# Inter-blocages
# (Deadlocks)

Abdelouahed Gherbi

Hiver 2020

# Plan

- Introduction
  - Qu'est ce qu'un inter-blocage
  - Conditions nécessaires pour l'inter-blocage
- Modélisation d'inter-blocage
- Prévention d'inter-blocage
- Détection d'inter-blocage
- Évitement d'inter-blocage

# Introduction aux inter-blocages

- Un système est composé d'un nombre fini de ressources
- Plusieurs processus sont en compétition pour les ressources du système
- Un processus doit demander (requête) une ressource avant de l'utiliser et doit la libérer après.

# Introduction aux inter-blocages

- Un processus utilise une ressource comme suit :
  - Demander la ressource (requête) :
    - si la ressource ne peut pas être allouée immédiatement le processus doit attendre
    - Se fait via un appel système : request(), open(), allocate()
    - Pour les ressources gérés par l'utilisateur : wait(s) ou s est un sémaphore
  - Utiliser la ressource
  - Libérer la ressource
    - Se fait via des appels système: release(), close(), free()
    - Se fait via signal(s) sur un sémaphore s pour les ressources gérés par l'utilisateur

# Introduction aux inter-blocages

- Qu’est ce qu’un inter-blocage ?

# Introduction aux inter-blocages

- Qu'est ce qu'un inter-blocage ?

  Une situation où un ensemble de processus sont en inter-blocage si chaque processus dans l'ensemble est en attente d'un événement qui ne peut être causé que par un autre processus dans l'ensemble

# Introduction aux inter-blocages

- Exemple 1 :

  « When two trains approach each other at a crossing, both shall come to a full stop and neither shall start up again until the other has gone »,

  -- A law passed by the Kansas ligislature in the early 20th centery [1]

# Introduction aux inter-blocages

- Exemple 2 :
  - semaphores A et B initialisés à 1

| $P_0$ | $P_1$ |
| --- | --- |
| …. | ……. |
| **wait (A);** | **wait(B);** |
| **…..** | **…..** |
| **wait (B);** | **wait(A);** |
| **……** | **….** |

# Introduction aux inter-blocages

- Il a été démontré [2] que les conditions suivantes doivent être vérifiées pour causer un inter-blocage

- **Exclusion Mutuelle:**
  - Chaque ressource doit être attribuée à un seul processus à la fois

- **Détention et attente:**
  - Les processus ayant déjà obtenu des ressources peuvent en demander des nouvelles

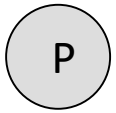
- **Pas de réquisition (No preemption):**
  - Les ressources déjà détenues par un processus ne peuvent lui être retirées de force.
  - Elles doivent être explicitement libérées par le processus.

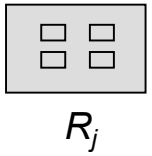
- **Attente circulaire:**
  - Il doit y avoir un ensemble de processus {P0, P1, …,Pn} en attente tels que : P0 est en attente pour une ressource détenue par P1, P1 est en attente d'une ressource détenue par P2, …, Pn–1 est en attente d'une ressource détenue par Pn, and Pn est en attente d'une ressource détenue par P0.

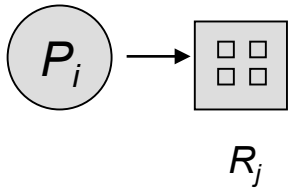
# Modélisation des inter-blocages

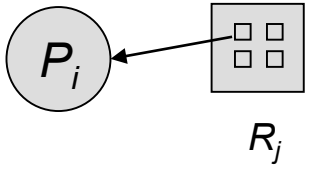
- Les inter-blocages peuvent être représentés (modélisés) en utilisant **un graphe orienté**
  - graphe d'allocation de ressources
- Ce graphe consiste en un ensemble N de nœuds et un ensemble A d'arcs.
- N est partitionné en deux types de nœuds :
  - $P = \{P_1, P_2, \ldots, P_n\}$ : ensemble de tous les processus du système
  - $R = \{R_1, R_2, \ldots, R_m\}$ : ensemble des types de ressources dans le système
- Arc de requête
  - Un arc orienté d'un processus P1 vers un type de ressource P1 $\rightarrow$ Rj
  - Signifie que le processus P1 a demandé une instance du type de ressource Rj et est actuellement en attente pour cette ressource
- Arc d'allocation
  - Un arc orienté d'un type de ressource Rj vers un processus Pi : Rj $\rightarrow$ Pi
  - Signifie qu'une instance de du type de ressource Rj est allouée au processus Pi

# Modélisation des interblocages

- Graphiquement :
- Processus 

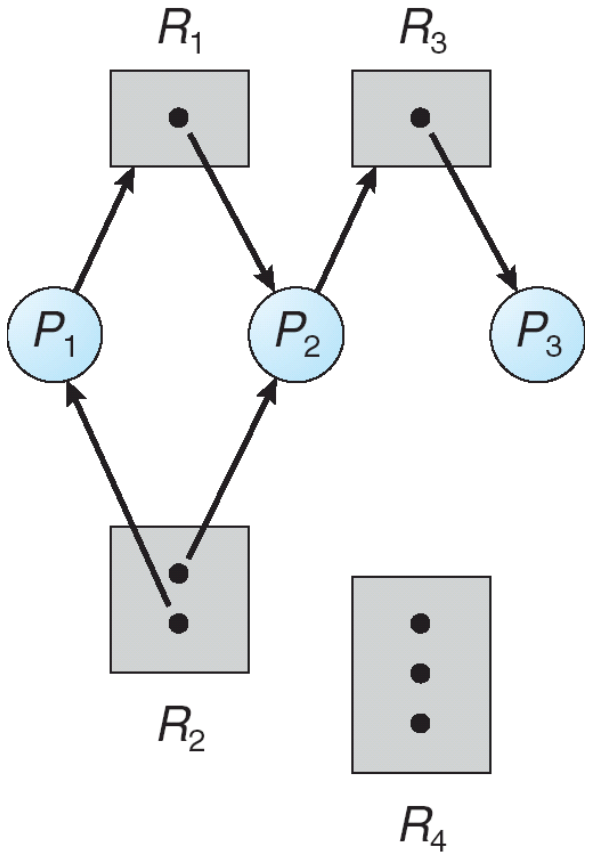
- Un type de ressource $R_j$ avec 4 instances 

