Université de Djelfa
Faculté des sciences exactes et informatique
3ème année licence informatique

L'interblocage de processus **Série de TD n°2** **3 LI module SEII**

Exercice 01 : Soit un système utilisant la détection d'interblocage. À l'instant t1 l'état du système est le suivant :

1. Calculer Rmax.
2. Quelle est la valeur maximale de la requête [X Y] de P3 pour laquelle l'état du système est sain ?
3. Après avoir fixé la valeur maximale de Req3, représenter le graphe d'allocation de ressources (*graphe d'état*) du système.
4. À l'instant t2 suivant, le processus P2 obtient la ressource qu'il demande, puis se termine et libère toutes les ressources qu'il détient. À quel processus l'allocateur devrait allouer ces ressources pour maintenir le système dans un état sain ?

	R_1	R_2
P_1	1	0
P_2	0	1
P_3	1	1
P_4	2	0

Allocate

	R_1	R_2
P_1	2	2
P_2	1	0
P_3	X	Y
P_4	1	2

Request

R_1	R_2
1	1

Available

Exercice N°2: Soit un système utilisant l'algorithme du Banquier. À l'instant t1 considéré, l'état du système et défini comme suit :

1. Calculer Rmax et Need.
2. Quelle est la valeur maximale de l'Annonce de P3 pour laquelle l'état du système est fiable ? (Justifier votre réponse).
3. En fixant l'Annonce de P3, à celle obtenue à la question 2, le processus P2 fait la requête [0 1 1]. Peut-on satisfaire immédiatement cette requête ? (Justifier votre réponse).

	R_1	R_2	R_3
P_1	2	2	1
P_2	1	1	1
P_3	X	Y	Z
P_4	1	2	2

Annonce

	R_1	R_2	R_3
P_1	1	1	0
P_2	1	1	1
P_3	1	0	1
P_4	0	1	0

Allocate

R_1	R_2	R_3
0	1	1

Available

Exercice N°3:

Soit l'état suivant d'un système à plusieurs classes de ressources :

	R_1	R_2	R_3	R_4
P_1	0	0	1	2
P_2	1	0	0	0
P_3	1	3	5	4
P_4	0	6	3	2
P_5	0	0	1	4

Allocate

	R_1	R_2	R_3	R_4
P_1	0	0	1	2
P_2	1	7	5	0
P_3	2	3	5	6
P_4	0	6	5	2
P_5	0	6	5	4

Annonce

R_1	R_2	R_3	R_4
3	14	11	12

Rmax

1. Calculer la matrice Need. (*Need = Annoce - Allocate*)
2. Montrer que l'état du système est fiable en trouvant une séquence fiable.
3. Si une demande du processus P_2 [0 4 2 0] arrive, doit-on la satisfaire immédiatement ?

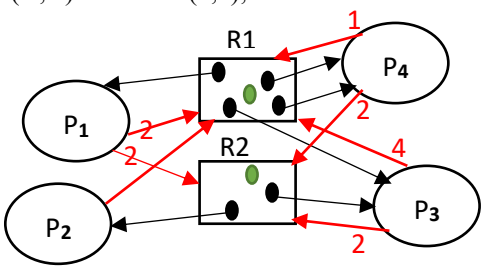
Solution exercice 01 :

1- Rmax = available+ $\sum$allocate(Pi,∗) =(1,1)+(1,0)+(0,1)+(1,1)+(2,0) =(5,3);

2- Maximum de request(P3,*)=[x, y] ? On applique l'algorithme état sain étape par étape, lorsque on trouve pas un Pi vérfiant request(Pi,*) <=Available alors on prend les valeurs maximales de x et y pour satisfaire l'équation : request(P3,*) <=Available.

- Etape 1 : choix de P2 : request(P2,*)=(1,0)<Avaible=(1,1);
  Available =(1,1)+alocate(P2,*)=(1,1)+(0,1) =**(1,2)**;
- Etape 1 : choix de P4 : request(P4,*)=(1,2)<Avaible=(1,2);
  Available =(1,2)+alocate(P4,*)=(1,2)+(2,0) =**(3,2)**;
- Etape 1 : choix de P1 : request(P1,*)=(2,2)<Avaible=(3,2);
- Available =(3,2)+alocate(P1,*)=(3,2)+(1,0) =**(4,2)**;
- Etape 2 : choix de P3 : request(P3,*)=(X,Y)<Avaible=(4,2);
  [x,y]=[4,2]

3- Graphe de l'allocation des ressources :



