Université Bordeaux I
U.F.R. Mathématiques et Informatique

IUP MIAGe 3^{eme} année
Année 1998-1999

Jeudi 21 janvier, 14 heures - 16 heures
Tous documents autorisés.

Programmation Parallèle
Alain Griffault

SUJET + CORRIGE

Exercice 1 (1 points) : Une recette de cuisine

Afin de préparer deux omelettes, une première aux lardons et aux champignons et une seconde aux lardons et aux oignons, le chef cuisinier défini huit tâches numérotées ainsi :

1. Faire revenir les lardons.
2. Couper les oignons.
3. Faire revenir les oignons.
4. Couper les champignons en lamelles.
5. Faire revenir les champignons.
6. Casser et battre les œufs.
7. Faire cuire ensemble : la moitié des lardons, la moitié des œufs et les champignons.
8. Faire cuire ensemble : la moitié des lardons, la moitié des œufs et les oignons.

Question 1 *Dessiner l'unique graphe d'ordonnancement des huit tâches, qui fait que le résultat final est bien celui attendu : les deux omelettes..*

Réponse à la question 1(1 points)

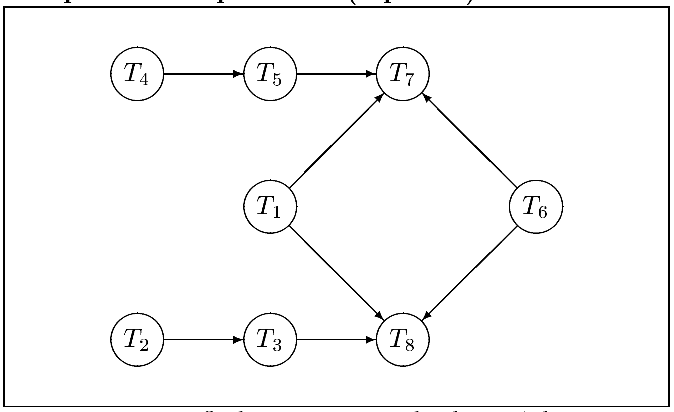

Figure 1: Ordonnancement des huit tâches.

Exercice 2 (4 points) : Ordonnancement de tâches avec des sémaphores

Question 2 *Donner une liste de sémaphores (avec les valeurs initiales pour chacun) afin de pouvoir ordonnancer la préparation des omelettes.*

Réponse à la question 2(1 points) Une façon simple de résoudre l'ordonnancement est de prendre un sémaphore par tâche dépendante, et d'initialiser ces sémaphores à zéro.

Question 3 *Redessiner le graphe précédent, ornementé de vos sémaphores afin d'indiquer quelles tâches les utilisent.*

Réponse à la question 3(1 points)

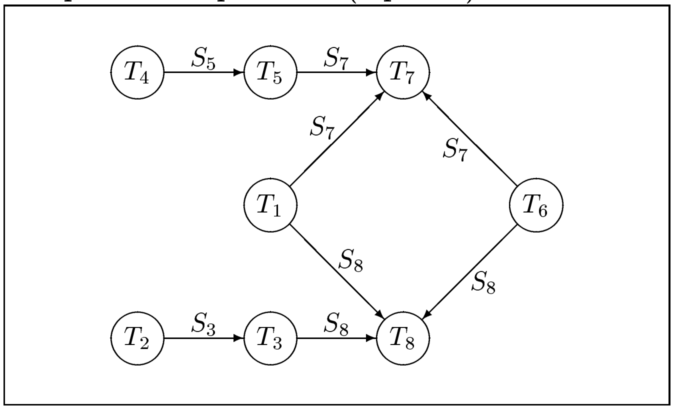

Figure 2: Les sémaphores pour les huit tâches.

Question 4 *Placer aux bons endroits dans les codes des huit tâches, les instructions P et V qui manipulent les sémaphores.*

Réponse à la question 4(2 points)

T_1

Faire revenir les lardons
$V(S_7)$
$V(S_8)$

T_2

Couper les oignons
$V(S_3)$

T_3

$P(S_3)$
Faire revenir les oignons
$V(S_8)$

T_4

Couper les champignons en lamelles
$V(S_5)$

T_5

$P(S_5)$
Faire revenir les champignons
$V(S_7)$

T_6

Casser et battre les œufs
$V(S_7)$
$V(S_8)$

T_7

$P(S_7)$
$P(S_7)$
$P(S_7)$
Omelette : lardons et champignons

T_8

$P(S_8)$
$P(S_8)$
$P(S_8)$
Omelette : lardons et oignons

Exercice 3 (4 points) : Ordonnancement de tâches avec des commandes gardées non-déterministes