- *Le processus $P_i$ demande une instance de $R_j$* 

- *Le processus $P_i$* détiens une instance of $R_j$ 

# Modélisation des inter-blocages

- Exemple
- Les ensembles P, R et A
  - $P = \{P_1, P_2, P_3\}$,
  - $R = \{R_1, R_2, R_3, R_4\}$,
  - $A = \{P_1 \text{-->} R_1, P_2 \text{--> } R_3, R_1 \text{--> } P_2, R_2 \text{--> } P_2\ R_2 \text{--> } P_1, R_3 \text{--> } P_3\}$
- Instances de ressources
  - *Une instance du type de ressources* $R_1$
  - *Deux instances du type de ressources* $R_2$
  - *Une instance du type de ressources* $R_3$
  - *Trois instances du type de ressources* $R_4$
- État des processus
  - *Processus* $P_1$ *détiens une instance du type de ressource* $R_2$ *et est en attente d'une instance du type de ressource* $R_1$
  - *Processus* $P_2$ *détiens une instance du type de ressource* $R_1$ *et une instance du type de ressource* $R_2$ *et est en attente d'une instance du type de ressource* $R_3$
  - *Processus P3 détiens une instance du type de ressource R3*



# Modélisation des inter-blocages

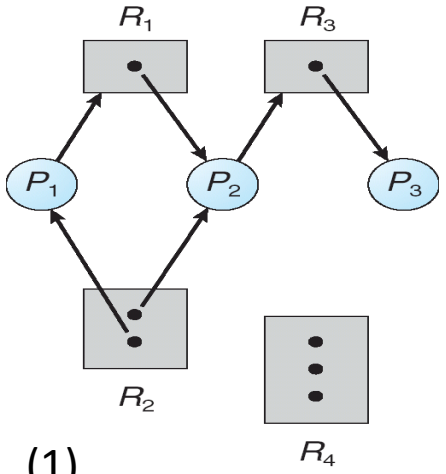
- Il peut être démontré que :

- Si la graphe d'allocation ne contient pas de cycles alors il n'y a pas d'inter-blocage

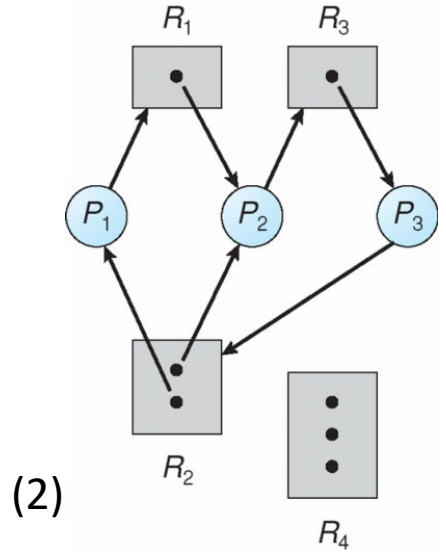
- Si le graphe contient un cycle alors
  - Si il y a une seule instance par type de ressources, alors inter-blocage
  - Si il y a plusieurs instances par type de ressources, alors il y a une possibilité d'inter-blocage

# Modélisation des inter-blocages

- Exemple
- On considère le graphe d'allocation de ressources (1)
- Si le processus $P_3$ demande une instance de type de ressource $R_2$
- On obtient le graphe d'allocation de ressource (2)
- Est ce qu'il existe des cycles dans ce graphe? Si oui quels sont les processus impliqués ?



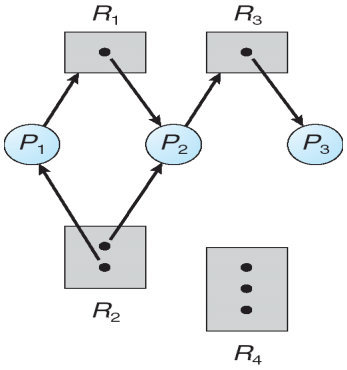

(1)



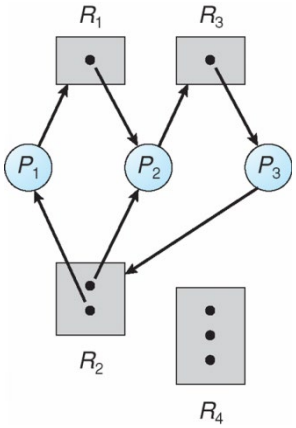

(2)

# Modélisation des inter-blocages

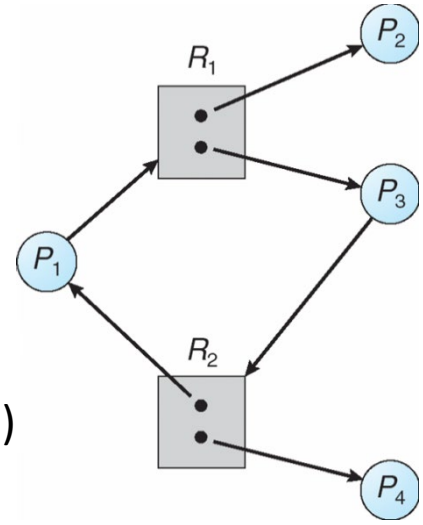
- Exemple
- On considère le graphe d'allocation de ressources précédent
- Si le processus $P_3$ demande une instance de type de ressource $R_2$




- On obtient le graphe de ressource b




- Oui il y a deux cycles :
  - P1-->R1-->P2-->R3-->P3-->R2-->P1
  - P2-->R3-->P3-->R2-->P2
  - Alors les processus P1, P2 et P3 sont en inter-blocage
- Qu'en est t-il du graphe (3) ?




