Université Eshahid Hamma Lakhdar El-Oued
Faculté des sciences exactes
Département d'Informatique

Module : Système Exploitation II
Année : 3ème Année Informatique
Le 17/01/2024 – Durée : 1h30

Contrôle SE2

Exercice 1 [05 Points]

1) Un état d'interblocage est un état non sûr. ☐ Vrai ☐ Faux
2) L'absence de préemption signifie la possibilité de retirer de force une ressource à un processus qui la détient. . ☐ Vrai ☐ Faux
3) L'attente active peut provoquer un interblocage. ☐ Vrai ☐ Faux
4) Donner une brève description de l'algorithme de Peterson.
5) Citer brièvement les stratégies de traitement d'un interblocage.

Exercice 2 [07.50 Points]

On considère un système de cinq (05) processus P0 à P4 et 3 ressources : A,B et C avec les matrices : allocation et Max et le tableau disponible .

Processus	Allocation				MAX			Disponible		
	A	B	C		A	B	C	A	B	C
P0	1	4	1		5	6	2	2	0	0
P1	0	1	0		2	1	0			
P2	1	1	1		5	6	1			
P3	3	2	3		9	9	6			
P4	2	1	1		3	2	1			

1-Donner le tableau de nombre total des ressources.

2-Calculer la matrice Need.

3-Est-ce que l'état du système est un état sain ?

4-Si la requête du processus P1est (0, 4, 2), est-ce qu'on peut satisfaire la requête immédiatement ?

Exercice 3 [07.50 Points]

Des développeurs désirent réaliser une application de gestion de salle de révision dans une école. Dans cette partie, nous supposerons qu'il existe une seule **salle de révision**, qui peut être utilisée par **un seul groupe** d'étudiants à la fois. Un groupe d'étudiants a besoin de réserver une salle pour pouvoir réviser. Le groupe peut être composé de **2 à 5** personnes, au maximum, car la capacité de la salle est de **5** . Le premier étudiant d'un groupe qui demande à réserver la salle peut le faire si elle

est libre, sinon il attend que le groupe qui l'utilise la libère. Les autres membres du groupe utilisent la salle directement, si l'un de leurs camarades l'a déjà réservé.

L'un des développeurs propose le code suivant pour gérer ce scénario.

```
Salle : Semaphore (=5) ;

Processus Etudiant {
        Reserver_salle (){
            P(salle)
          <utliser_salle>
            V(salle)
 }
```

1) Quelles sont les conditions de bonne synchronisation des processus (bonne contrôle de section critique) qui ne sont pas respectées dans ce code ? avec justification.
2) Proposer une autre implémentation du code (fonction réserver_salle) de façon à corriger ce problème.

Bonne Chance

CORRECTION

Exercice 1 [05 Points]

1) Un état d'interblocage est un état non sûr. ◙Vrai ☐ Faux

0.75

2) L'absence de préemption signifie la possibilité de retirer de force une ressource à un processus qui la détient. . ◙ Vrai ☐ Faux

0.75

3) L'attente active peut provoquer un interblocage. ◙ Vrai ☐ Faux

0.75

4) Donner une brève description de l'algorithme de Peterson.

L'algorithme de Peterson est un algorithme d'exclusion mutuelle. Cet algorithme est basé sur une approche par attente active. Il est constitué de deux parties : le protocole d'entrée dans la section critique et le protocole de sortie. L'algorithme présenté est une version pouvant fonctionner avec deux threads. .

L'algorithme de Peterson nécessite les éléments et les informations suivantes : 1.50

- Chaque thread dispose d'un numéro l'identifiant, à savoir dans le cas avec deux threads les numéros 1 et 2

-Il faut disposer d'un tableau (dont chaque case correspond à un thread grâce à l'utilisation des numéros précités) et d'une variable. Le tableau pourra contenir deux valeurs, à savoir une valeur indiquant qu'un thread souhaite entrer en section critique ou y est et qu'un thread ne souhaite pas entrer en section critique. Par défaut aucun thread ne souhaite entrer dans la section critique. Un processus ne peut entrer en SC que si l'autre thread ne désire pas y accéder

5) Les stratégies de traitement d'un interblocage :

Ignorer complètement les problèmes d'interblocage - Les détecter et y remédier - Les éviter (allocation dynamique de ressources) -Les prévenir (empêcher l'apparition de l'une des quatre conditions)

1.25

Exercice 2 [07.50 Points]

Processus	Allocation				MAX			Disponible		
	A	B	C		A	B	C	A	B	C
P0	1	4	1		5	6	2	2	0	0
P1	0	1	0		2	1	0			
P2	1	1	1		5	6	1			

P3	3	2	3		9	9	6
P4	2	1	1		3	2	1

1-le tableau de nombre total des ressources.

