Université Mohammed V- Rabat
École Mohammadia d'Ingénieurs
Département Génie Informatique

Filière Informatique

SYSTÈMES D'EXPLOITATION

Mr N. EL Faddouli (elfaddouli@emi.ac.ma)
Mme Z. Bakkoury (bakkoury@emi.ac.ma)

2022-2023

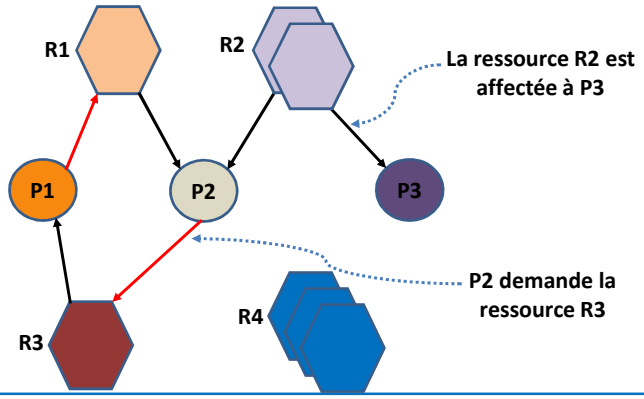
Plan du cours

- Introduction et Rappels
- Gestion des processus: Synchronisation avec attente active
  - Définitions et rappels
  - Algorithmes avec attente active
- Gestion des processus: Synchronisation sans attente active
  - Les sémaphores
  - Les moniteurs
  - L'interblocage (Dead-lock)
- Gestion de la mémoire
  - Définitions et rappels
  - La mémoire virtuelle
- Les entrées/Sorties

L'INTERBLOCAGE

Interblocage

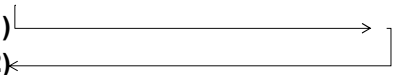
- [Tanenbaum] Un **ensemble** de processus est en interblocage si **chaque** processus, de l'ensemble, est bloqué en attente d'un événement (*une libération de ressource: un signal, un message, un sémaphore,* …) qui ne peut être causé que par un autre processus de l'ensemble.



Interblocage: Exemple

Utilisation de deux sémaphores, chacun pour obtenir une ressource.
Sémaphore: Mutex1 = Mutex2 = 1

Chaque mutex permet l'accès à une ressource



Prog1	**Prog2**
P(mutex1)	**P(mutex2)**
P(mutex2)	**P(mutex1)**
SC (*ressource 1 + ressource 2***)**	**SC (***ressource 1 + ressource 2***)**
V(mutex2)	**V(mutex1)**
V(mutex1)	**V(mutex2)**

Interblocage *(suite)*

Conditions provoquant un interblocage:

- **L'exclusion mutuelle** pour la prise d'une ressource: Chaque ressource est soit disponible, soit attribuée à un **seul** processus.
- **La détention et l'attente**: les processus ayant déjà obtenu des ressources peuvent en demander de nouvelles ressources détenues par d'autre processus.
- **Pas de réquisition**: les ressources doivent être libérées par le processus qui les détient. Ces ressources ne peuvent pas être retirées de force.
- **L'attente circulaire**: Deux ou plusieurs processus, chacun d'eux attendant une ressource détenue par un autre.

Interblocage *(suite)*

Stratégies de lutte contre les blocages mutuels

- **La politique de l'autruche**
- **Évitement de l'interblocage**
- **Prévention de l'interblocage**
- **Détection et résolution de l'interblocage**

Interblocage: La politique de L'autruche

- L'approche la plus simple
- Ignorer le problème (*Ignorer les possibilités d'interblocage*)
- Stratégie de la plupart des systèmes courants
- UNIX et Windows utilisent cette approche

Raisonnable si:

- Les interblocages se produisent rarement
- Le coût de la prévention est élevé

Interblocage: Évitement de l'interblocage

- Ce mécanisme contrôle pas à pas la progression d'un système de tâches (processus) d'une façon à garantir qu'aucun blocage ne se produira dans le futur.
- Le système doit avoir à l'avance des informations sur les futures demandes de chaque processus (***Nombre de chaque type de ressources***)
- **État sûr** : s'il n'est pas en interblocage et qu'il existe un ordre d'ordonnancement (d'allocation des ressources) dans lequel chaque processus peut être exécuté jusqu'à la fin, même si tous les processus demanderaient immédiatement toutes les ressources dont ils ont besoin pour finir leurs travaux.
- **État risqué** : à partir duquel l'interblocage est inéluctable
- **État d'interblocage**.

Interblocage: Évitement de l'interblocage *(suite)*

Principe : Si un processus présente au système une demande d'allocation de ressource, le système doit vérifier si le nouvel état obtenu serait un état "**risqué**".