(3)

# Méthodes de traitement des inter-blocages

- Ignorer le problème et prétendre que les inter-blocages n'arrivent jamais dans un système (Politique de l'autruche)
  - Utilisée par la plus part des systèmes d'exploitation incluant UNIX
- Prévention des inter-blocages
- Détection des inter-blocages
- Évitement des inter-blocages

# Prévention des inter-blocages

- Cette méthode est basée sur l'idée de s'assurer que l'une des conditions d'inter-blocage ne soit pas vérifiée

- **Exclusion mutuelle**
  - pas nécessaire pour les ressources partageables
  - doit être vérifiée pour les autres

- **Détention et attente**
  - Il suffit de garantir qu'à chaque fois qu'un processus demande une ressource, il ne doit pas déjà détenir aucune autre
  - Le processus doit demander et être alloué toutes les ressources avant qu'il commence à exécuter
  - Ou permettre à un processus de demander une ressource seulement quand il n'en a pas
  - Faible utilisation des ressources et possibilité de famine

- **Pas de réquisition (No Preemption)**
  - Si un processus qui détient des ressources demande une autre ressource qui ne peut pas lui être immédiatement attribuée (il doit donc attendre) , alors toutes les ressources actuellement détenues par ce processus sont libérés
  - Les ressources réquisitionnées sont ajoutées à la liste des ressources pour lesquelles le processus est en attente
  - Le processus sera remis en marche quand il peut reprendre ses ressources anciennes, ainsi que les nouvelles qu'il demande
  - Difficile à implémenter pour des ressources qui ne peuvent pas être non réquisitionnées (imprimantes, bandes magnétiques, etc.)

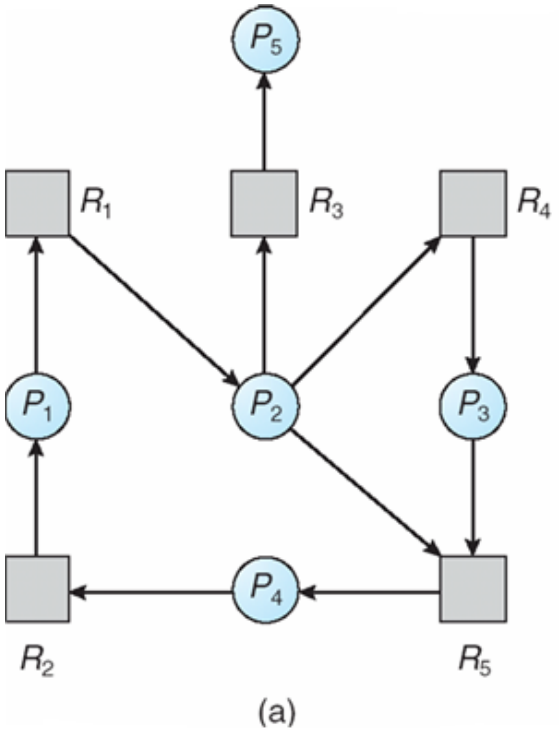
- **Attente circulaire**
  - Imposer un ordre total sur tous les types de ressources et exiger que chaque processus demande les ressources dans un ordre croissant

# Détection des inter-blocages

- Permettre au système d'entrer dans un état d'inter-blocage
- Utiliser un algorithme de détection d'inter-blocages
- Si un inter-blocage est détecté, appliquer une procédure de récupération (recovery) du système
- Pour la détection d'inter-blocages, on distingue deux cas
  - Instance unique pour chaque type de ressource
  - Instances multiples pour chaque type de ressource

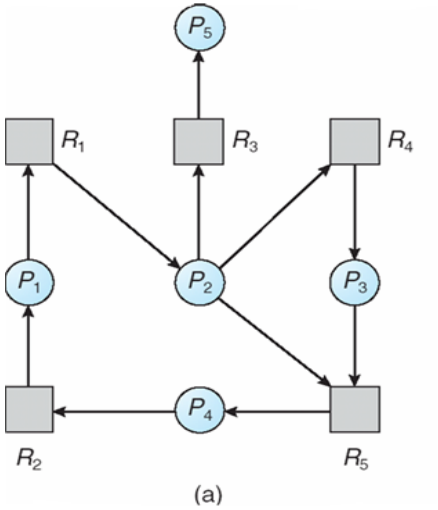
# Détection des inter-blocages (Instance unique par type de ressource)

- On peut définir un algorithme qui utilise le **graphe d'attente** (wait-for graph)
- Le graph d'attente est une variante du graphe d'allocation de ressources obtenu en éliminant les nœuds de ressources et en réunissant les arcs adéquats
- En particulier :
  - Un arc de Pi --> Pj dans un graphe d'attente implique que Pi attend que Pj libère la ressource dont il a besoin
  - Un arc Pi-->Pj existe dans un graphe d'attente ssi le graphe d'allocation de ressources correspondant contient deux arcs Pi-->Rq et Rq-->Pj pour une ressource Rq donnée
- Exemple : Quel est le graphe d'attente correspondant au graphe d'allocation suivant?

