UNIVERSITE A.MIRA-BEJAIA
FACULTE DES SCIENCES EXACTES
DEPARTEMENT INFORMATIQUE

Série 3 : Interblocage

Exercice1:

1) Bes[i,*]=Max[i,*]-Alloc[i,*] Disp[j]=Init[j]-∑Alloc[*,j] i:n° ligne (processus), j:n° colonne (resource).

Processus	Allocation				Max				Bes				Disp			
	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
P1	0	1	1	0	0	2	1	0	0	1	0	0	1	5	1	0
P2	1	2	3	1	1	6	5	2	0	4	2	1				
P3	1	3	6	5	2	3	6	6	1	0	0	1				
P4	0	6	3	2	1	6	5	2	1	0	2	0				
P5	0	0	1	4	0	6	5	6	0	6	4	2				

2) Une séquence saine est une suite de processus qui peuvent s'exécuter (en ayant le max de ressources) et terminer. Donc il faut avoir Bes[i,*]≤Disp[*]. Ici le seul processus qui vérifie cette relation est P1 (Bes[1,*]<Disp[*])

3) Le système est dans un état sain si on peut trouver une séquence saine complète (avec tous les processus).

P1 se termine et libère les ressources qu'il a acquis → Disp=[1 6 2 0]

Bes[4,*]<Disp[*] → P4 peut terminer et libérer ses ressources → Disp=[1 12 5 2]

P2, P3 et P5 peuvent se terminer ! Donc la séquence peut être : P1→P4→P2→P3→P5

Il existe plusieurs séquences possibles mais elles commencent toutes par P1 puis P4.

Exercice2 :

a) Init[j]=Disp[j]+Alloc[*,j] →Init[R1]=Disp[R1]+Alloc[*,R1]→Init[R1]=5+(0+0+5)=10

Même raisonnement : Init[R2]=2+(0+1+1)=4 ; Init[R3]=1+(0+2+1)=4 ;

Init[R4]=1+(0+2+1)=4 →Init=[10 4 4 4]

b)

Processus	Allocation				Max				Bes				Disponible			
	R1	R2	R3	R4	R1	R2	R3	R4	R1	R2	R3	R4	R1	R2	R3	R4
P1	0	0	0	0	4	2	3	1	4	2	3	1	5	2	1	1
P2	0	1	2	2	2	1	4	4	2	0	2	2				
P3	5	1	1	1	5	1	2	2	0	0	1	1				

Même raisonnement que pour exerice1 : P3 peut commencer la séquence

P3 se termine et libère ses ressources : Disp=[10 3 2 2]

P2 peut se terminer et libérer ses ressources : Disp=[10 4 4 4] (égale à Init !)

Donc P1 peut se terminer donc la séquence saine complète (unique !) : P3→P2→P1

c) Algorithme du banquier appliqué à chaque requête dans l'ordre :

1) P1 demande 2 instances de R4→Req[1,4]=2 :

i) Requête valide si Req[i,j]≤Bes[i,j]? Req[1,4](=2)>Bes[1,4](=1] →Requête refusée passer à la prochaine requête.

2) Req[3,3]=1

i) Req[3,3](=1)=Bes[3,3](=1)→Requête valide continuer

ii) Requête satisfaisable si Req[i,j]≤Disp[j] ? Req[3,3](=1)=Disp[3](=1)→Requête satisfaisable continuer

iii) Créer un état virtuel

Processus	Allocation				Max				Bes				Disponible			
	R1	R2	R3	R4	R1	R2	R3	R4	R1	R2	R3	R4	R1	R2	R3	R4
P1	0	0	0	0	4	2	3	1	4	2	3	1	5	2	0	1
P2	0	1	2	2	2	1	4	4	2	0	2	2				
P3	5	1	2	1	5	1	2	2	0	0	0	1				

iv) état virtuel sain ? même raisonnement que pour la question b : P3→P2→P1 donc état virtuel sain continuer

v) état virtuel devient réel : requête satisfaite et ressource allouée. La prochaine requête se basera sur le nouvel état réel.

Processus	Allocation				Max				Bes				Disponible			
	R1	R2	R3	R4	R1	R2	R3	R4	R1	R2	R3	R4	R1	R2	R3	R4
P1	0	0	0	0	4	2	3	1	4	2	3	1	5	2	0	1
P2	0	1	2	2	2	1	4	4	2	0	2	2				
P3	5	1	2	1	5	1	2	2	0	0	0	1				

