Systèmes d'Exploitation Programmation Concurrente

INTERBLOCAGE (DEADLOCK)

1. Exemples introductifs
2. Conditions d'interblocage
3. Modélisation des interblocages -- Graphe d'allocation de ressources
4. Solutions à l'interblocage
   - ⇨ Ignorer l'interblocage
   - ⇨ Détection-guérison
   - ⇨ Prévention
   - ⇨ Evitement -- Algorithme »Banquier »

Ressources

- ✓ Les ressources sont, généralement, demandées par des appels système et allouées par le système d'exploitation
- ✓ **4 Etapes**

  1) Demande de la ressource (explicite ou implicite)

     ↳ Allocation ou Attente

  2) Allocation de la ressource

  3) Utilisation de la ressource

  4) Libération de la ressource (volontaire /Forcée --par préemption)

Interblocage --Définition

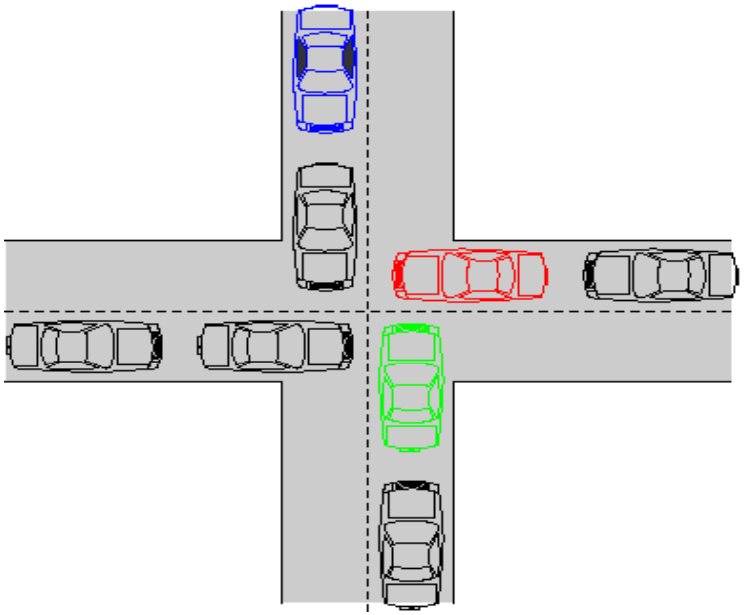
- **Interblocage --Deadlock**
  - Un ensemble de processus est dit en interblocage si chaque élément de cet ensemble attend un événement qui ne peut venir que d'un processus de cet ensemble
    - Exemple d'événement: libération d'une ressource

Famine (Privation) --Définition

- ✓ Un processus est en **famine** (starvation) lorsqu'il attend infiniment longtemps la satisfaction de sa requête (elle n'est jamais satisfaite).

Exemples Introductifs

✓ ***Exemple 1 :*** Considérons un croisement où les voitures s'immobilisent mutuellement : aucune voiture en peut progresser que si on ne prend pas les mesures spéciales.