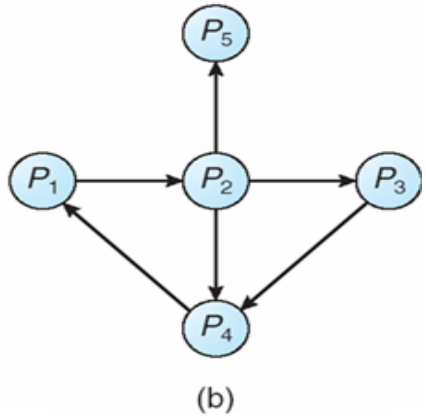


# Détection des inter-blocages (Instance unique par type de ressource)

Graphe d'allocation de ressources




Graphe d'attente correspondant

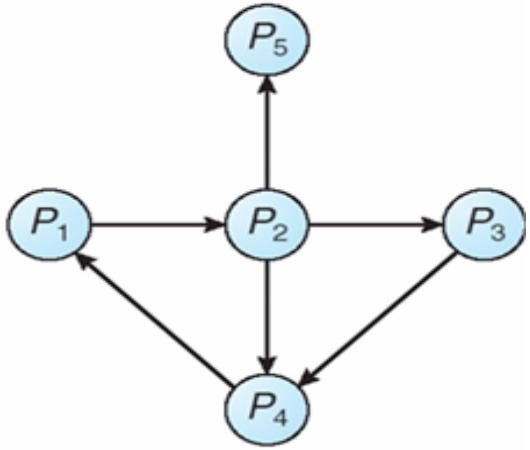


# Détection des inter-blocages (Instance Unique par type de ressource)

- Un inter-blocage existe dans le système si et seulement si le graphe d'attente contient un cycle

- Complexité de la solution: afin de détecter les inter-blocages, le système d'exploitation doit :
  - maintenir le graphe d'attente
  - Invoquer périodiquement un algorithme de recherche de cycle dans le graphe

- Exemple :
  - D'après notre graphe d'attente, est ce que le système présente des inter-blocages ?
  - Si oui, quels sont les processus impliqués dans ces inter-blocages ?



(b)

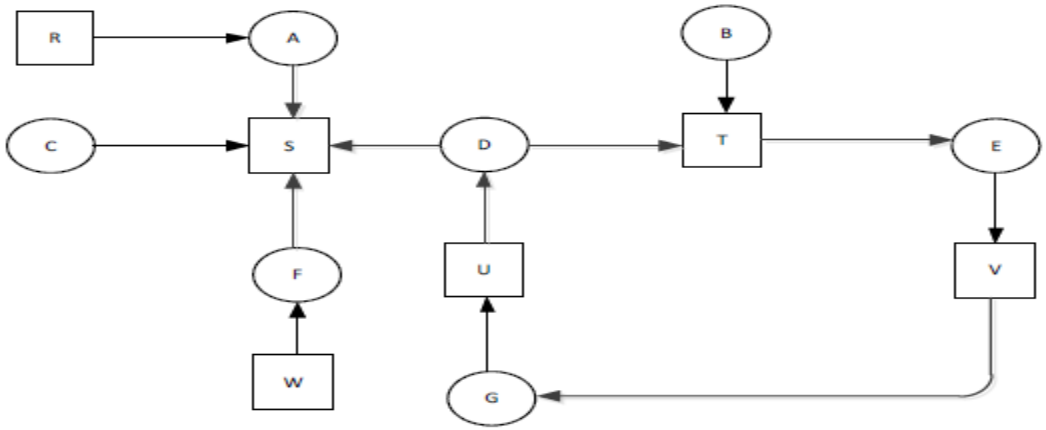
# Détection des inter-blocages (Instance Unique par type de ressource) Exercice #1

1. Considérons l'attribution des ressources suivante :
   - A détient R et demande S ;
   - B demandes T ;
   - C demandes S ;
   - D détient U et demande S et T ;
   - E détient T et demande V ;
   - F détient W et demande S ;
   - G détient V et demande U.

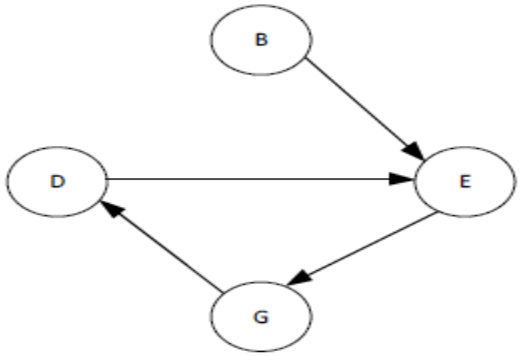
Construire le graphe d'allocation des ressources. Est-ce qu'il y a un inter-blocage ? Si oui, quels sont les processus concernés ?

# Détection des inter-blocages (Instance Unique par type de ressource) Exercice #1

Graphe d'allocation de ressources




Comme il y une instance unique par type de ressource, on utilise le graphe d'attente :




Dans ce graphe d'attente il y a un cycle D→E→G→D donc les processus {D,E,G} sont en inter-blocage

# Détection des inter-blocages (Plusieurs Instances par type de ressource)

- L'algorithme basé sur le graphe d'attente n'est pas applicable pour un système d'allocation de ressources avec plusieurs instances par type de ressources

- **Disponible**
  - Un tableau (vecteur) de longueur *m* qui indique le nombre de ressources (instance) disponibles pour chaque type

- **Allocation**
  - Une matrice *n* x *m* qui définit le nombre de ressources de chaque type (instance) actuellement allouées à chaque processus.

