+

Révision système 2022

**Exercice N°1**

Soit la définition suivante d’un sémaphore :

**P(s)** : **début**

Tant que e(s) <= 0 faire **rien (attente active)**

e(s) : = e(s) -1 ;

**Fin**

1. Quel est l’inconvénient de cette définition P(s) par rapport à celle vue en cours ?

P(s) :

E(s)=e(s)-1

Si e(s)<0 bloqué un processus dans la fille d’attente

2- Quelle est la différence entre les primitives de sémaphores (P(s) et V(s) et

Les primitives de moniteurs wait (c), signal (c) ?

**Solution exercice**

1/inconvénient de cette définition c’est le problème de l’attente active, les sémaphores sont utilisés pour éviter l’attente active

2/la fonction p(s) bloque un processus si la valeur de sémaphore <0 par contre wait() bloque automatiquement un processus qui l’exécute

La même chose pour v(s) débloque un processus si la valeur de sémaphore e(s) <=0, par contre signale réveil un processus automatiquement

**Exercice 2**

**Vous avez le programme suivant :**

#include <stdio.h>

#include <unistd.h>

int x = 0;

int main()

{

x++ ;

fork(); // père et fils1 (x=1)

if (fork()!=0) { père et fils1// père ( !=0,x=1) -> fils2 (=0,x=1) / fils1( !=0,x=1) -> fils11(=0,x=1)

if (fork()==0) { père ( !=0,x=1) -> fils3 (=0,x=1) // fils1( !=0,x=1)-> fils12(=0,x=1)

fork(); //fils3(x=1)-> fils31(x=1) /fil12(x=1)-> fils121(x=1)

x++;fls3 x=2, fils31 x= 2,fils12 x=2, fils 121 x=2

}

x++; fls3 x=3, fils31 x= 3,fils12 x=3, fils 121 x=3,père x=2, fils x=2,fil11 x=2, fils x=2

}

printf(" x = %d\n", x);

return 0;

}

a) Combien de processus le programme crée-t-il lors de son exécution.

B) Dessinez l'arbre de processus et indiquez la valeur de x

**Solution exercice N°2**

X=1,x=2

**Exercice N°3**

1- d’après le graphe d’allocation des ressources :

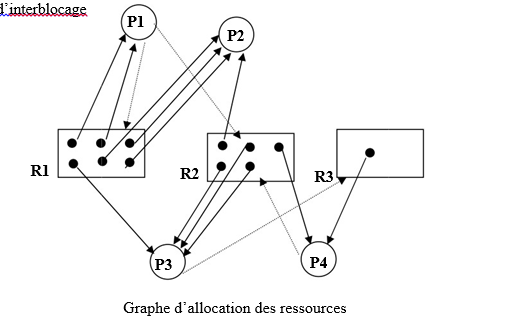
- déduire le vecteur des ressources totales dans le système.

- déduire la matrice d’allocation des ressources de système.

- Déduire la matrice requête

- Déduire le vecteur des ressources disponible dans le système.

2-appliquer l’algorithme de banquier pour détecter une situation d’interblocage



**Solution exercice 3**

**Vecteur total (r1=6, r2=5, r3=1)**

|  |  |  |  |
| --- | --- | --- | --- |
|  | R1 | R2 | R3 |
| P1 | 1 | 1 | 0 |
| P2 | 0 | 0 | 0 |
| P3 | 0 | 0 | 1 |
| P4 | 0 | 1 | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | R1(6) | R2(5) | R3(1) |
| P1 | 2 | 0 | 0 |
| P2 | 3 | 1 | 0 |
| P3 | 1 | 3 | 0 |
| P4 | 0 | 1 | 1 |

**Matrice requête**

**Matrice allocation**

**Vecteur disponible = total – allocation = (6 5 1 ) – ( 6 5 1)=(0 0 0)**

**Détection d’inter blocage :**

**Req p1(1 1 0)<= disponible(0 0 0) non**

**Req p2(0 0 0) <= disponible (0 0 0) oui => p2 termine son exécution, disponible = disponible + allocation p2 = (0 0 0) + (3 1 0)= (3 1 0)**

