Les fichiers

Une structure de données très particulière…

Les fichiers, principales caractéristiques

- Structure FIFO (first in first out)
- Extérieurs aux logiciels
  - D'où risques nombreux de difficultés : absence, modification externe, structure interne, lien éventuel avec un langage de programmation…
- Taille pratiquement illimitée (et généralement inconnue au départ)
- Accès séquentiel, direct, avec index…
- Contenu textuel ou binaire

La gestion des fichiers

- Centralisée, gérée par le système d'exploitation
  - Nom physique du fichier
  - Espace disque alloué (cylindres, pistes, secteurs, blocs…)
  - Gestion des Droits (propriétaire, utilisateur, groupes d'utilisateurs, …) lecture, écriture, exécution…

La structure interne des fichiers

- Historiquement, réalisée par le système de gestion des fichiers général, indépendamment du langage
  - + : le langage utilisé pour exploiter peut être différent de celui qui a créé
  - - : système de gestion très lourd, inadapté aux petits environnements (PC)

La structure interne des fichiers

- Aujourd'hui, le système d'exploitation fait le minimum (Unix, NT, Dos…)
- Chaque langage de programmation met en place sa propre gestion
  - + : indépendance du système
  - - : incompatibilité avec les structures internes des autres langages
  - - : obligation de refaire la gestion dans chaque langage

Les formalités

- Déclaration

```
monFichier fichier séquentiel d'Infos
```

- Ouverture

```
ouvrir monFichier en lecture
```

- Lecture ou écriture

```
lire(monFichier, uneInfo)
écrire(monFichier, uneInfo)
positionner(monFichier, unIndex)
```
(n°d'info)

Les formalités (suite)

- Fin de fichier

  ```
  si finDeFichier(monFichier) alors…
  ```

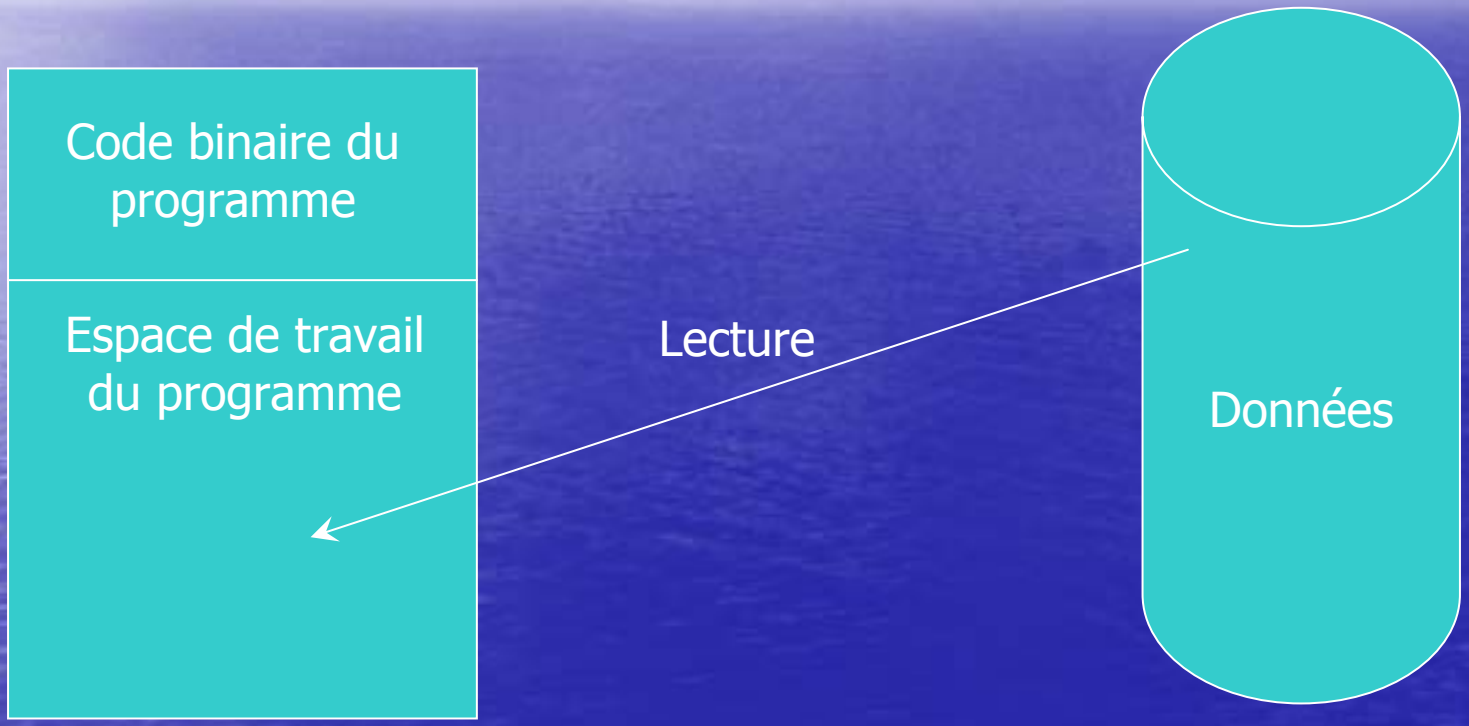
  vaut Vrai si la lecture vient d'échouer
- Fermeture

  ```
  fermer(monFichier)
  ```
- Tout dysfonctionnement entraîne une levée d'exception…