- **Requête***:*
  - Une matrice *n* x *m* qui indique les requêtes (demandes) actuelles de chaque processus.
  - Sir *Requête* [*i*][j] = *k*, alors le processus $P_i$ demande *k* instances de plus de du type de ressource $R_j$.

# Détection des inter-blocages (Plusieurs Instance par type de ressource)

1. Soit *Travail* et *Fin* deux tableaux de longueurs *m* et *n*, respectivement
   On initialise *Travail* et *Fin* comme suit :
   (a) *Travail* = *Disponible*
   (b) for *i* = 1,2, …, *n*,
   if *Allocation*[i] $\neq$ 0, then *Fin*[i] = false;
   else *Fin*[i] = *true*
2. Trouver un index *i* tel que :
   (a) *Fin*[*i*] == *false*
   *(b)* *Requête* [i] $\leq$ *Travail*
   Si un tel i n'existe pas passer à l'étape 4

3. *Travail* = *Travail* + *Allocation[i]*
   *Fin*[*i*] = *true*
   Aller à étape 2

4. If *Fin*[*i*] == false, pour certain *i*, $1 \leq i \leq n$,
   then le system est dans un état d'inter-blocage.
   Pour chaque processus Pi tel que *Fin*[*i*] == *false*, $P_i$ est en inter-blocage

# Détection des inter-blocages (Plusieurs Instance par type de ressource)

- Exemple 1

- On considère un système composé de :
  - Cinq (5) processus $P_0$ à $P_4$;
  - Trois (3) types de ressources : A (7 instances), *B* (2 instances), et *C* (6 instances)

- A l'instant $T_0$ on l'état d'allocation de ressources est comme suit :

| | *Allocation* | *Requête* | *Disponible* |
|---|---|---|---|
| | *A B C* | *A B C* | *A B C* |
| $P_0$ | 0 1 0 | 0 0 0 | 0 0 0 |
| $P_1$ | 2 0 0 | 2 0 2 | |
| $P_2$ | 3 0 3 | 0 0 0 | |
| $P_3$ | 2 1 1 | 1 0 0 | |
| $P_4$ | 0 0 2 | 0 0 2 | |

- Est ce que ce système est en état d'inter-blocage ?

# Détection des inter-blocages (Plusieurs Instance par type de ressource)

- Exemple 2

- On considère un système composé de :
  - Cinq (5) processus $P_0$ à $P_4$;
  - Trois (3) types de ressources : A (7 instances), *B* (2 instances), et *C* (6 instances)

- Le processus P2 fait une requête supplémentaire pour instance de type C:

| | Allocation | Requête | Disponible |
|---|---|---|---|
| | *A B C* | *A B C* | *A B C* |
| $P_0$ | 0 1 0 | 0 0 0 | 0 0 0 |
| $P_1$ | 2 0 0 | 2 0 2 | |
| $P_2$ | 3 0 3 | 0 0 **1** | |
| $P_3$ | 2 1 1 | 1 0 0 | |
| $P_4$ | 0 0 2 | 0 0 2 | |

- Est-ce que le système est en état d'inter-blocage ? Si oui quels sont les processus impliqués?

# Détection des inter-blocages (Plusieurs Instance par type de ressource) Exercice #2

2. On considère un système composé de 4 types de ressources :
   - Quatre (4) dérouleurs de bandes (DB)
   - Deux (2) tables traçantes (TR)
   - Trois (3) scanners (S)
   - Un (1) lecteur de CD (CD)

Requête

| DB | TR | S | CD |
|---|---|---|---|
| 2 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 2 | 1 | 0 | 0 |

Le système comporte trois (3) processus. Le processus P1 possède un scanner ; le processus P2 possède deux dérouleurs de bandes et un lecteur CD ; le processus P3 possède une table traçante et deux scanners. Chaque processus a besoin de ressources supplémentaires comme le montre la matrice Requête.

a. Donner la matrice d'allocation
b. Donner le tableau de ressources disponibles
c. Exécuter l'algorithme de détection d'inter-blocage pour vérifier si le système est en inter-blocage et si oui quels sont les processus impliqués
d. Supposons que le processus 2 ait besoin de trois scanners et de la table traçante. Est-ce que le système est en état d'inter-blocage ?

# Détection des inter-blocages
# (Plusieurs Instance par type de ressource)
# Exercice #2

a.
Allocation

| | DB | TR | S | CD |
|---|---|---|---|---|
| P1 | 0 | 0 | 1 | 0 |
| P2 | 2 | 0 | 0 | 1 |
| P3 | 0 | 1 | 2 | 0 |

Requête

| DB | TR | S | CD |
|---|---|---|---|
| 2 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 2 | 1 | 0 | 0 |

b.
Disponible

| DB | TR | D | CD |
|---|---|---|---|
| 2 | 1 | 0 | 0 |

c.
Exécution de l'algorithme de détection

Travail

| | DB | TR | S | CD |
|---|---|---|---|---|
| Initialement | 2 | 1 | 0 | 0 |
| Itération 1 | 2 | 2 | 2 | 0 |
| Itération 2 | 4 | 2 | 2 | 1 |
| Itération 3 | | | | |

Tableau Fin

| | P1 | P2 | P3 |
|---|---|---|---|
| Initialement | false | false | false |
| Itération 1 | false | false | true |
| Itération 2 | false | true | true |
| Itération 3 | true | True | true |