**Req p3 (0 0 1 ) <= disponible(3 1 0) non**

**Req p4(0 1 0) <= disponible (3 1 0) oui => p4 termine son exécution, disponible = disponible + allocation p4 = (3 1 0) + (0 1 1 ) = (3 2 1 )**

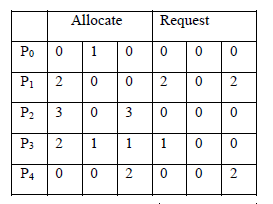
**Req p1(1 1 0 ) <= disponible (3 2 1) oui => p1 termine son exécution, disponible = disponible+ allocation p1= (3 2 1 ) + (2 0 0)=(5 2 1 )**

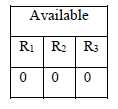
**Req p3(0 0 1 )<= disponible (5 2 1)oui => p3 termine son exécution, disponible = disponible+ allocation p3=(5 2 1 )+ (1 3 0)=(6 5 1) =total**

**Pas d’interblocage : etat sain = p2 -> p4 -> p1 -> p3**

**Exercice N°4**

Soit un système utilisant la **détection** d’interblocages À l’instant **t1** l’état du système est présenté par les matrices suivantes :





Disponible

1. Calculer vecteur des ressources totale. (7 2 6)

2. Déterminer une suite d’exécution des processus pour que l’état du système soit sain (sans interblocage).

3. Supposons que P2 fait la demande requête p2 = [0 0 1] :

*-* Calculer la nouvelle valeur de vecteur disponible**.**

*-* Le système reste-t-il toujours en état sain ? Si oui donner la suite saine sinon donner les processus qui sont en interblocages.

**Solution exercice 4**

**1-Rmax =** Available+ΣAllocate(Pi,\*)= Available+ Allocate(P0,\*)+ Allocate(P1,\*)+ Allocate(P2,\*)+ Allocate(P3,\*)+Allocate(P4,\*)+Allocate(P5,\*)=

(0,0,0)+(0,1,0)+(2,0,0)+(3,0,3)+(2,1,1)+(0,0,2)=(7,2,6);

**2-** Algorithme **TestSain** pour déterminer une suite saine (Au départ tous les processus sont non marqué, à chaque étape on va marquer (sélectionner) le premier processus Pi vérifiant l'équation request (Pi,\*)<= available )

**Etape 01 :** request(P0,\* ) =(0,0,0)<= Available **=(0,0,0) alors**

Available= Available+Allocate(P0,\*) = (0,0,0)+(0,1,0)=(0,1,0) ;

Marquer (barrer) P0. La suite saine : S={ P0};

**Etape 02 :** request(P2,\* ) =(0,0,0)<= Available **=(0,1,0); alors**

Available= Available+Allocate(P2,\*)=(0,1,0)+(3,0,3)=(3,1,3).

Marquer (barrer) P2 et ajouter P2 dans la suite saine : S={ P0, P2};

**Etape 03 :** request(P1,\* ) =(2,0,2)<= Available **=(3,1,3); alors**

Available= Available+Allocate(P1,\*)=(3,1,3)+(2,0,0)=(5,1,3).

Marquer (barrer) P1 , S={ P0, P2, P1};

**Etape 04 :** request(P3,\* ) =(1,0,0)<= Available **=(5,1,3); alors**

Available= Available+Allocate(P3,\*)=(5,1,3)+(2,1,1)=(7,2,4).

Marquer (barrer) P3 , S={ P0, P2, P1, P3 };

**Etape 05 :** request(P4,\* ) =(0,0,2)<= Available **=(7,2,4); alors**

Available= Available+Allocate(P3,\*)=(7,2,4)+(0,0,2)=(7,2,6)=Rmax

Marquer (barrer) P4 , S={ P0, P2, P1, P3, P4 };

**La Suite Saine est : S={ P0, P2, P1, P3, P4 }**

3- P2 fait la demande request p2 = [0 0 1], et on a Rmax **=** Available+ΣAllocate(Pi,\*) -> Available **=** Rmax -ΣAllocate(Pi,\*) alors pas de relation entre Request et Available ainsi Available ne changera pas -> Available =(0,0,0).