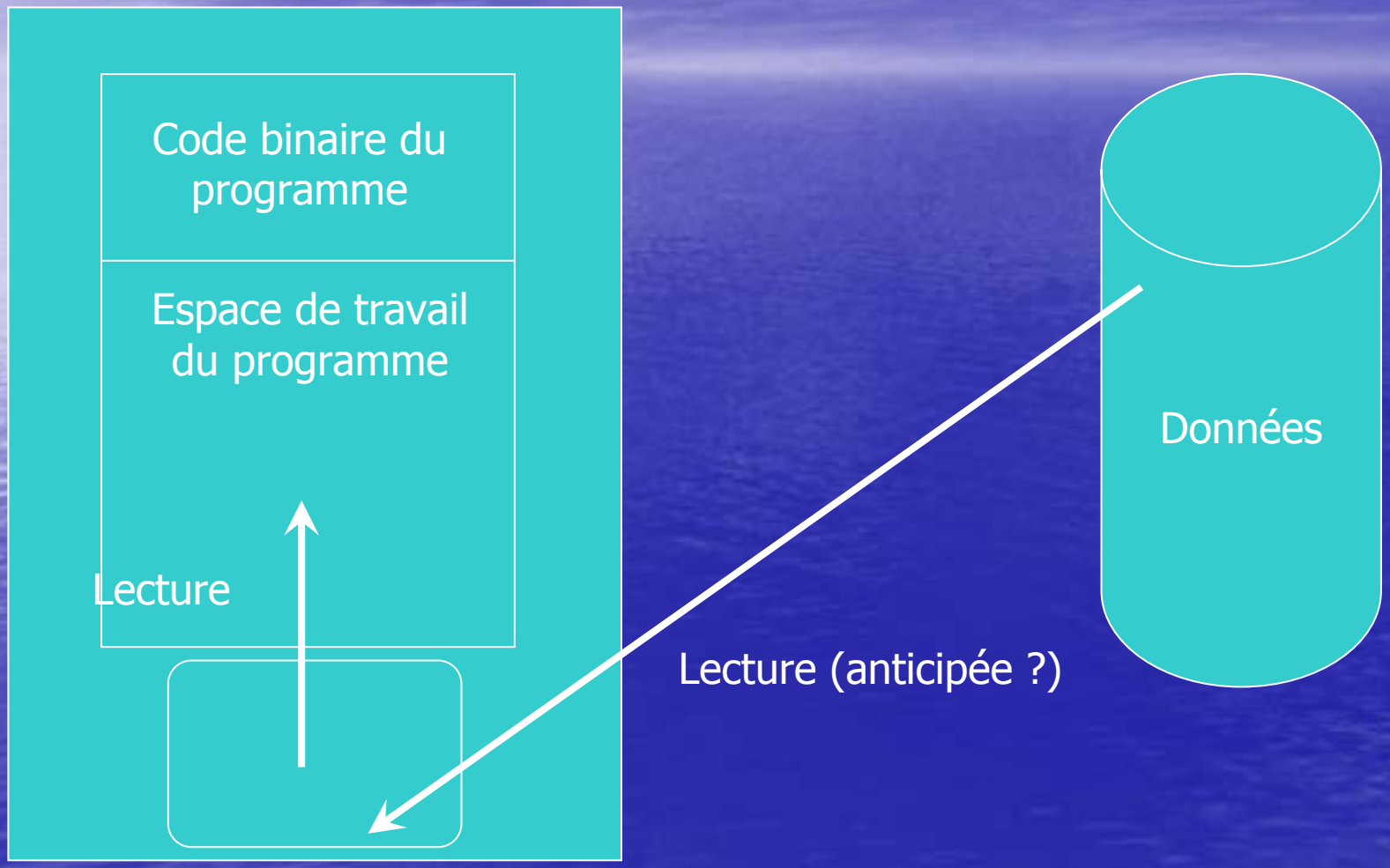
Un exemple

```
monFichier fichier direct d'entiers
(val, cpt, max, position) entiers
…
ouvrir monFichier en lecture
cpt:=0; max:=0; position:=-1
lire(monFichier, val)
tantque non finDeFichier(monFichier) faire
    cpt := cpt+1
    si val > max alors
        max := val
        position:=cpt
    lire(monFichier, val)
si position ≠ -1 alors
    positionner(monFichier, position)
    lire(monFichier, val)
    si val ≠ max alors leverException
fermer(monFichier)
```