3) Req[2,4]=1

i) Req[2,4](=1)<Bes[2,4](=2)→Requête valide continuer

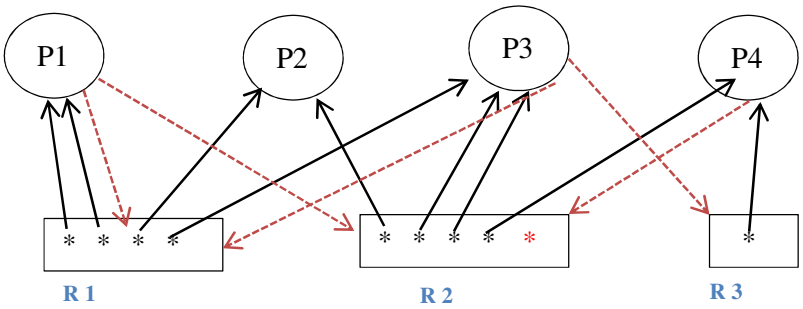
ii) Req[2,4](=1)=Disp[4](=1)→Requête satisfaisable continuer

iii) état virtuel

Processus	Allocation				Max				Bes				Disponible			
	R1	R2	R3	R4	R1	R2	R3	R4	R1	R2	R3	R4	R1	R2	R3	R4
P1	0	0	0	0	4	2	3	1	4	2	3	1	5	2	0	0
P2	0	1	2	3	2	1	4	4	2	0	2	1				
P3	5	1	2	1	5	1	2	2	0	0	0	1				

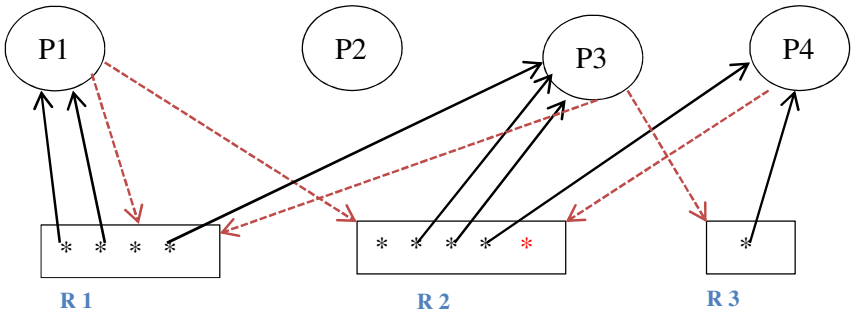
vi) état virtuel non sain ! il n'existe aucune séquence saine (P3 ne peut plus commencer la séquence et donc ni P2 ni P1 ne peuvent se terminer) →Requête retardée passer à la prochaine requête.

Exercice3 : 1) Graphe d'allocation

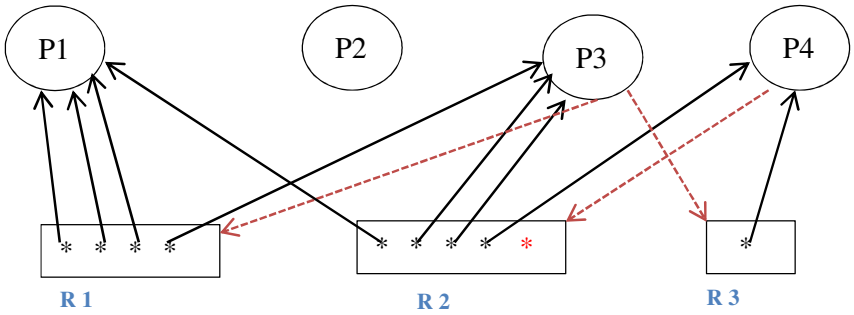


2) Réduction de graphe :

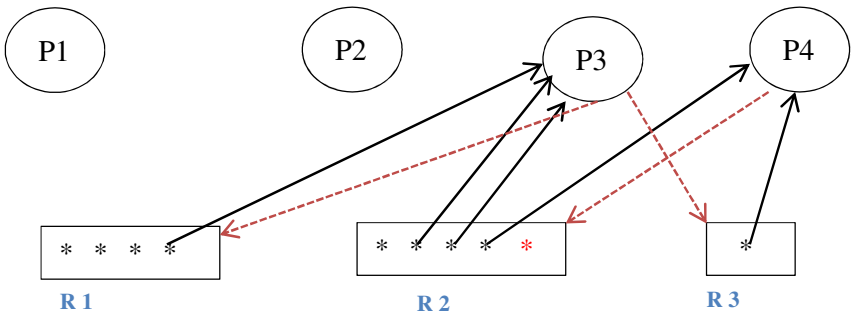
1) P2 n'a pas d'arcs de requêtes →effacer ses arcs d'allocation



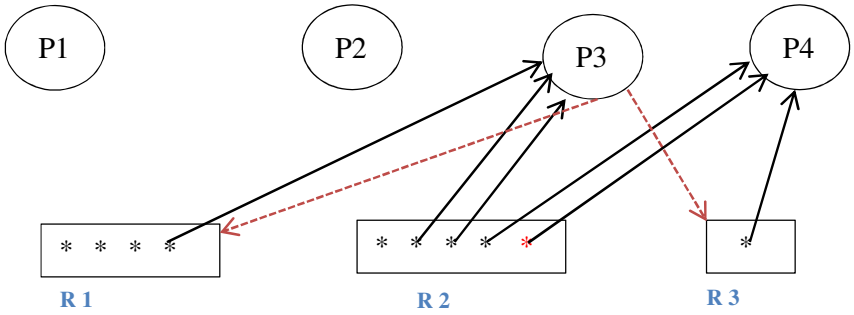
2) P1 a un arc de requête vers une instance de R1 qui est disponible et un arc de requête vers une instance de R2 qui est disponible → changer ses arcs de requêtes en arcs d'allocations.



