Interblocage

Exemple :

P1	P2
1. Demander(Périph1)	1. Demander(Périph2)
2. Demander(Périph2)	2. Demander(Périph1)
<Traitement>	<Traitement>
3. Liberer(Périph1)	3. Liberer(Périph2)
4. Liberer(Périph2)	4. Liberer(Périph1)

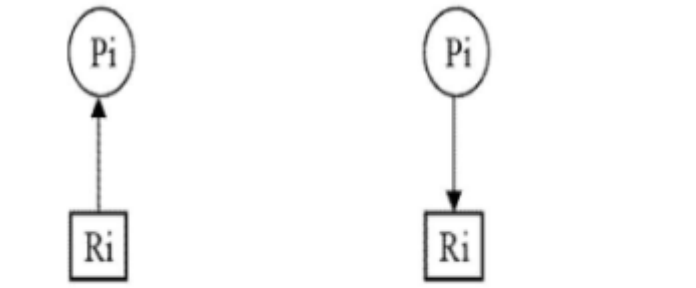
Exemple d'interblocage

Si P1 d'abord puis P2 ou P2 puis P1 alors pas d'interblocage. Mais s'ils s'exécutent en parallèle lors interblocage (chacun souhaite accéder à une ressource déjà utilisée par l'autre processus).

▼ Conditions nécessaires à l'interblocage : (conditions doivent être en simultané)

- Exclusion mutuelle : Une ressource au moins doit se trouver dans un mode non partageable et nécessite une EM pour son utilisation.
- Occupation et attente : Il peut exister un processus occupant au moins une ressource et qui attend d'acquérir des ressources détenues par d'autres processus (allocation partielle)
- Pas de réquisition : Les ressources déjà détenues ne peuvent être retirées de force à un processus. Elles doivent être explicitement libérées par le processus qui les détient.
- Attente circulaire

▼ Modélisation : ressources avec un carré et processus avec un cercle :

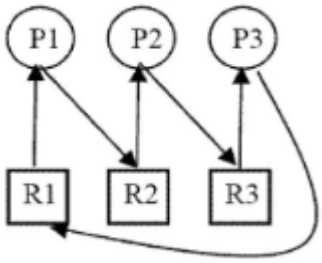


Pi détient la ressource Ri Pi demande la ressource Ri

Représentation de l'allocation de ressources

Théorème : Si chaque type de ressource possède exactement un seul exemplaire alors : Il y a
situation d'interblocage si et seulement si le graphe d'allocation possède un circuit.

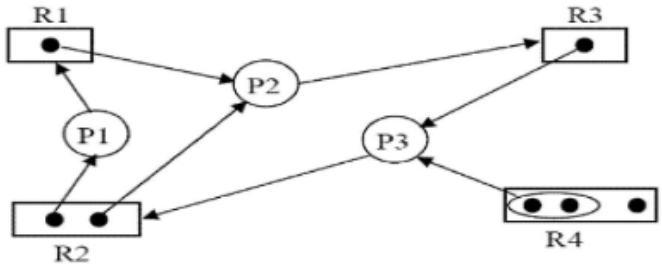
Exemple :



Théorème : Dans le cas de plusieurs exemplaires par type de ressource : Si le graphe d'allocation est sans circuit alors aucun processus n'est dans une situation d'inter-blocage.

Cycle1 : R2, P2, R3, P3, R2
==> **interblocage**

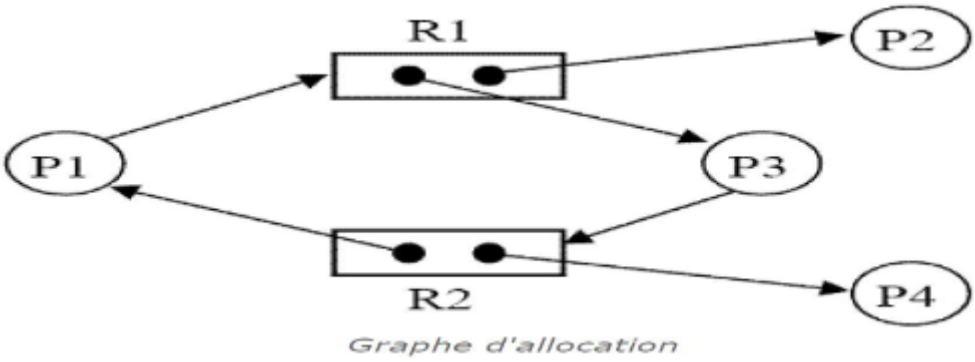
Cycle2 : R2, P1,R1,P2,R3,P3,R2
==> **interblocage.**



Représentation des exemplaires d'une ressource

▼ Allocation avec des matrices

Exemple avec les matrices :



Graphe d'allocation

DISPONIBLE[]

R1	R2
0	0

DEMANDE[], D={ P1, P3 }

	R1	R2
P1	1	0
P2	0	0
P3	0	1
P4	0	0

ALLOCATION[]

	R1	R2
P1	0	1
P2	1	0
P3	1	0
P4	0	1

Formule : Demande[i] ≤ DISPONIBLE + Σ Allocation[j]

⇔ DEMANDE[1] = (1, 0) ≤ (0,0) + (1,0) + (0,1)

Formule : Demande[i] ≤ DISPONIBLE + Σ Allocation[j] ⇔ DEMANDE[1] = (1, 0) ≤ (0,0) + (1,0) + (0,1)

