IUT de Lens – DUT Info S3 – SE3 : Principes des systèmes d'exploitation

TD n^{o}2 : Fichiers binaires

1 Exemple introductif

Dans cet exemple (`FichierBinaire.java` sur Moodle), on va enregistrer dans un fichier binaire une liste de produits avec leur référence (un entier) et leur prix (un nombre flottant). La procédure `ecrire()` va générer un fichier, la procédure `lire()` va afficher le contenu de ce fichier et la procédure `lireALEnvers()` illustre le déplacement de la tête de lecture/écriture.

Utilisez la commande `hexdump -C /tmp/catalogue.bin` pour comprendre la représentation des données.

```
import java.io.*;
import java.util.*;
import java.nio.*;
import java.nio.channels.*;
import java.nio.file.*;

class FichierBinaire {
  class Produit {
    int ref; // une référence
    float prix; // un prix

    // nombre d'octets pour stocker un produit
    static final int BYTES=Integer.BYTES+Float.BYTES;
  };

  FileChannel f; // le fichier binaire
  ByteBuffer buf; // le tampon pour écrire dans le fichier

  /**
   * écrire un produit à la position courante du fichier
   */
  void ecrireProduit(Produit prod) throws IOException {
    // copier le produit dans le tampon
    buf.clear(); // avant d'écrire, on vide le tampon
    buf.putInt(prod.ref);
    buf.putFloat(prod.prix);
    // copier le tampon dans le fichier
    buf.flip(); // passage à une lecture du tampon
    while(buf.hasRemaining()) // tant qu'on n'a pas écrit tout le buffer
      f.write(buf);
  }

  /**
   * lire un produit à la position courante du fichier
   */
  Produit lireProduit() throws IOException {
    // copie du fichier vers le tampon
    buf.clear(); // avant d'écrire, on vide le tampon
    while(buf.hasRemaining()) // tant qu'on n'a pas rempli le buffer
      if(f.read(buf)==-1)
        return null;
    // copie du tampon vers le produit
    buf.flip(); // passage à une lecture du tampon
    Produit prod=new Produit();
    // il faut relire les données dans le même ordre que lors de l'écriture
    prod.ref=buf.getInt();
    prod.prix=buf.getFloat();
    return prod;
  }

  FichierBinaire(String filename) throws IOException {
    //ouverture en lecture/écriture, avec création du fichier
    f=FileChannel.open(
                       FileSystems.getDefault().getPath(filename),
                       StandardOpenOption.READ,
                       StandardOpenOption.WRITE,
                        StandardOpenOption.CREATE);
    // création d'un buffer juste assez grand pour contenir un produit
    buf=ByteBuffer.allocate(Produit.BYTES);
  }

  /**
   * création du fichier
   */
```

```
  void ecrire() throws IOException {
    Produit prod=new Produit();
    for(int id=1;id<=5;id++) {
      prod.ref=id;
      prod.prix=id*10;
      ecrireProduit(prod);
    }
  }

  /**
   * relecture du fichier
   */
  void lire() throws IOException {
    Produit prod;
    f.position(0); // revenir au début du fichier

    while((prod=lireProduit())!=null)
      System.out.println(prod.ref+"\t"+prod.prix);
  }

  /**
   * relecture du fichier à l'envers
   */
  void lireALEnvers() throws IOException {
    Produit prod;
    long pos=f.size()-Produit.BYTES; // position du dernier produit

    while(pos>=0) {
      f.position(pos);
      prod=lireProduit();
      System.out.println(prod.ref+"\t"+prod.prix);
      pos-=Produit.BYTES;
    }
  }


  void run() throws IOException {
    ecrire();
    lire();
    lireALEnvers();
    f.close();
  }

  public static void main(String[] args) {
    try {
      FichierBinaire bin=new FichierBinaire("/tmp/catalogue.bin");
      bin.run();
    }
    catch(Exception e) {
      e.printStackTrace();
      System.exit(1);
    }
  }
}
```

2 Exercices

Exercice 1 : Pour chaque produit d'un site de vente, on doit enregistrer son identifiant (int), son prix (float) et la quantité en stock (int). Ces informations seront stockées dans un fichier binaire. Dans cet exercice, on ne cherchera pas à trier les produits dans le fichier. Toutes les modifications devront se faire directement sur le fichier (autrement dit, on suppose qu'on ne peut pas charger tout le fichier en mémoire).

Écrivez un programme pour permettre à un utilisateur

- d'afficher la liste des produits,
- d'afficher les informations sur un produit désigné par son identifiant,
- d'ajouter un produit (si le produit existe déjà, on met à jour le prix et la quantité),
- d'enregistrer une variation de quantité pour un produit,
- de supprimer un produit désigné par son identifiant.
- pour les besoin des tests, ajouter n produits créés aléatoirement (on ne cherchera pas à garantir l'unicité de l'identifiant).

Expliquez pourquoi il faut utiliser un fichier binaire dans cet exercice.

Exercice 2 : Reprenez l'exercice précédent en garantissant qu'à tout moment le fichier est trié par ordre croissant des identifiants de produit.

Exercice 3 : Programmez la commande `cat`.

Exercice 4 : Programmez la commande `cmp`.