Pour vérifier si le système reste-t-il toujours en état sain, on applique l'algorithme Testsain. Ainsi en "Etape 01" on a request(P0,\* ) =(0,0,0)<= Available =(0,0,0) et

Available= Available+Allocate(P0,\*) = (0,0,0)+(0,1,0)=(0,1,0) ->Available = (0,1,0).

En "Etape 02" in n'exista pas un processus Pi qui vérifier request (Pi,\*)<= available=(0,1,0) ainsi l'algorithme Testsain s'arrête à la 2ème étape et les processus P1 , P2 ,P3 et P4 sont en interbloquage.

**Exercice N°5**

Un stade d’athlétisme peut recevoir les athlètes de trois (3) clubs A, B et C qui viennent s’y entraîner. Pour organiser les entraînements, on impose la règle suivante :

A un instant donné, Dans le stade il doit y avoir au Max 2 clubs et N athlètes par club.

Soit nbA, nbB et nbC le nombre d’athlètes des clubs A, B et C ayant accédé au stade (en vérifiant la condition précédente)

Un athlète d’un club X (X=A ou B ou C) sera bloqué lorsqu’il y a déjà N athlètes de son club ou des athlètes des deux autres clubs.

1/On vous demande de proposer un schéma de synchronisation des processus: Processus A, Processus B et Processus C correspondant respectivement à des athlètes des clubs A, B et C, et ce en utilisant des moniteurs.

Déclarez clairement vos variables et précisez leurs initialisations.

**Solution exercice 5**

Monotor Stade {

int nbA=0, nbB=0, nbC=0 ;

cond Astop, Bstop, Cstop ;

public void entrerA(){

while(nbA==N || nbB\*nbC!=0)

wait(Astop) ;

nbA++ ;

}

public void sortieA(){

nbA-- ;

signal(Astop)

if (nbA==0) { signal (Bstop) ; signal(Cstop);}

} // de même pour B et C

}

**Exercice N°6**

On considère trois processus P1, P2 et P3. Le processus P1 produit des messages qu'il dépose dans un tampon T1. P2 prélève les messages contenus dans T1, les traite puis dépose les résultats dans un tampon T2. P3 prélève les messages contenus dans T2 et les consomme.

1) Ecrire les algorithmes de P1, P2 et P3 de façon à garantir le non-interblocage.

2) On considère maintenant que les Pi (i=1..3) travaillent sur le même tampon T (au lieu de T1 et T2). Réétudierd la question 1).

**Solution exercice 6**

S1,S2,S3,S4,mutex1,mutex2 : sémaphore init n1,0,n2,0,1,1 ; /\* ni=taille de Ti, i=1,2 \*/

**Processus P1**

**While (true) do** {

<Produire un message>

P(S1) ;

P(mutex1) ;

<Dépôt du message dansT1> ;

V(mutex1)

V(S2) ;

}

**Processus P2**

**While (true) do{**

P(S2) ;

P(mutex1) ;

<Prélever un message deT1> ;

V(mutex1) ;

V(S1) ;

<Traiter le message> ;

P(S3) ;

P(mutex2) ;

<Dépôt du résultat dans T2> ;

V(mutex2) ;

V(S4) ;

}

**Processus P3**

**While (true) do**

{P(S4) ;

P(mutex2) ;

<Prélever un message de T2> ;

V(mutex2) ;

V(S3) ;

<Traitement du message> ;

}

**Exercice N° 7 examen 2014**

**Vous avez le système suivant :**

**R1 R2 R3**

**P1** 3 3 0

**P2** 3 0 0

**P3** 3 0 3

**P4** 0 3 0

**Allocation**

**Q1 :** calculer la matrice besoin

**R1 R2 R3**

**P1** 6 6 3

**P2** 3 3 3

**P3 X Y Z**

**P4** 3 6 6

**Max**

**R1 R2 R3**

0 3 3

**Disponible**

**Q2 :** Quelle est la valeur maximale demander par P3 (x,y,z) pour laquelle l’état de système reste stable et sain (en appliquant l’algorithme de banquier pour trouver x, y et z).