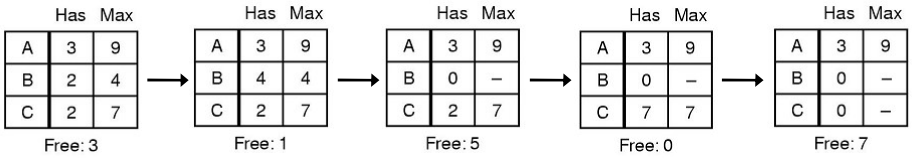
➔ Pour éviter les interblocages, il est important de ne pas s'avancer vers un état **non sûr** (risqué).

➔ Éviter dynamiquement les interblocages en allouant les ressources avec précaution.

Algorithme de banquier: Proposé par Dijkstra en 1965 en se basant sur l'utilisation des **matrices d'allocation des ressources.**

Interblocage: Évitement de l'interblocage *(suite)*

Exemple 1: Si tous les processus demandent leurs maximum de ressources, est-ce que l'état de départ est sécuritaire?



Exécution de tous les processus jusqu'à la fin

➔ l'état initial était sûr

Exemple 2:

	Allocation	*Max*	*Available*
	A B C	*A B C*	*A B C*
T_0	0 1 0	7 5 3	3 3 2
T_1	2 0 0	3 2 2	
T_2	3 0 2	9 0 2	
T_3	2 1 1	2 2 2	
T_4	0 0 2	4 3 3	

T1 → T3 → T0 → T2 → T4

Interblocage: Prévention de l'interblocage

- Inconvénients :
  - Algorithme relativement coûteux
  - Caractère irréaliste de la connaissance préalable des ressources nécessaires à la terminaison d'un processus
- Éviter les interblocages est pratiquement impossible

➔ Les prévenir en empêchant l'apparition des 4 conditions de leur existence.

Interblocage: Prévention de l'interblocage

Condition de détention et d'attente

- Obliger un processus à demander toutes les ressources d'un coup, en début d'exécution.
- Difficile de prévoir ces ressources.
- Les processus qui demandent beaucoup de ressources risquent d'attendre trop longtemps (famine)
- À chaque fois qu'un processus doit faire la demande de nouvelles ressources, le forcer à relâcher au préalable celles détenues.

Interblocage: Prévention de l'interblocage

Condition d'attente circulaire

Idée : Définir une relation d'ordre pour les ressources du systèmes.

Principe : un processus ne peut demander que les ressources dont le numéro est supérieur au numéro de toutes les ressources déjà détenues

Interblocage: Prévention de l'interblocage

Condition de non réquisition

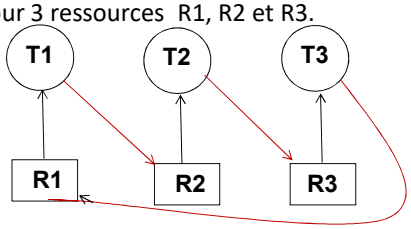
- Définir un ordre de priorité entre les processus
- Retirer les ressources au processus les plus (moins) prioritaires

➔ La prévention de l'interblocage dégrade la performance du système

Interblocage: Détection et résolution

- Le système construit dynamiquement un **graphe d'allocation** de ressources aux processus (threads) demandeurs:
  - Les ressources: $R = \{R_1, R_2, ..., R_m\}$
  - Les processus (Threads): $T = \{T_1, T_2, ..., T_n\}$
  - Arc d'affectation: Une ressource allouée à un processus $R_j \rightarrow T_i$
  - Arc de demande: Un processus attend la libération d'une ressource $T_i \rightarrow R_j$
- Le système doit vérifier s'il y a des processus en interblocage.
- **Exemple:** graphe d'allocation pour 3 ressources R1, R2 et R3.

Le processus T1 demande R1
Le processus T2 demande R2
Le processus T3 demande R3
Le processus T1 demande R2
Le processus T2 demande R3
Le processus T3 demande R1



Interblocage: Détection et résolution *(suite)*

Le système vérifie s'il y a des interblocages:

- À chaque modification du graphe suite à une demande d'une ressource (coûteuse en terme de temps CPU).
- À un intervalle régulier de temps
- Lorsqu'un indicateur d'interblocage est activé, par exemple le taux d'utilisation du CPU devient inférieur à un certain seuil (*la détection peut être tardive*)

Interblocage: Détection et résolution *(suite)*

Résolution:

• Tuer des processus (priorité, ...)

• Résoudre par la réquisition de ressource: Retirer temporairement une ressource à un processus pour l'attribuer à un autre (Quelle ressources? De quel processus?)

• Résoudre par rollback: Restaurer un état antérieur et éviter de retomber dans la même situation

➔ La reprise (*résolution*) n'est pas évidente.

Famine

- Un processus peut ne jamais avoir une ressource tout en n'étant pas en interblocage

- La famine peut être évitée en utilisant une politique d'allocation des ressources