

Examen semestriel

Module de « Systèmes d'exploitation2 »

Corrigé

Exercice 1 : (05 points)

Question 1 : Quelles critiques peut-on faire aux solutions matérielles du problème de l'exclusion mutuelle ?

Réponse : Ce sont des solutions spécifiques pour certains ordinateurs. Elles ne peuvent pas être appliquées dans tous les cas.

(1 point)

Question 2 : Les moniteurs sont considérés comme des outils de synchronisation "avancés" par rapport aux sémaphores. Expliquez brièvement pourquoi.

Réponse : Car les moniteurs cachent les détails de synchronisation. Le code des processus devient moins lourd.

(1 point)

Question 3 : Quelles critiques peut-on faire à l'algorithme du banquier ?

Réponse :

- ***Coûteux*** : L'algorithme est en effet très coûteux en temps d'exécution et en mémoire pour le système. Puisqu'il faut maintenir plusieurs matrices, et déclencher à chaque demande de ressource, l'algorithme de vérification de l'état sain qui demande $m \times n^2$ opérations. (m est le nombre de types de ressources et n est le nombre de processus).
- ***Théorique*** : L'algorithme exige que chaque processus déclare à l'avance les ressources qu'il doit utiliser, en type et en nombre. Cette contrainte est difficile à réaliser dans la pratique.
- ***Pessimiste*** : L'algorithme peut retarder une demande de ressources dès qu'il y a risque d'interblocage (mais en réalité l'interblocage peut ne pas se produire).

(2 points)

Question 4 : Présentez brièvement le principe de la méthode "détection-guérison" des interblocages.

Réponse :

Le principe de la méthode consiste à lancer périodiquement un algorithme de détection d'interblocage. Lorsque l'interblocage est détecté, on applique une méthode de guérison : manuelle, arrêt de processus ou réquisition de ressources.

(1 point)

Exercice 2 : (10 points)

Deux villes A et B sont reliées par une seule voie de chemin de fer.



Les règles de circulation sont les suivantes :

- La voie ne doit jamais être empruntée simultanément par deux trains allant en sens inverse
- La voie peut être empruntée par un ou plusieurs trains allant tous dans le même sens
- La priorité de parcours est la même pour les deux sens.

On considère deux classes de processus : les trains allant de A vers B : « **T-AB** » et les trains allant de B vers A : « **T-BA** ».

Processus T-AB

Début

```
Entree_A();  
<Circulation sur la voie de A vers B>  
Sortie_B();
```

Fin.

Processus T-BA

Début

```
Entree_B();  
<Circulation sur la voie de B vers A>  
Sortie_A();
```

Fin.

1) Quelle est la différence entre ce problème et le modèle des lecteurs/rédacteurs ?

Réponse :

La différence est que dans le modèle lecteur/rédacteur, on ne peut trouver qu'un seul rédacteur à la fois entrain d'occuper la ressource partagée (fichier) alors que dans ce problème la ressource (la voie) peut être occupée par plusieurs processus en même temps et ceci pour les deux classes T-AB et T-BA.

(1 point)

2) En utilisant les sémaphores, écrire les codes des quatre procédures *entree_A()*, *entree_B()*, *sortie_A()* et *sortie_B()* de façon à ce que les processus respectent les règles de circulation sur la voie. Précisez clairement vos déclarations et initialisations.

Déclarations et initialisations :

nbA : entier (init à 0) Compte le nombre de trains allant de A vers B

nbB : entier (init à 0) Compte le nombre de trains allant de B vers A.

S1 : Sémaphore (init à 1) pour permettre que des trains de sens contraire ne s'engagent pas dans la voie en même temps.

M1 : Sémaphore (init à 1) pour protéger la mise à jour en exclusion mutuelle de la variable protégée nbA

M2 : Sémaphore (init à 1) pour protéger la mise à jour en exclusion mutuelle de la variable protégée nbB

(1 point)

Processus T-AB**Début**

```

P(S1)
P(M1)
nbA++
si (nbA==1) alors P(S2) fsi
V(M1)
V(S1)

```

<Circulation sur la voie de A vers B>

```

P(M1)
nbA - -
si (nbA==0) alors V(S2) fsi
V(M1)

```

Fin.**Processus T-BA****Début**

```

P(S1)
P(M2)
nbB++
si (nbB==1) alors P(S2) fsi
V(M2)
V(S1)

```

<Circulation sur la voie de B vers A>

```

P(M2)
nbB - -
si (nbB==0) alors V(S2) fsi
V(M1)

```

Fin.

(3 points)

3) Expliquer pourquoi la solution suivante (avec moniteurs) n'est pas correcte.