Question 5 *Compléter le source suivant afin d'obtenir les deux omelettes.*

Réponse à la question 5(4 points)

```
Omelettes ::

// déclaration et initialisation des variables
  var
    booleen lardonsRevenus = false;
    booleen oignonsCoupes = false;
    booleen oignonsRevenus = false;
    booleen champignonsCoupes = false;
    booleen champignonsRevenus = false;
    booleen oeufsBattus = false;
    booleen omeletteLardonsChampignons = false;
    booleen omeletteLardonsOignons = false;

// une structure répétitive qui contient les commandes gardées.
  *[
     (lardonsRevenus == false)
       --> T1; lardonsRevenus = true;

        []
     (oignonsCoupes == false)
       --> T2; oignonsCoupes = true;

        []
     (oignonsCoupes == true) et (oignonsRevenus == false)
       --> T3; oignonsRevenus = true;

        []
     (champignonsCoupes == false)
       --> T4; champignonsCoupes = true;

        []
     (champignonsCoupes == true) et (champignonsRevenus == false)
       --> T5; champignonsRevenus = true;

        []
     (oeufsBattus == false)
       --> T6; oeufsBattus = true;

        []
     (champignonsRevenus == true) et (lardonsRevenus == true) et
     (oeufsBattus == true) et (omeletteLardonsChampignons == false)
       --> T7; omeletteLardonsChampignons = true;

        []
     (champignonsRevenus == true) et (oignonsRevenus == true) et
     (oeufsBattus == true) et (omeletteLardonsOignons == false)
       --> T8; omeletteLardonsOignons = true;
  ]
```

Exercice 4 (6 points) : Ordonnancement de tâches en Java

Question 6 *Modifier les classes* `Oeuf`, `Oignon`, `Lardon` et `Champignon` *afin d'obtenir les deux omelettes, et non la sortie suivante sur écran.*

Une sortie écran.

```
Faire revenir les lardons : début
Couper les champignons : début
Faire revenir les champignons : début
Couper les oignons : début
Faire revenir les oignons : début
Casser et battre les oeufs : début
Omelette lardons et champignons : début
Omelette lardons et oignons : début
Couper les oignons : fin
Casser et battre les oeufs : fin
Couper les champignons : fin
Omelette lardons et champignons : fin
Omelette lardons et oignons : fin
Faire revenir les lardons : fin
Faire revenir les oignons : fin
Faire revenir les champignons : fin
```

Le source Java.

```
import java.util.*;

class T1 extends Thread {
  Lardon lardons;
  int duree;

  T1(Lardon lardons, int duree) {
    this.lardons = lardons;
    this.duree = duree;
  }

  public void run() {lardons.faireRevenir(this,duree);}
}

class T2 extends Thread {

  Champignon champignons;
  int duree;

  T2(Champignon champignons, int duree) {
    this.champignons = champignons;
    this.duree = duree;
  }

  public void run() {champignons.couper(this,duree);}
}

class T3 extends Thread {

  Champignon champignons;
  int duree;

  T3(Champignon champignons, int duree) {
```

```
    this.champignons = champignons;
    this.duree = duree;
  }

  public void run() {champignons.faireRevenir(this,duree);}
}

class T4 extends Thread {

  Oignon oignons;
  int duree;

  T4(Oignon oignons, int duree) {
    this.oignons = oignons;
    this.duree = duree;
  }

  public void run() {oignons.couper(this,duree);}
}


class T5 extends Thread {

  Oignon oignons;
  int duree;

  T5(Oignon oignons, int duree) {
    this.oignons = oignons;
    this.duree = duree;
  }

  public void run() {oignons.faireRevenir(this,duree);}
}

class T6 extends Thread {

  Oeuf oeufs;
  int duree;

  T6(Oeuf oeufs, int duree) {
    this.oeufs = oeufs;
    this.duree = duree;
  }

  public void run() {oeufs.casserBattre(this,duree);}
}

class T7 extends Thread {

  Oeuf oeufs;
  Lardon lardons;
  Champignon champignons;
  int duree;

  T7(Lardon lardons, Champignon champignons, Oeuf oeufs, int duree) {
    this.lardons = lardons;
    this.champignons = champignons;
```

```java
    this.oeufs = oeufs;
    this.duree = duree;
  }

  public void run() {
    lardons.prendreMoitie();
    champignons.prendre();
    oeufs.prendreMoitie();
    System.out.println("Omelette lardons et champignons : début");
    try {sleep(duree);}catch(InterruptedException e){}
    System.out.println("Omelette lardons et champignons : fin");
  }
}

class T8 extends Thread {

  Oeuf oeufs;
  Lardon lardons;
  Oignon oignons;
  int duree;

  T8(Lardon lardons,Oignon oignons , Oeuf oeufs, int duree) {
    this.lardons = lardons;
    this.oignons = oignons;
    this.oeufs = oeufs;
    this.duree = duree;
  }

  public void run() {
    lardons.prendreMoitie();
    oignons.prendre();
    oeufs.prendreMoitie();
    System.out.println("Omelette lardons et oignons : début");
    try {sleep(duree);}catch(InterruptedException e){}
    System.out.println("Omelette lardons et oignons : fin");
  }
}

public class ChefCuisinier {

  public static Oeuf oeufs = new Oeuf();
  public static Lardon lardons = new Lardon();
  public static Champignon champignons = new Champignon();
  public static Oignon oignons = new Oignon();

  public static void main(String []args) {
    T1 t1 = new T1(lardons,100); t1.start();
    T2 t2 = new T2(champignons,50); t2.start();
    T3 t3 = new T3(champignons,200); t3.start();
    T4 t4 = new T4(oignons,10); t4.start();
    T5 t5 = new T5(oignons,150); t5.start();
    T6 t6 = new T6(oeufs,20); t6.start();
    T7 t7 = new T7(lardons,champignons,oeufs,50); t7.start();
    T8 t8 = new T8(lardons,oignons,oeufs,50); t8.start();
  }
}
```

