, string aP, int aA)
    {
        Nom = aN;
        Prenom = aP;
        Age = aA;
    }
}

// Sérialisation
        Personne p = new Personne("Minon", "François", 46);
              // objet permettant la sérialisation xml
        XmlSerializer xmlSerializer = new XmlSerializer( typeof(Personne));

        StreamWriter sw = new StreamWriter("Personne.xml");
        xmlSerializer.Serialize(sw, p);
        sw.Close();

// Désérialisation

        // objet permettant la désérialisation xml
        XmlSerializer xmlSerializer = new XmlSerializer(typeof(Personne));
        StreamReader sr = new StreamReader("Personne.xml");
        Personne p = (Personne)xmlSerializer.Deserialize(sr);
        sr.Close();
```