Moniteur AB ; Int nbA=0, nbB=0 ; Condition ca, cb ;	
Entree_A() { nbA++ ; si (nbB>0) alors ca.wait() fsi }	Entree_B() { nbB++ ; si (nbA>0) alors cb.wait() fsi }
Sortie_B() { nbA-- ; si (nbA==0) alors cb.signal() fsi }	Sortie_A() { nbB - - ; si (nbB==0) alors ca.signal() fsi }

Réponse :

L'erreur dans la solution donnée est qu'en utilisant un seul compteur (nbA et nbB) pour chaque classe on n'arrivera pas à différencier le nombre de processus entrain d'utiliser la voie de celui qui est attente. Et cette solution peut induire un problème d'interblocage. comme par exemple le cas où A1, B1, A2 arrivent successivement, à la sortie de A1, nbA sera égal à 1 et donc B1 reste bloqué sur cb en attente d'être débloqué et A2 reste bloqué sur ca en attente d'être débloqué par B1.

(2 points)

4) Donnez une correction de la solution erronée.

M = Moniteur**Var****Int nbA, nbB, attA, attB;****Condition ca, cb ;****Procedure Entry Entree_A()****Begin** **si (nbB>0) alors attA++ ; ca.wait() fsi** **nbA++ ;** **si (attA>0) alors attA-- ; ca.signal() ; fsi****End;**

Procedure Entry Sortie_B()

```
Begin
  nbA-- ;
  si (nbA==0) alors
    si (attB>0) alors attB-- ; cb.signal() fsi
  fsi
End;
```

Procedure Entry Entree_B()

```
Begin
  si (nbA>0) alors attB++ ; cb.wait() fsi
  nbB++ ;
  si (attB>0) alors attB-- ; cb.signal() ; fsi
End;
```

Procedure Entry Sortie_A()

```
Begin
  nbB-- ;
  si (nbB==0) alors
    si (attA>0) alors attA-- ; ca.signal() fsi
  fsi
End;
```

```
Begin /*Initialisation */
  nbA=0, nbB=0, attA=0, attB=0 ;
End.
```

(3 points)

Exercice 3 : (05 points)

Dans un système il y a 4 processus (Pi, i=0..4) qui partagent 6 types de ressources (Ri, i=0..6). Le système possède une seule ressource de chaque type.

L'état courant du système est décrit par les matrices d'allocation (ALLOCATION) et de besoin (NEED) suivantes, ainsi que le vecteur de disponibilité instantanée (AVAILABLE)=[0 1 0 0 0 0] :

ALLOCATION =

	R1	R2	R3	R4	R5	R6
P1	1	0	1	0	0	0
P2	0	0	0	1	0	0
P3	0	0	0	0	1	0
P4	0	0	0	0	0	1

NEED=

	R1	R2	R3	R4	R5	R6
P1	0	1	0	0	0	0
P2	0	1	1	0	0	0
P3	1	1	0	0	0	0
P4	0	1	1	1	0	0

1. L'état courant est-il sûr (sain) ? Justifier.

Réponse :

En appliquant l'algorithme de vérification de l'état sain, on trouve que le vecteur Finsh est entièrement égal à Vrai :

V	V	V	V
---	---	---	---

L'état du système est donc sain.

(1 point)

2. Si P3 demande une ressource R2 peut on lui accorder immédiatement sa requête? Justifier.

Réponse :

On applique l'algorithme du banquier :

Etape 1 : Request3 <=Need3 ? oui

Etape2 : Request3<=Available ? oui

Etape3 : Sauvegarde de l'état du système.

Modification des matrices

Algorithme de vérification de l'état sain :

Work

0	0	0	0	0	0
---	---	---	---	---	---

Finish

F	F	F	F
---	---	---	---

Le vecteur Finish final n'est pas entièrement fégal à Vrai .

L'état du système n'est pas sain. Annuler les modifications sur les matrices. On ne pas satisfaire la requête immédiatement.

(2 points)

3. Si P1 demande une ressource R2 peut on lui accorder immédiatement sa requête? Justifier.

Réponse :

On applique l'algorithme du banquier :

Etape 1 : Request1 <=Need3 ? oui

Etape2 : Request1<=Available ? oui

Etape3 : Sauvegarde de l'état du système.

Modification des matrices

Algorithme de vérification de l'état sain :

Work

0	0	0	0	0	0
---	---	---	---	---	---

Finish

F	F	F	F
---	---	---	---

I=0

Work

1	1	1	0	0	0
---	---	---	---	---	---

Finish

V	F	F	F
---	---	---	---

I=2

Work

1	1	1	1	0	0
---	---	---	---	---	---

Finish

V	V	F	F
---	---	---	---

I=3

Work

1	1	1	1	1	0
---	---	---	---	---	---

Finish

V	V	V	F
---	---	---	---

I=4

Work

1	1	1	1	1	1
---	---	---	---	---	---

Finish

V	V	V	V
---	---	---	---

Le vecteur Finish final est entièrement fégal à Vrai .

L'état du système est sain. La requête peut être accordée.

(2 points)