Université de Chlef
Département Informatique
Filière : 1 ère Année Master - Ingénierie des Logiciels

Janvier 2016

Examen semestriel

Module de Systèmes d'exploitation

Corrigé

Exercice 1 : (7 points)

On considère le problème classique de synchronisation du "Producteur/Consommateur" mais avec la variante suivante :

- On dispose de deux producteurs différents P1 et P2 qui peuvent produire des messages et les déposer dans le buffer : P1 dépose un message, mais P2 dépose deux à la fois.
- Il n'ya qu'un seul consommateur C.
- Le buffer est limité (de taille N) .

Proposez une synchronisation des processus P1, P2 et C en utilisant les sémaphores.

Réponse :

Empty : semaphore (init à N) / pour représenter le nombre de case vides*
Full : semaphore (init à 0) / pour représenter le nombre de cases pleines*
In : entier (init à 0) / représente l'indice de l'élément qui vient d'être déposé*
Out : entier (init à 0) / représente l'indice de l'élément qui est prêt à être prélevé*
N : Constante / représente la taille du buffer*
Mutex : sémaphore (init à 1) / est utilisé pour protéger en exclusion mutuelle la variable partagée In.*
/ En effet, comme , il y'a deux producteurs , la variable in devient une variable partagée entre eux.*

```
Processus Producteur P1
Début
    Cycle
        Produire un message dans ZoneP1;
        Wait(Mutex);
        Wait(Empty);
        Buffer[In] :=ZoneP1 ;
        In :=In+1 mod N;
        Signal(Mutex);
        Signal(Full);
    Fin Cycle
Fin.
```

```
Processus Producteur P2
Début
    Cycle
        Produire un message dans ZoneP2;
        Wait(Mutex);
        Wait(Empty);
        Wait(Empty);
        For I:=1 jusqu'à 2

        Buffer[In] :=ZoneP2-I;
            In :=In+1 mod N;
        Fin_Pour ;
        Signal(Mutex);
        Signal(Full);
        Signal(Full);
    Fin Cycle
Fin.
```

```
Processus Consommateur
Début
    Cycle
        Wait(Full)
        ZoneC :=Buffer[Out] ;
        Out :=Out+1 mod N ;
        Signal(Empty) ;
        Consommer le message de ZoneC
    Fin Cycle
Fin.
```

(7 points)

Exercice 2 : (5 points)

On considère trois processus P1, P2 et P3 qui demandent pour leur exécution des ressources critiques R1, R2 et R3. L'exécution des processus se fait selon le schéma suivant :

Processus P1	Processus P2	Processus P3
Debut Demander (R1) Demander (R2) Exécution Libérer(R2) Libérer(R1) Fin.	Debut Demander (R2) Demander (R3) Exécution Libérer(R3) Libérer(R2) Fin.	Debut Demander (R3) Demander (R1) Exécution Libérer(R1) Libérer(R3) Fin.

Question 1 : Montrer qu'on peut avoir un interblocage avec ces trois processus.

Réponse :

Supposons que l'ordre d'exécution soit le suivant :

(1) Le processus P1 demande la ressource R1 et l'obtient
(2) Le processus P2 demande la ressource R2 et l'obtient
(3) Le processus P3 demande la ressource R3 et l'obtient
(4) Le processus P1 demande la ressource R2, mais il ne peut pas l'avoir car elle est occupée par P2
(5) Le processus P2 demande la ressource R3, mais il ne peut pas l'avoir car elle est occupée par P3
(6) Le processus P3 demande la ressource R1, mais il ne peut pas l'avoir car elle est occupée par P1

et l'interblocage s'installe.Les processus sont dans une interminable liste circulaire d'attente.

(2.5 points)

Question 2 : Comment peut-on éviter cet interblocage ?

Réponse :

Pour éviter l'interblocage, on fait en sorte que l'une au moins des 4 conditions nécessaires et suffisantes d'apparition de l'interblocage ne soit pas vérifiée. Par exemple, pour éviter d'avoir une liste circulaire d'attente, on peut adopter le protocole suivant (présenté en cours) :
On oblige tous les processus à demander les ressources dans l'ordre croissant des ressources :
Le processus P1 doit demander la ressource R1 et ensuite la ressource R2 (c'est ce qu'il fait).
Le processus P2 doit demander la ressource R2 et ensuite la ressource R3 (c'est ce qu'il fait).
Le processus P3 doit demander la ressource ***R1 et ensuite la ressource R3*** *.*

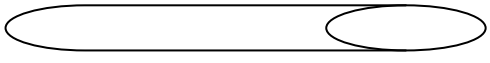
(2.5 points)

<u>Exercice 3 :</u> (4 points)

Question 1 : Expliquez le principe des communications par "tubes" entre processus.

Réponse :

Un tube est un dispositif de communication qui permet une communication à sens unique entre deux processus Père et Fils. Après avoir créé le tube (grâce à une instruction spéciale), le processus émetteur écrit les données sur son côté du tube. Le processus récepteur lit les donnés envoyées au fur et à mesure de leur arrivée à partir de son extrémité du tube . Les tubes sont des dispositifs séquentiels; les données sont toujours lues dans l'ordre où elles ont été écrites.

Processus émetteur →  → processus récepteur

(2 points)

Question 2 : Donner les avantages et les inconvénients de la méthode de communication par "tubes".

Réponse :

Avantage : C'est le système de l'IPC lui-même qui s'occupe de la synchronisation des processus émetteur et récepteur lors du transfert.
Inconvénient : Ne fonctionne que pour des processus liés (de type Père-Fils) , et le transfert est à sens unique.

(2 points)

Exercice 4 **: (4 points)**
Donner le code java permettant à un thread client d'envoyer 10 lignes de textes (correspondant aux chiffres 1, 2, ... 10) à un thread serveur, en utilisant les sockets. On crée le socket au début et on le ferme à la fin.

Réponse :

```
import java.io.*;
import java.net.*;
/** Le processus client se connecte au serveur "Machine_Serveur" sur le port 8080
 */
public class Client {
  static final int port = 8080;

  public static void main(String[] args) throws Exception {
      Socket socket = new Socket("Machine_Serveur", port);   // Ouverture du socket
      System.out.println("SOCKET = " + socket);

      PrintWriter pred = new PrintWriter(       //Buffer qui servira à l'envoi
                    new BufferedWriter(
                      new OutputStreamWriter(socket.getOutputStream())),
                    true);

      System.out.println("Début des transferts");
      /*Boucle d'envoi des lignes de textes */
      for (int i = 0; i < 10; i++) {
        pred.println(i);        // envoi d'un message
      }
      System.out.println("Fin des Transferts");     // message de terminaison
      pred.println("END") ;
      pred.close();
      socket.close();                       //Fermeture du socket
  }
}
```

(4 points)