4- À l'instant t2 suivant :

- le processus P2 obtient la ressource qu'il demande ➔ **available = available-request(P_2,*)= (1,1)-(1,0)=(0,1) && allocate(P2,*)= allocate(P2,*)+ request(P2,*)=(0,1)+(1,0)=(1,1);**
- Puis P2 se termine et libère toutes les ressources qu'il détient ➔ **available = available-allocate(P_2,*)=(0,1)+(1,1)=(1,2);**

  Pour détermine le prochain processus auquel l'allocateur devrait allouer ces ressources pour maintenir le système dans un état sain on 02 solutions :
  - Solution 01 : On modifier le graphe d'allocation, l'allocateur devrait allouer ces ressources pour maintenir le système dans un état sain au processus dont le système peut satisfaire leur demande (01 unité de R1 et 02 unités de R2 seront disponibles après la fin de P2). Le système peut satisfaire ainsi la demande (request (P_4,*)=(1,2)) de processus P4.
  - Solution 02 : on applique l'algorithme Testsain sur le nouvel état de système (sans P2) ainsi le processus à choisir sera le premier processus marqué :

    Etape 01 : seulement le processus P4 sera marqué parceque request (P4,*)<=available=(2,2);

Solution exercice 02:

1- **Rmax = available+ allocate(P1,*)+…+ allocate(P4,*)= (0,1,1)+(3,3,2)=(3,4,3);**
Need = annonce(Pi,*)-allocate(Pi,*) =

	R_1	R_2	R_3
P_1	2	2	1
P_2	1	1	1
P_3	X	Y	Z
P_4	1	2	2

Annonce

−

	R_1	R_2	R_3
P_1	1	1	0
P_2	1	1	1
P_3	1	0	1
P_4	0	1	0

Allocate

=

	R_1	R_2	R_3
P_1	1	1	1
P_2	0	0	0
P_3	x-1	y	z-1
P_4	1	1	2

need

2- **Max de annonce(P3,*)= [x,y,z] pour que le système soit en état faible (pas d'inteblocage)**
On applique l'algorithme de Banquier :
Etape01: choix de P2, Available= (0,1,1)+(1,1,1)=(1,2,2);
Etape02: choix de P1, Available= (1,2,2)+(1,1,0)=(2,3,2);
Etape03: choix de P4, Available= (2,3,2)+(0,1,0)=(2,4,2);
Etape04: choix de P3, need(p3,*) =(x-1,y,z-1)<= (2,4,2) ➔ max(w,y,z) =(3,4,3)

3- **P2 fait la requête [0,1,1] ,**

On ne peut pas satisfaire la demande de P2 immédiatement puisque le requête : request (p2,*)= (0,1,1)> need(P2,*)=(0,0,0);

Solution exercice 03:

1. Calculer la matrice Need. (*Need = Annoce - Allocate*)

Need =

	R_1	R_2	R_3	R_4
P_1	0	0	0	0
P_2	0	7	5	0
P_3	1	0	0	2
P_4	0	0	2	0
P_5	0	6	4	0

2. Montrer que l'état du système est fiable en trouvant une séquence fiable
   Tous d'abord on calcul : Available = Rmax -∑Allocate(Pi,*) = (3,14,11,12)-(2,9,10,12)=**(1,5,1,0)**

On applique l'algorithme de Banquier :
Etape01: choix (marquer) de P1, Available= (1,5,1,0) +(0,0,1,2)=(1,5,2,2) ; F={P1}
Etape02: choix de P3, Available= (1,5,2,2) +(1,3,5,4) =(2,8,7,6); F={P1,P3}
Etape03: choix de P2, Available= (3,8,7,6) +(1,0,0,0) =(3,8,7,6); F={P1,P3,P2}
Etape04: choix de P4, Available= (4,8,7,6) +(0,6,3,2) =(3,14,10,8); F={P1,P3,P2,P4,}
Etape05: choix de P5, Available= (4,14,10,8) +(0,0,1,4) = (3,14,11,12) = Rmax; F={P1,P3,P2,P4,P5}

Tous les processus sont marqués et la une séquence fiable F={P1,P3,P2,P4,P5}

3. Si une demande du processus P2 [0 4 2 0] arrive, doit-on la satisfaire immédiatement ?
4. On a request (P2, *) = (0,4,2,0) et Available =(1,5,1,0) alors on ne peut pas être satisfait par ce que request (P2,R3)=2 n'est pas inférieure de Available(R3)=1.