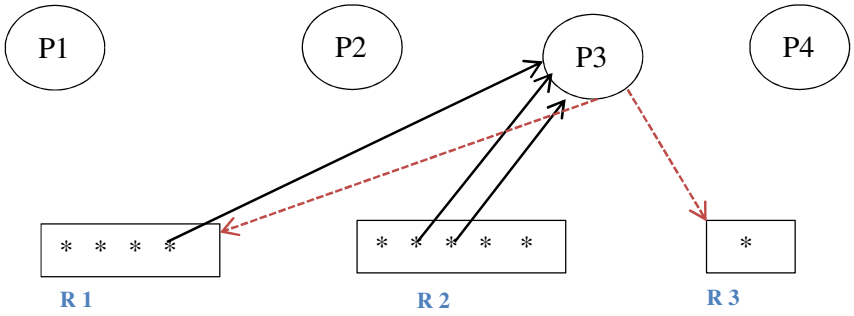
3) P1 n'as plus d'arcs de requêtes → effacer ses arcs d'allocations



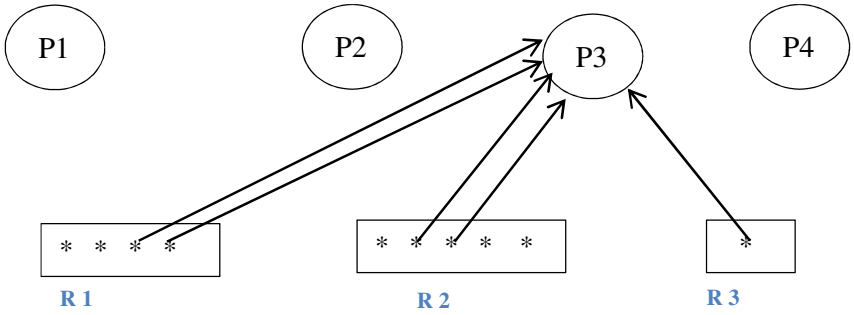
4) P4 a un arc de requête vers une instance de R2 qui est disponible → changer cet arc de requête en arc d'allocation



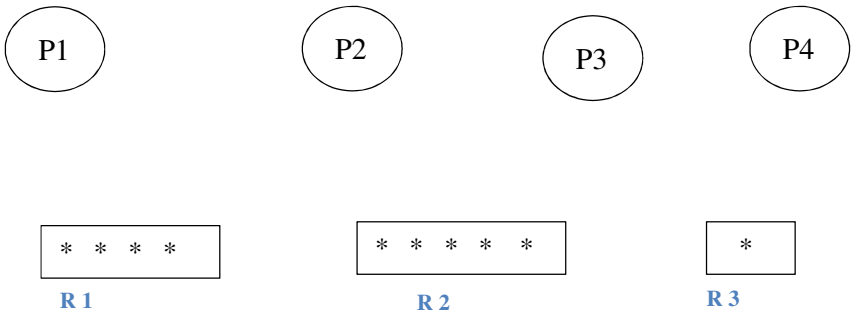
5) P4 n'as plus d'arcs de requêtes →effacer ses arcs d'allocation



6) P3 a un arc de requête vers une instance de R1 qui est disponible et un arc de requête vers une instance de R3 qui est disponible → changer ses arcs de requêtes en arcs d'allocations.



7) P3 n'as plus d'arcs de requête →effacer ses arcs d'allocation



Conclusion : il n'existe plus aucun arc→ le système n'est pas en interblocage.

Exercice4 :

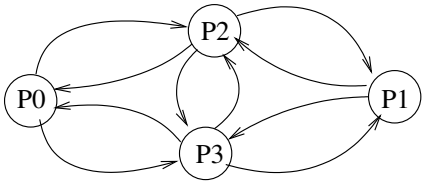
1) Table d'utilisation

Processus	Allocation R1 R2 R3	Requêtes R1 R2 R3	Disponible R1 R2 R3
P0	1 0 1	0 1 0	0 0 0
P1	1 0 1	0 1 0	
P2	1 1 0	0 0 1	
P3	0 1 1	1 0 0	

2) Tous les processus ont des arcs de requêtes ; Toutes les ressources sont requises ; il n'existe aucune ressource disponible

Conclusion : On ne peut pas réduire le graphe → le système est en interblocage.

3) Graphe d'attente. : il existe clairement un cycle dans ce graphe : P0 →P2 →P1 →P3 →P0



4) La correction d'un interblocage consiste à briser le cycle en choisissant de réquisitionner une ressource ou même tuer un processus bloqué dans le cycle (pour le choix du processus ou la ressource voir le cours page 6). Ici on peut choisir de réquisitionner R2 car elle affectera que 2 processus et elle a le plus de requêtes par rapport aux autres. Pour les processus ils sont tous pareils ! Mais on peut choisir de tuer le processus le plus récent (ici on n'a pas les dates !) car il y'aura moins de perte de données.

Mme YAICI