Université de Chlef
Département Informatique
Filière : 3ème LMD (S5)

Février 2010

Examen semestriel

Module de Systèmes d'exploitation II

durée 1H30

Corrigé

Exercice 1 : La solution au problème du coiffeur endormi suivante présente un problème. Lequel ? Expliquez clairement.

Processus Coiffeur	*Processus Client*	Déclarations :
Début	Début	
Cycle	Wait(mutex)	SClient, SCoiffeur :
Wait(SClient)	Si (attente<Nbre_Chaise)	sémaphores initialisés à 0.
Wait(mutex)	Alors	Mutex : sémaphore, initialisé
Attente :=attente-1	Attente :=attente+1 ;	à 1.
Signal(SCoiffeur)	Signal(mutex)	Attente : entier initialisé à 0.
Signal(mutex)	Signal(SClient)	
Couper_cheveux	Wait(SCoiffeur)	
FinCycle	Obtenir_coupe	
Fin	Sinon Signal(mutex)	
	Finsi	
	Fin	

Réponse :

Le problème de cette solution réside dans le fait que dans le processus Client , l'instruction Signal(mutex) doit suivre Signal(Sclient) et non pas la précéder. La conséquence de cette erreur, est qu'on peut avoir une situation où plusieurs processus clients arrivent au salon , alors que le coiffeur est endormi, et lancent chacun un signal(Sclient) qu'aucun ne parvient au Coiffeur ; ce qui constitue un manquement par rapport aux contraintes du problème.

(06 points)

Exercice 2 : Expliquez brièvement les différentes manières de créer un thread en java.

Réponse **:**

Pour créer un thread en Java, on peut utiliser l'une des deux méthodes suivantes :

1/'hériter de la classe Thread et de redéfinir la méthode run(),
2/ implémenter l'interface Runnable.

(4 Points)

Exercice 3 : Ecrire en langage Java une classe implémentant les sémaphores.

Réponse **:**

```
/* La classe Semaphore. */

public class Semaphore {
  private int value;

  public Semaphore(int n) {
    value = n;
  }

  public synchronized void wait() {
    while (value <= 0) {
      try { wait(); } catch (InterruptedException e) {};
    };
    value--;
  }

  public synchronized void signal() {
    value++;
    notifyAll();
  }
}
```

(06 points)

Exercice 4 : Dans un système, on trouve une seule classe de ressources de onze (11) instances. Cinq (05) processus partagent ces ressources. Chaque processus peut demander au maximum trois (03) ressources. Peut-on avoir un cas d'interblocage ? Justifiez.

Réponse :

Au pire, chaque processus a obtenu déjà 2 ressources, ce qui a mobilisé 10 ressources. Le premier qui demandera sa troisième ressource pourra terminer et rendre une ressource au moins; cette ressource pourra permettre à un autre processus de terminer, ...et ainsi de suite. Nous avons donc une séquence saine, et par conséquent l'état du système est sain et il n'y a aucun risque d'interblocage.

(02 points)

Comment pouvez-vous généraliser le résultat précédent ?.

Réponse :

On peut généraliser le résultat précédent. Soient N, T, M respectivement le nombre de processus, le nombre de ressources au maximum demandées par chaque processus et le nombre total de ressources. Pour que le système ne risque pas d'interblocage, il faudrait que : M soit égal au minimum à : $M = N(T - 1) + 1$.

(02 points)