Réponse à la question 6(6 points)

```
class Lardon {
  static boolean revenu = false;
  static Boolean verrou = new Boolean(true);

  synchronized public void faireRevenir(Thread thread, int duree) {
    System.out.println("Faire revenir les lardons : début");
    try {thread.sleep(duree);}catch(InterruptedException e){}
    System.out.println("Faire revenir les lardons : fin");

    revenu = true;
    synchronized(verrou){verrou.notifyAll();}
  }

  synchronized public void prendreMoitie() {
    while (revenu == false) {
      try {synchronized(verrou){verrou.wait();}}
      catch(InterruptedException e){}
    }
  }
}

class Oignon {
  static boolean coupe = false;
  static boolean revenu = false;
  static Boolean verrou = new Boolean(true);

  synchronized public void couper(Thread thread, int duree) {
    System.out.println("Couper les oignons : début");
    try {thread.sleep(duree);}catch(InterruptedException e){}
    System.out.println("Couper les oignons : fin");

    coupe = true;
    synchronized(verrou){verrou.notifyAll();}
  }

  synchronized public void faireRevenir(Thread thread, int duree) {
    while (coupe == false) {
      try {synchronized(verrou){verrou.wait();}}
      catch(InterruptedException e){}
    }

    System.out.println("Faire revenir les oignons : début");
    try {thread.sleep(duree);}catch(InterruptedException e){}
    System.out.println("Faire revenir les oignons : fin");

    revenu = true;
    synchronized(verrou){verrou.notifyAll();}
  }

  synchronized public void prendre() {
    while (revenu == false) {
      try {synchronized(verrou){verrou.wait();}}
      catch(InterruptedException e){}
    }
  }
}
```

```
class Champignon {
  static boolean coupe = false;
  static boolean revenu = false;
  static Boolean verrou = new Boolean(true);

  public void couper(Thread thread, int duree) {
    System.out.println("Couper les champignons : début");
    try {thread.sleep(duree);}catch(InterruptedException e){}
    System.out.println("Couper les champignons : fin");

    coupe = true;
    synchronized(verrou){verrou.notifyAll();}
  }

  public void faireRevenir(Thread thread, int duree) {
    while (coupe == false) {
      try {synchronized(verrou){verrou.wait();}}
      catch(InterruptedException e){}
    }

    System.out.println("Faire revenir les champignons : début");
    try {thread.sleep(duree);}catch(InterruptedException e){}
    System.out.println("Faire revenir les champignons : fin");

    revenu = true;
    synchronized(verrou){verrou.notifyAll();}
  }

  public void prendre() {
    while (revenu == false) {
      try {synchronized(verrou){verrou.wait();}}
      catch(InterruptedException e){}
    }
  }
}

class Oeuf {
  static boolean battu = false;
  static Boolean verrou = new Boolean(true);

  public void casserBattre(Thread thread, int duree) {
    System.out.println("Casser et battre les oeufs : début");
    try {thread.sleep(duree);}catch(InterruptedException e){}
    System.out.println("Casser et battre les oeufs : fin");

    battu = true;
    synchronized(verrou){verrou.notifyAll();}
  }

  synchronized public void prendreMoitie() {
    while (battu == false) {
      try {synchronized(verrou){verrou.wait();}}
      catch(InterruptedException e){}
    }
  }
}
```

Exercice 4 (6 points) : Un serveur à jeton en Java

Question 7 *Compléter les sources suivant afin qu'il n'y ai jamais plus de vingt clients simultanés dans une télé-discussion.*
Le serveur avertira "gentiment" le client comme quoi la limite maximale de connexions simultanées est atteinte, qu'il pourra réessayer plus tard, puis fermera la connexion.