Le système n'est pas en inter-blocage

# Détection des inter-blocages
# (Plusieurs Instance par type de ressource)
# Exercice #2

d. P2 ait besoin de trois scanners et de la table traçante donc la matrice requête est comme suit :

Requête

| DB | TR | S | CD |
|---|---|---|---|
| 2 | 0 | 0 | 1 |
| 0 | 1 | 3 | 0 |
| 2 | 1 | 0 | 0 |

Allocation

| | DB | TR | S | CD |
|---|---|---|---|---|
| P1 | 0 | 0 | 1 | 0 |
| P2 | 2 | 0 | 0 | 1 |
| P3 | 0 | 1 | 2 | 0 |

Travail

| | DB | TR | S | CD |
|---|---|---|---|---|
| Initialement | 2 | 1 | 0 | 0 |
| Itération 1 | 2 | 2 | 2 | 0 |

Tableau Fin

| | P1 | P2 | P3 |
|---|---|---|---|
| Initialement | false | false | false |
| Itération 1 | false | false | true |

Donc le système est en inter-blocage et les processus impliqués sont {P1, P2}

# Détection des inter-blocages
# (Plusieurs Instance par type de ressource)
# Exercice #3

3. Considérons un système gérant quatre processus, P1 à P4, et trois types de ressources R1, R2 et R3 (3 R1, 2 R2 et 2 R3). Les ressources sont attribuées comme suit :
    - P1 détient une ressource de type R1 et demande une ressource de type R2 ;
    - P2 détient 2 ressources de type R2 et demande une ressource de type R1 et une ressource de type R3 ;
    - P3 détient 1 ressource de type R1 et demande une ressource de type R2 ;
    - P4 détient 2 ressources de type R3 et demande une ressource de type R1 ;

Est-ce qu'il y a un inter-blocage ? Si oui, quels sont les processus concernés ?

# Détection des inter-blocages
# (Plusieurs Instance par type de ressource)
# Exercice #3

Matrice d'allocation

| | R1 | R2 | R3 |
|---|---|---|---|
| P1 | 1 | 0 | 0 |
| P2 | 0 | 2 | 0 |
| P3 | 1 | 0 | 0 |
| P4 | 0 | 0 | 2 |

Requête

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

Tableau Disponible

| R1 | R2 | R3 |
|---|---|---|
| 1 | 0 | 0 |

Exécution de l'algorithme de détection d'inter-blocage

Travail

| | R1 | R2 | R3 |
|---|---|---|---|
| Initialement | 1 | 0 | 0 |
| Itération 1 | 1 | 0 | 2 |
| Itération 2 | 1 | 2 | 2 |
| Itération 3 | 2 | 2 | 2 |
| Itération 4 | 3 | 2 | 2 |

Tableau Fin

| | P1 | P2 | P3 | P4 |
|---|---|---|---|---|
| Initialement | false | false | false | false |
| Itération 1 | false | false | false | true |
| Itération 2 | false | true | false | true |
| Itération 3 | True | True | False | true |
| Itération 4 | True | True | True | true |

Le système n'est pas en interblocage

# Évitement des inter-blocages

- Une méthode alternative basée sur une connaissance a priori d'informations sur comment les ressources vont être demandées

- Le système, sachant pour chaque processus, la séquence complète des requêtes et libérations
  - peut décider **lors de chaque requête** si le processus doit ou non attendre afin d'éviter un possible inter-blocage future

- Pour faire cette décision, le système considère pour chaque requête :
  - Les ressources actuellement disponibles
  - Les ressources actuellement allouées pour chaque processus
  - Les requêtes et libérations futures pour chaque processus

- Un algorithme d'évitement d'inter-blocage examine **dynamiquement** l'état d'allocation des ressources afin d'assurer que la condition d'attente circulaire n'arrive jamais

- L'état d'allocation des ressources est défini par le nombre de ressources disponibles, allouées et le nombre maximum des demandes des processus

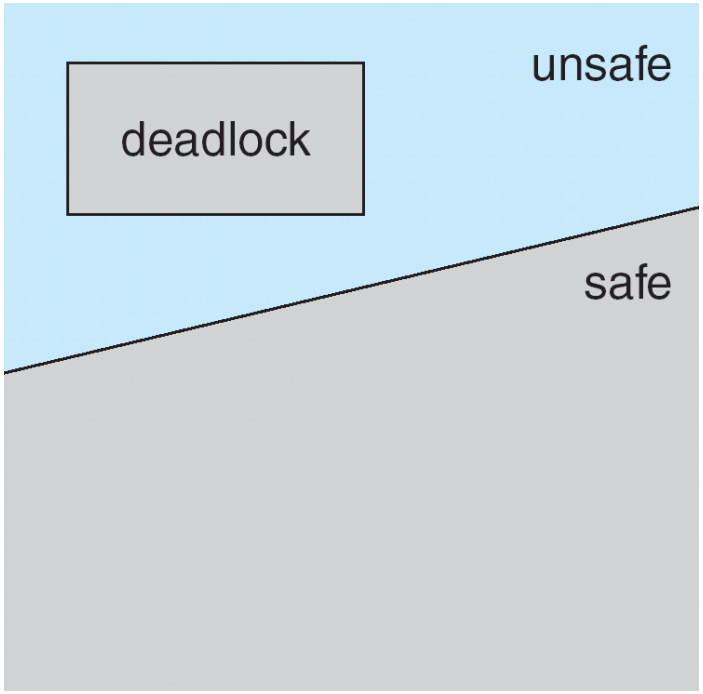
# Évitement des inter-blocages

- Notion **d'état sûr** (safe state)
  - Un état est **sûr** si le système peut allouer les ressources pour chaque processus dans un certain ordre tout en continuant à éviter l'inter-blocage