▼ Eviter les interblocage

- Séquence Saine : Une séquence P1, P2, ..., Pn est saine si pour chaque processus Pi, les demandes de ressources de Pi peuvent être satisfaite par les ressources disponibles plus les ressources détenues par tous les Pj avec i>j.
- Algorithme du banquier : évaluer le risque d'interblocage pouvant être provoqué par une demande de ressource ⇒ Si une demande présente ce risque, le système doit la mettre en attente même si elle peut être satisfaite avec les ressources dispo
    - Algo de détermination d'état sain :

Travail : Tableau [1..*M*] d'entiers ;
Fini : Tableau [1.. *N*| de Booléen ;
1. *Travail* := DISPONIBLE ;
   Pour *i* = 1 jusqu'à *N* faire *Fini [i]* = faux ;
2. Trouver *i* tel que *Fini [i]* = faux et *BESOINi* <= *Travail*
   Si *i* n'existe pas aller à 4 ;
3. *Travail* = *Travail* + *Allocationi* ;
   Fini [i] = vrai ;
   Aller à 2 ;
4. Si *Fini [i]* = vrai pour tout *i* alors le système est dans un état sain ;

- Algo de requête :

1. Si *DEMANDEi* ≤ *BESOINi* Alors Aller à 2
   Sinon *Erreur* : excéder sa demande maximale
2. Si *DEMANDEi* ≤ *DISPONIBLE* Alors Aller à 3
   Sinon *Attente*
3. /* On suppose que le système a alloué les ressources demandées par le processus Pi */
   DISPONIBLE = *DISPONIBLE* - *DEMANDEi*
   ALLOCATIONi = *ALLOCATIONi* + *DEMANDEi*
   BESOINi = *BESOINi* - *DEMANDEi*
4. Si cet état est sain Alors allocation avec succès
   Sinon /* Restaurer l'état initial */
   DISPONIBLE = *DISPONIBLE* + *DEMANDEi*
   ALLOCATIONi = *ALLOCATIONi* - *DEMANDEi*
   BESOINi = *BESOINi* + *DEMANDEi*

▼ TD10 EX1

Soit un système qui ne contient qu'une seule ressource de chaque type. L'état du système est comme suivi :

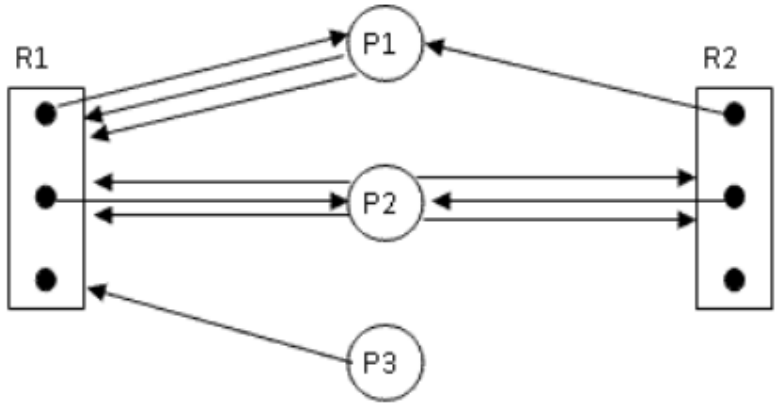
- Le processus A détient la ressource R et veut la ressource S.
- Le processus B ne détient aucune ressource mais veut la ressource T.
- Le processus C ne détient aucune ressource mais veut la ressource S.
- Le processus D détient la ressource U et veut les ressources S et T
- Le processus E détient la ressource T et veut la ressource V.
- Le processus F détient la ressource W et veut la ressource S.
- Le processus G détient la ressource V et veut la ressource U.

- Ce système est-il en situation d'interblocage ?

⇒ On construit le graphe ⇒ On trouve un circuit et les ressources ne sont disponibles qu'en un seul exemplaire ⇒ Interblocage

▼ TD10 EX2

L'état d'un système est représenté par le graphe suivant :



Demande de ressources

- Donner la représentation par matrice
- Y a t-il interblocage ?

DISPONIBLE[] :

R1	R2
1	1

ALLOCATION[] :

	R1	R2
P1	1	1

P2	1	1
P3	0	0

DEMANDE [] :

	R1	R2
P1	2	0
P2	2	2
P3	1	0

D = { P1, P2 } ⇒ Car ils forment le circuit

Note : (x, y) < (w, z) ssi x<w ET y<z. Si on ne peux pas vérifier pour aucun processus (exemple : (2,0) < (1,1)), alors il y a interblocage

DEMANDE[i] ≤ DISPONIBLE[] + ALLOCATION[P3] car P3 ∉ D

DEMANDE[1] = (2,0) ≤ (1, 1) + (0, 0) ⇒ FAUSSE

DEMANDE[2] = (2,2) ≤ (1,1) + (0,0) ⇒ FAUSSE

⇒ Interblocage

▼ TD10 EX4

L'état des allocations de ressources d'un système est donné par la représentation suivante :

	ALLOCATION					DEMANDE					DISPONIBLE			
	R0	R1	R2	R3		R0	R1	R2	R3	R0	R1	R2	R3
P0	1	0	0	0	P0	0	0	1	0	0	0	0	0
P1	0	0	0	0	P1	1	1	1	0				
P2	0	0	1	1	P2	0	1	0	0				
P3	0	0	0	0	P3	0	1	0	1				
P4	0	1	0	0	P4	0	0	0	1				

Matrice d'allocation et de demande

- Représenter le graphe d'allocation.
- Représenter le graphe des attentes

Voir notes papier

⇒ Le graphe d’attente montre qu’il y a un interblocage