**Q3 :** Combien faut-il ajouter de ressources (pour chaque Ri) de manière à ce que les processus puissent s’exécuter sans contrainte (en prendre en considération la valeur de x, y et z trouves dans la question 2 : matrice Max).

**Q4 :** Si le processus P1 déposé la requête (0, 2, 1), cette requête est-elle acceptée immédiatement ? (Justifiez votre réponse).

**Q5** : quelle est le nombre totale des exemplaires de ressources R1, R2 et R3 dans ce système ?

**Solution Exercice N° 7**

**Vous avez le système suivant :**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **R1** | **R2** | **R3** |
| **P1** | 6 | 6 | 3 |
| **P2** | 3 | 3 | 3 |
| **P3** | **X** | **Y** | **Z** |
| **P4** | 3 | 6 | 6 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **R1** | **R2** | **R3** |
| **P1** | 3 | 3 | 0 |
| **P2** | 3 | 0 | 0 |
| **P3** | 3 | 0 | 3 |
| **P4** | 0 | 3 | 0 |

|  |  |  |
| --- | --- | --- |
| **R1** | **R2** | **R3** |
| 0 | 3 | 3 |

**Disponible**

**Max**

**Allocation**

**Q1 :** calculer la matrice besoin

|  |  |  |  |
| --- | --- | --- | --- |
|  | **R1** | **R2** | **R3** |
| **P1** | 3 | 3 | 0 |
| **P2** | 0 | 3 | 3 |
| **P3** | **X-3** | **Y** | **Z-3** |
| **P4** | 3 | 3 | 6 |

**Q2 :** Quelle est la valeur maximale demander par P3 (x,y,z) pour laquelle l’état de système reste stable et sain (en appliquant l’algorithme de banquier pour trouver x, y et z).**(1 pts)**

Application l’algorithme de banquier

X=9, y =6, z=6

**Q3 :** Combien faut-il ajouter de ressources (pour chaque Ri) de manière à ce que les processus puissent s’exécuter sans contrainte (en prendre en considération la valeur de x, y et z trouves dans la question 2 : matrice Max). **(1 pts)**

Utilisation de la matrice besoin

|  |  |  |  |
| --- | --- | --- | --- |
|  | **R1** | **R2** | **R3** |
| **P1** | 3 | 3 | 3 |
| **P2** | 0 | 3 | 3 |
| **P3** | **6** | **6** | **3** |
| **P4** | 3 | 3 | 6 |

Besoin de r1 =3+0+6+3=12

Besoin de r2 =3+3|+6+3=15

Besoin de r3 =3+3+3+6= 15

Ressource ajouter = besoin –disponible

=(12,15,15)-(0,3,3)=(12,12,12)

**Q4 :** Si le processus P1 déposé la requête (0, 2, 1), cette requête est-elle acceptée immédiatement ? (Justifiez votre réponse). **(1 pts)**

La requête de p1 est acceptée

P1(0,2,1) < = disponible(0,3,3)

P1((0,2,1)< = besoin p1(3,3,3)

Disponible = disponible-requête p1= (0,3,3)-(0,2,1)=(0,1,2)

Besoin p1 =besoin p1 –requête p1=(3,3,3)-(0,2,1)= (3,1,2)

Allocation p1 = allocation p1 + requête p1=(3,3,0)+(0,2,1)=(3,5,1)

**Q5** : quelle est le nombre totale des exemplaires de ressources R1, R2 et R3 dans ce système ? **(0,5 pts)**

Ressources totale = (r1=9, r2=6,r3=6)