- Formellement :
  - Un système est dans un état sûr s'il existe **une séquence sûre**
  - Une séquence de processus (P1, P2, …, Pn) est **une séquence sûre** pour l'état d'allocation courant Si :
    - Pour chaque Pi, les requêtes que Pi peut encore faire peuvent être satisfaites avec les ressources actuellement disponibles plus celles détenues pas les Pj tel que j < I
  - Dans cette situation, si les ressources nécessaires pour Pi ne sont pas immédiatement disponibles, alors Pi peut attendre jusqu'à ce que tous les Pj terminent
  - Quand les Pj terminent, Pi peut obtenir toutes ses ressources , fait sa tache et termine en retournant ses ressources
  - Quand Pi termine, Pi+1 peut obtenir ses ressources et ainsi de suite.
  - Si une telle séquence n'existe pas, l'état du système est dit non sûr (unsafe)

# Évitement des inter-blocages

- Un état **sûr** n'est pas un état d'inter-blocage
- Inversement, un état d'inter-blocage est un état **non sûr**
- Les états **non sûr** ne sont pas tous des états d'inter-blocage mais peuvent conduire à un inter-blocage
- Tant et aussi longtemps que le système est dans un état sur, il peut éviter l'inter-blocages



# Évitement des inter-blocages

- Exemple

- On considère un système composé de
  - Douze (12) dérouleurs de bandes magnétiques (tapes drives) DBM
  - Trois processus P0, P1 et P2 tels que
    - Processus P0 a besoin de 10 DBM; processus P1 peut avoir besoin de 4 DBM; et processus P2 peut avoir besoin de 9 DBM

| | Besoins maximum | Besoins actuels |
|---|---|---|
| P0 | 10 | 5 |
| P1 | 4 | 2 |
| P2 | 9 | 7 |

- A l'instant T0 :
  - P0 détiens 5 DBM; P1 détiens 2 DBM et P3 détiens 2 DBM

- Est ce que le système est dans un état sûr à l'instant T0

# Évitement des inter-blocages

- Exemple

- On considère un système composé de
  - Douze (12) dérouleurs de bandes magnétiques (tapes drives) DBM
  - Trois processus P0, P1 et P2 tels que
    - Processus P0 a besoin de 10 DBM; processus P1 peut avoir besoin de 4 DBM; et processus P2 peut avoir besoin de 9 DBM

| | Besoins maximum | Besoins actuels |
|---|---|---|
| P0 | 10 | 5 |
| P1 | 4 | 2 |
| P2 | 9 | 7 |

- A l'instant T0 :
  - P0 détiens 5 DBM; P1 détiens 2 DBM et P3 détiens 2 DBM

- A l'instant T0 : il y a donc 3 DBM disponibles

- La séquence P1, P0, P2 est une séquence sûre

# Évitement des inter-blocages

- Algorithme du banquier (banker's algorithm) – Dijkstra 1965
  - Algorithme d'évitement d'inter-blocage dans un système avec plusieurs instances par type de ressources
  - Ce nom est choisi car l'algorithme peut être utilisé dans un système bancaire pour assurer que la banque n'utilisera jamais les liquidités disponibles de sorte qu'elle ne sera jamais en mesure de satisfaire les besoins de ses clients

- Quand un processus entre dans le système, il doit déclarer le nombre maximum d'instances pour chaque type de ressource dans le système dont il aura besoin.

- Quand un processus fait une requête pour un ensemble de ressources

- Le système doit déterminer **si l'allocation de ces ressources va laisser** le système dans **un état sûr**
  - Si oui, les ressources sont allouées
  - Si non, le processus doit attendre jusqu'à ce que d'autres processus libèrent assez de ressources

# Évitement des inter-blocages

- Structures de données pour l'implémentation de l'algorithme du banquier

- n : nombre de processus, m : nombre de types de ressources

- **Disponible** *:*
  - Tableau de longueur *m*
  - Si Disponible [*j*] = *k*, il y a k instances disponibles du type de ressource $R_j$
- **Max***:*
  - Matrice n x m
  - Si Max [i,j] = k, le processus Pi peut demander au plus k instances du type de ressources Rj
- **Allocation***:*
  - Matrice n x m matrix.
  - Si Allocation[i,j] = k, le processus Pi est actuellement alloué k instances de Rj
- **Besoin***:*
  - Matrice n x m
  - Si Besoin[i,j] = k, le processus Pi pourra avoir besoin de k instances de Rj supplémentaires pour compléter sa tache

- Quelle est la relation entre Besoin[i,j], Max[i,j] et Allocation[i,j] ?

# Évitement des inter-blocages

- Structures de données pour l'implémentation de l'algorithme du banquier

- n : nombre de processus, m : nombre de types de ressources

- **Disponible** *:*
  - Tableau de longueur *m*.
  - Si Disponbile [*j*] = *k*, il y a k instances disponibles du type de ressource $R_j$
- **Max***:*
  - Matrice n x m .
  - Si Max [i,j] = k, le processus Pi peut demander au plus k instances du type de ressources Rj
- **Allocation***:*
  - Matrice n x m matrix.
  - Si Allocation[i,j] = k, le processus Pi est actuellement alloué k instances de Rj
- **Besoin***:*
  - Matrice n x m
  - Si Besoin[i,j] = k, le processus Pi pourra avoir besoin de k instances de Rj supplémentaires pour compléter sa tache

- Quelle est la relation entre Besoin[I,j], Max[I,j] et Allocation[I,j] ?