Total = allocation + disponible = (9,9,6)

1.00

2-Calculer la matrice Need.

Need = Max – Allocation

Need =

	A	B	C
P0	1	4	1
P1	0	1	0
P2	1	1	1
P3	3	2	3
P4	2	1	1

2.00

3- L'état du système est un état sain ? on applique l'algorithme de Banquier

Disponible= (2,0,0)

Reste=[P0,P1,P2,P3,P4]

Etape 1 : Recherche dans la matrice Need : Need[i]<=Disponible , i=0..4

∋P1 : Need[1]=(2,0,0)<=(2,0,0)

Exécution de P1 donc : Disponible=Disponible+ Allocation[1]=(2,0,0)+(0,1,0)

Disponible= (2,1,0)

3.00

Reste=[P0,P2,P3,P4]

Etape 2 : Recherche dans la matrice Need : Need[i]<=Disponible , i=0,2..4

∋P4 : Need[4]=(1,1,0)<=(2,1,0)

Exécution de P4 donc : Disponible=Disponible+ Allocation[4]=(2,1,0)+(2,1,1)

Disponible= (4,2,1)

Reste=[P0,P2,P3]

Etape 3 : Recherche dans la matrice Need : Need[i]<=Disponible , i=0..3

∋P0 : Need[0]=(4,2,1)<=(4,2,1)

Exécution de P0 donc : Disponible=Disponible+ Allocation[0]=(4,2,1)+(1,4,1)

Disponible= (5,6,2)

Reste=[P2,P3]

Etape 4 : Recherche dans la matrice Need : Need[i]<=Disponible , i=2,3

∋P2 : Need[2]=(4,5,0)<=(5,6,2)

Exécution de P2 donc : Disponible=Disponible+ Allocation[2]=(5,6,2)+(1,1,1)

Disponible= (6,7,3)

Reste=[P3]

Etape 5 : Recherche dans la matrice Need : Need[i]<=Disponible , i=3

∋P3 : Need[3]=(6,7,3)<=(5,6,2)

Exécution de P3 donc : Disponible=Disponible+ Allocation[3]=(6,7,3)+(3,2,3)

Disponible= (9,9,6)

Reste=[ ]

Toutes les processus sont exécutés, donc l'état est sain selon la suite (P1,P4,P0,P2,P3)

4- la requête du processus P1est (0, 4, 2), est-ce qu'on peut satisfaire la requête immédiatement ?

Pour répondre à cette question, on doit d'abord vérifier que cette requête peut satisfaire avec les ressources disponibles : Req(P1) <=Disponible.

La condition est incorrecte car elle demande 4 ressources de B et 2 ressources de C qu'elles sont indisponibles : (0,4,2)>(2,0,0)

1.50

Donc : On ne peut pas satisfaire cette requête immédiatement.

Exercice 3 [07.50 Points]

1)La condition non respectée est : l'exclusion mutuelle.

Parce qu'un étudiant peut utiliser la salle (SC) en même temps qu'un étudiant d'un groupe différent : accès non autorisée à la section critique.

2.50

2)La solution proposée :

Const n=10 ; // Nombre de groupes

Salle : Semaphore(=1) //protège la salle qu'elle peut accèder par 1 seul groupe

Max : Semaphore[n]=(5,5,5,…,5) //max[i] vérifie le nombre max de salle de chaque groupe

mutex : semaphore[n]=(1,1,1,…,1) //mutex[i] protège la variable groupe[i]

int groupe[n]=(0 ;0 ;0….. ;0) //groupe[i]= nb etudiants d'un groupe i

Processus Etudiant

5.00

```
Reserver_salle(groupe_id)
P(mutex[groupe_id]) ;
groupe[groupe_id]++ ;
If (groupe[groupe_id]==1){      // le premier de même groupe, réserve la salle
        P(salle) ;
}
P(Max[groupe_id]) ;
V(mutex[groupe_id]) ;
```

<<utiliser_salle()>>

```
P(mutex[groupe_id]) ;
groupe[groupe_id]-- ;
If (groupe[groupe_id]==0)     // le dernier de même groupe
{V(salle) ;}
Else{
V(Max[groupe_id]) ;
}
V(mutex[groupe_id]) ;
}
```