Les accélérateurs : sans tampon



Les accélérateurs : avec tampon



Les accélérateurs

- Notion de tampon (buffer)
  - Dans le langage (exemple Java)

```
BufferedReader unFichier = null;
String sonContenu=new String("");
try          {
             unFichier = new BufferedReader(new FileReader(…));
             do{sonContenu=unFichier.readLine();} while (…);
             unFichier.close();
             }
catch (FileNotFoundException ex)
             {System.err.println("Pas de fichier de ce nom");}
catch (IOException ex) {System.err.println("Erreur en lecture");}
```

  - Par le système de gestion des fichiers (blocs et inodes Unix…)
- Problèmes éventuels liés au vidage ou à la taille du tampon

Les structures internes

- format des enregistrements internes…
  - format indéfini (U)
  - format fixe (F) ou fixe avec blocs (FB)
  - format variable (V) ou variable avec blocs (VB)

Les modes d'accès

- séquentiel
  - impossible de positionner, sauf en accumulant (et éventuellement en comptant) les lectures…
- direct
  - très pratique (accès aléatoire comme dans un tableau)
  - perd beaucoup d'avantages de la bufferisation
- séquentiel indexé
  - direct mais par l'intermédiaire d'une clef liée au numéro d'enregistrement
  - problème de la gestion de ce lien (tableau, hachage, …)

Efficacités comparées (structure, mode d'accès)

	Effort	Séquentiel	Direct	Séq. indexé
U	Aucun	Bon	Mauvais	Mauvais
F	Gros	Très bon	Très bon	Très bon
FB	Gros	Excellent	Excellent	Excellent
V	Petit	Bon	Mauvais	Mauvais
VB	Gros	Très bon	Mauvais	Mauvais

Choix du type de contenu (binaire ou textuel)

- Binaire
  - inutile de transcoder
  - généralement non portable
  - rapide
  - résultat généralement de taille fixe
  - relative confidentialité
- Textuel
  - obligation de transcoder les nombres
  - portable
  - lent
  - résultat généralement de taille variable
  - fichier modifiable de l'extérieur

XML (eXtended Markup Language)

- Un format **textuel** préparé pour le Web et contrôlable

```
<?xml version="1.0"?>
<exemple>
    <cours>
        <nom>algorithmique</nom>
        <responsable>
            <nom>Fournier</nom>
            <prenom>Jean-Pierre</prenom>
        </responsable>
        <volume>1</volume>
    </cours>
</exemple>
```

DTD(Document Type Definition)

- Pour vérifier qu'un fichier xml ne contient pas n'importe quoi…
  - après <?xml…?> :

    <!DOCTYPE exemple SYSTEM "exemples.dtd">
  - dans le fichier exemples.dtd :
    - <!ELEMENT exemple (cours+)>
    - <!ELEMENT cours (nom, responsable+, volume*)>
    - <!ELEMENT nom (#PCDATA)>
    - <!ELEMENT responsable (nom, prenom)>
    - <!ELEMENT volume (#PCDATA)>