  - *Besoin*[*i,j]* = *Max*[*i,j*] – *Allocation* [*i,j*]

# Évitement des inter-blocages

- Algorithme de sûreté
  - Algorithme qui détermine si un système est dans un état sûr ou non

1. Soit *Travail* et *Fin* des tableaux (vecteurs) de longueurs *m* et *n*, respectivement.
   On initialise :
   *Travail = Disponible*
   *Fin* [*i*] = *false* pour *i* = 0, 1, …, *n*- 1

2. Trouve un index *i* tel que :
   (a) *Fin* [*i*] = *false*
   (b) *Besoin[i]* $\leq$ *Travail*
   Si un tel i n'existe pas aller à étape 4

3. *Travail= Travail + Allocation[i]*
   *Fin*[*i*] = *true*
   Aller à étape 2

4. Si *Fin* [*i*] == true pour tout *i*, alors le système est dans un état sûr

# Évitement des inter-blocages

- Algorithme de demande de ressources (Resource-Request Algorithm)

- $Requête_i$ est un tableau (vecteur)
  - Représente la requête du processus $P_i$
  - Si $Requête_i$ [j] = k, le processus $P_i$ demande k instances du type de ressource Rj
- Quand un processus fait une requête les actions suivantes sont mise en œuvre

1. Si $Requête_i \leq$ Besoin[i] alors aller à étape 2.
   sinon provoque une condition d'erreur car le processus a dépassé son maximum de réclamation
2. Si $Requête_i \leq Disponible$, alors aller à étape 3.
   sinon $P_i$ doit attendre car les ressources ne sont pas disponibles
3. **Prétendre** (simuler) d'allouer les ressources demandées au processus $P_i$ en modifiant l'état comme suit :

   $Disponible = Disponible - Requête_i;$
   $Allocation[i] = Allocation[i] + Requête_i;$
   $Besoin[i] = Besoin[i] - Requête_i;$

   - *Si état est sûr $\Rightarrow$ les ressources sont allouées à Pi*
   - *Si l'état n'est pas sûr $\Rightarrow$ Pi doit attendre, et restaurer l'ancien état d'allocation*

# Évitement des inter-blocages

- Exemple
- Un Système composé de :
  - 5 processus $P_0$ … $P_4$
  - 3 types de ressources : *A* (10 instances), *B* (5 instances), and *C* (7 instances)
- A l'instant $T_0$ l'état du système est comme suit :

| | *Allocation* | *Max* |
|---|---|---|
| | *A B C* | *A B C* |
| $P_0$ | 0 1 0 | 7 5 3 |
| $P_1$ | 2 0 0 | 3 2 2 |
| $P_2$ | 3 0 2 | 9 0 2 |
| $P_3$ | 2 1 1 | 2 2 2 |
| $P_4$ | 0 0 2 | 4 3 3 |

- Déterminer le vecteur Disponible et la matrice Besoin ?
- Est ce que le système est dans un état sûr ?

# Évitement des inter-blocages

- Exemple
- Un Système composé de :
  - 5 processus $P_0$ through $P_4$
  - 3 types de ressources : *A* (10 instances), *B* (5instances), and *C* (7 instances)
- A l'instant $T_0$ l'état du système est comme suit :

| | *Allocation* | *Max* | *Disponible* | *Besoin* |
|---|---|---|---|---|
| | *A B C* | *A B C* | *A B C* | *ABC* |
| $P_0$ | 0 1 0 | 7 5 3 | 3 3 2 | 743 |
| $P_1$ | 2 0 0 | 3 2 2 | | 122 |
| $P_2$ | 3 0 2 | 9 0 2 | | 600 |
| $P_3$ | 2 1 1 | 2 2 2 | | 011 |
| $P_4$ | 0 0 2 | 4 3 3 | | 431 |

- Le système est dans un état sûr. La séquence P1,P3, P4, P2 et P0 satisfait le critère de sûreté
- Si le processus P1 fait une requête pour une instance supplémentaire du type de ressource A et deux instances du type de ressource C. Est-ce que le système doit accepter cette requête ?

# Évitement des inter-blocages

- Le processus P1 fait une requête pour une instance supplémentaire du type de ressource A et deux instances du type de ressource C. Requete1 = (1, 0, 2)

- Appliquons l'algorithme du banquier :
- Requete1 <= Besoin [1] : OK
- Requete1 <= Disponible : OK
- On **prétend** que la requête est acceptée et **on teste si le système est dans un état sûr** suite à cette requête

- A la suite de cette requête le système est d'état suivant

| | *Allocation* | *Max* | *Disponible* | *Besoin* |
|---|---|---|---|---|
| | *A B C* | *A B C* | *A B C* | *ABC* |
| $P_0$ | 0 1 0 | 7 5 3 | 2 3 0 | 7 4 3 |
| $P_1$ | 3 0 2 | 3 2 2 | | **0 2 0** |
| $P_2$ | 3 0 2 | 9 0 2 | | 600 |
| $P_3$ | 2 1 1 | 2 2 2 | | 011 |
| $P_4$ | 0 0 2 | 4 3 3 | | 431 |

- Le système est dans un état sûr. La séquence P1,P3, P4, P0 et P2 satisfait le critère de sûreté
- Est-ce qu'une requête (3,3,0) par le processus P4 peut être accordée dans cet état?
- Est-ce qu'une requête (0,2,0) par le processus P0 peut être accordée ?

# Références

[1] SILBERSCHATZ, A. et P.B. GALVIN, *Operating System Concepts*. 8th Edition, Addison Wesley.

[2] Coffman, E.G., Elphick, M.J. and Shoshani, A "System deadlocks" Computing survey, vol. 32, pp. 67-78, June 1971