Réponse à la question 7(6 points) C'est dans la classe Connexion que le nombre de connexion est le plus facile à determiner. En effet le serveur n'a pas connaissance des fins de connexions, et ce n'est pas le rôle du nettoyeur.

Un autre argument, est le besoin d'avertir le client. Tous les dialogues étant gérés par la classe connexion, il semble normal que cet avertissement se fasse dans cette classe.

```
import java.net.*;
import java.util.*;

public class Serveur extends Thread {
  protected ServerSocket ecoute;
  protected Vector connexions;
  protected Nettoyeur nettoyeur;

  public Serveur(int port) {
    try {
      ecoute = new ServerSocket(port);
    }
    catch(IOException e) {
      System.err.println(e.getMessage());
      System.exit(1);
    }
    System.out.println("Serveur en écoute sur le port : "+port);
    connexions = new Vector();
    nettoyeur = new Nettoyeur(this);
    this.start();
  }

  public void run() {
    try {
      while (true) {
        Socket client = ecoute.accept();
        System.out.println("Demande de connexion... ");
        Connexion c = new Connexion(client, nettoyeur, this);
        synchronized (connexions) {connexions.addElement(c);}
      }
    }
    catch(IOException e) {
      System.err.println(e.getMessage());
      System.exit(1);
    }
  }

  public static void main(String[] argv) {
    if (argv.length != 1) {
      System.err.println("Usage : java Serveur <port>");
    }
    else {
      new Serveur(Integer.parseInt(argv[0]));
    }
  }
}
```

```
import java.io.*;
import java.net.*;
import java.util.*;

class Connexion extends Thread {
  protected Socket client;
  protected Serveur serveur;
  protected Nettoyeur nettoyeur;
  protected DataInputStream in;
  protected PrintStream out;

  protected static int nombreConnexion = 0;
  protected static int nombreConnexionMax = 2;

  public Connexion(Socket clientSocket, Nettoyeur n, Serveur s) {
    client = clientSocket;
    nettoyeur = n;
    serveur = s;
    try {
      in = new DataInputStream(client.getInputStream());
      out = new PrintStream(client.getOutputStream());
    }
    catch(IOException e) {
      try {
        client.close();
      }
      catch(IOException el) {}
      System.err.println(e.getMessage());
      return;
    }
    if (nombreConnexion < nombreConnexionMax) {
      this.start();
    }
    else {
      this.refus();
    }
  }

  public void refus() {
    this.out.println("Desole, mais ce serveur est limite a " +
                     nombreConnexionMax +
                     " connexions simultanees.\n");
    try {
      client.close();
    }
    catch(IOException e) {}
    System.out.println("Fin de connexion...");
    synchronized(serveur.connexions) {serveur.connexions.notify();}
  }

  public void run() {
    String ligne;
    Connexion c;
    nombreConnexion++;
    try {
      while (true) {
        ligne = in.readLine();
```

```
        synchronized(serveur.connexions) {
          for (int i = 0; i<serveur.connexions.size();i++) {
            c = (Connexion) serveur.connexions.elementAt(i);
            c.out.println(ligne);
          }
        }
        if (ligne.endsWith("FIN"))
          break;
      }
    }
    catch(IOException e) {
    }
    finally {
      try {
        client.close();
      }
      catch(IOException e) {}
      System.out.println("Fin de connexion...");
      synchronized(serveur.connexions) {serveur.connexions.notify();}
      nombreConnexion--;
    }
  }
}

import java.io.*;
import java.net.*;
import java.util.*;

class Nettoyeur extends Thread {
  protected Serveur serveur;

  protected Nettoyeur(Serveur s) {
    serveur = s;
    this.start();
  }

  public void run() {
    Connexion c;
    while (true) {
      try {
        synchronized(serveur.connexions) {serveur.connexions.wait();}
      }
      catch(InterruptedException e) {
      }
      synchronized(serveur.connexions) {
        for (int i = serveur.connexions.size()-1;i>=0;i--) {
          c = (Connexion) serveur.connexions.elementAt(i);
          if (!c.isAlive()) {
            serveur.connexions.removeElementAt(i);
            System.out.println("Fin de connexion : OK");
          }
        }
      }
    }
  }
}
```