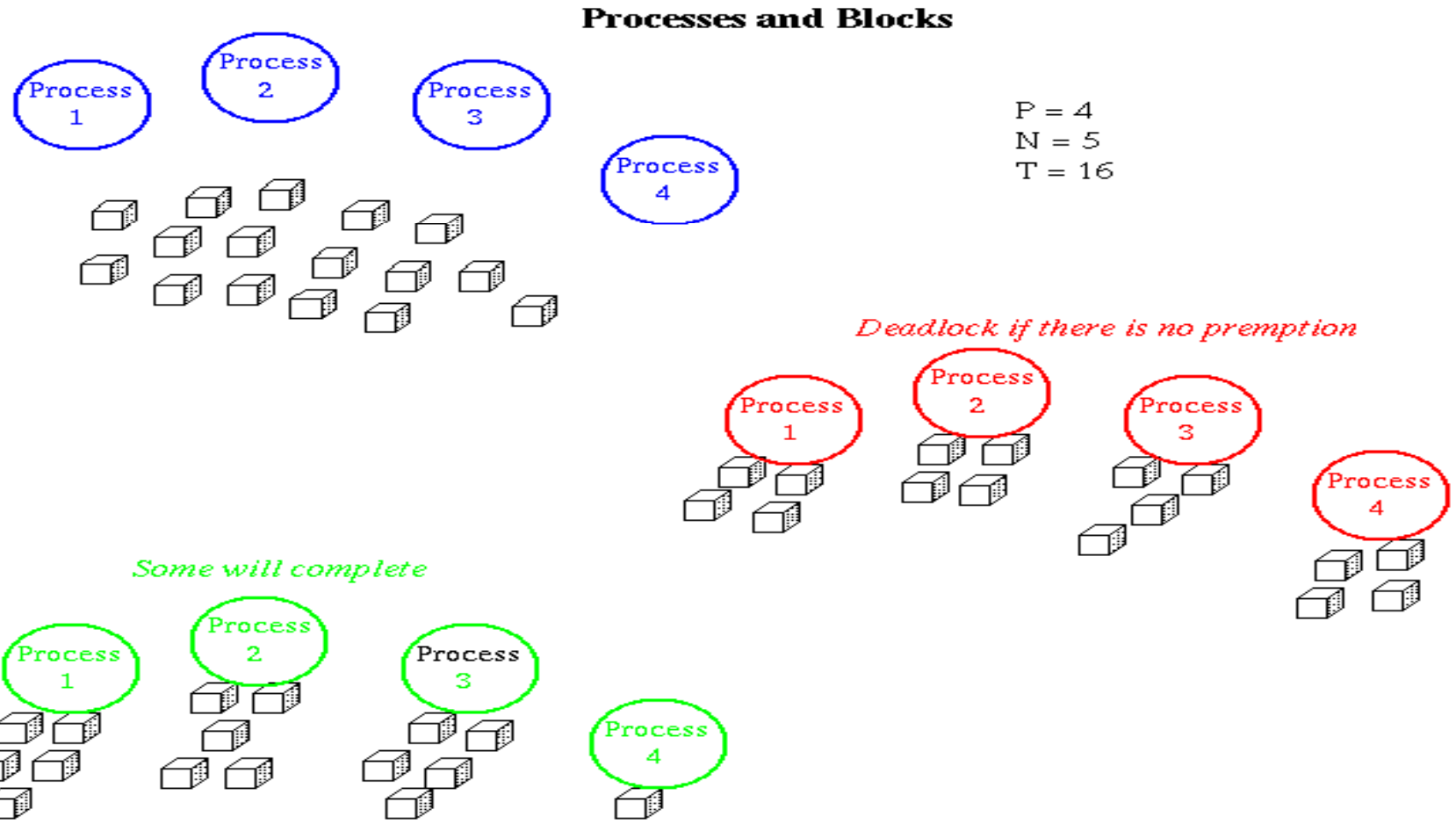


✓ ***Exemple 2 :*** Considérons deux processus P1 et P2 et x, y 2 sémaphores d'exclusion mutuelle:

P1	***P2***
…	….
P(x)	P(y)
P(y)	P(x)

Exemples Introductifs (suite)

✓ ***Exemple 3 :*** Processus et blocs -- Considérons avoir P processus et T blocs. Pour s'exécuter, chaque processus a besoin de N blocs.



Conditions d'apparition du phénomène d'interblocage

- La concurrence entre processus et l'allocation dynamique de ressources peuvent introduire un interblocage. Ce phénomène résulte de la conjonction de quatre circonstances :
  - **Exclusion mutuelle:** Chaque ressource est utilisable exclusivement par un processus, et un seul, car son partage donnerait des résultats incohérents.
  - **Pas de réquisition:** les ressources déjà allouées restent nécessaires aux processus qui les ont reçue et elles ne peuvent être réquisitionnées.
  - **Possession et attente:** chaque processus ne peut progresser que s'il obtient les ressources qu'il requiert dynamiquement par des demandes successives; il attend donc les ressources de sa dernière requête avant de poursuivre son déroulement.
  - **Attente circulaire:** plusieurs processus en attente de ressources allouées à d 'autres processus et il s'est formé une attente circulaire

Modélisation des interblocages

- Par un ***graphe d'allocation des ressources***, soit *GA(N, A)* orienté biparti où
  - ***N : 2 partitions de nœuds***
    - un processus P est représenté par 

    - une ressource R est représentée par 

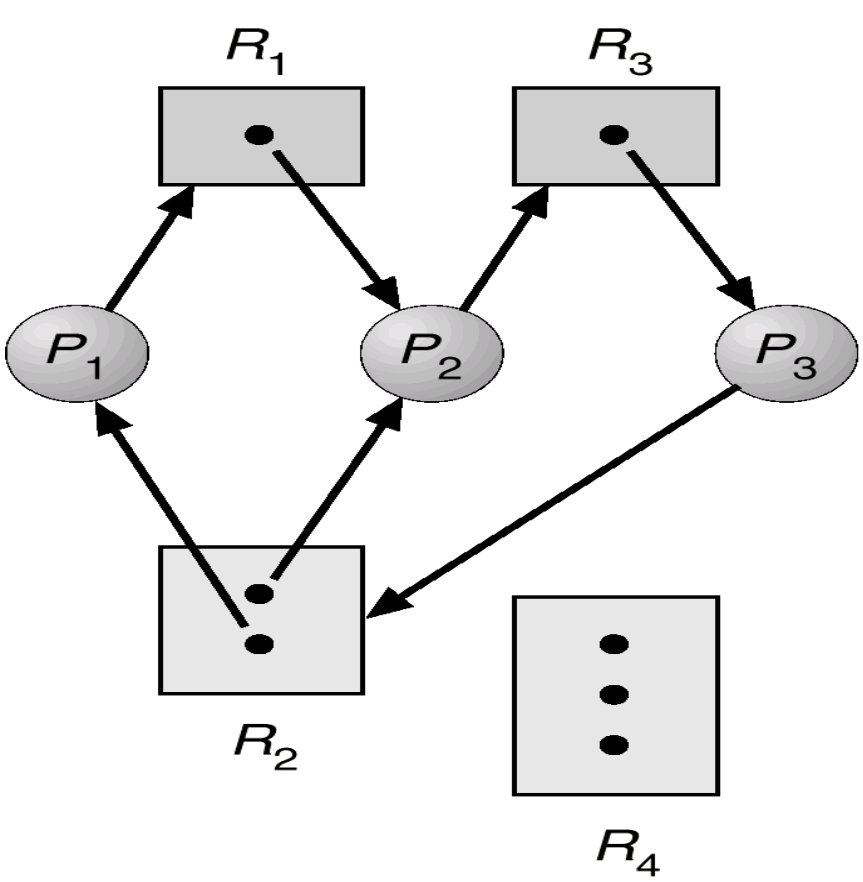
  - ***A : ensemble des arcs :***
    - un processus demande une ressource : 

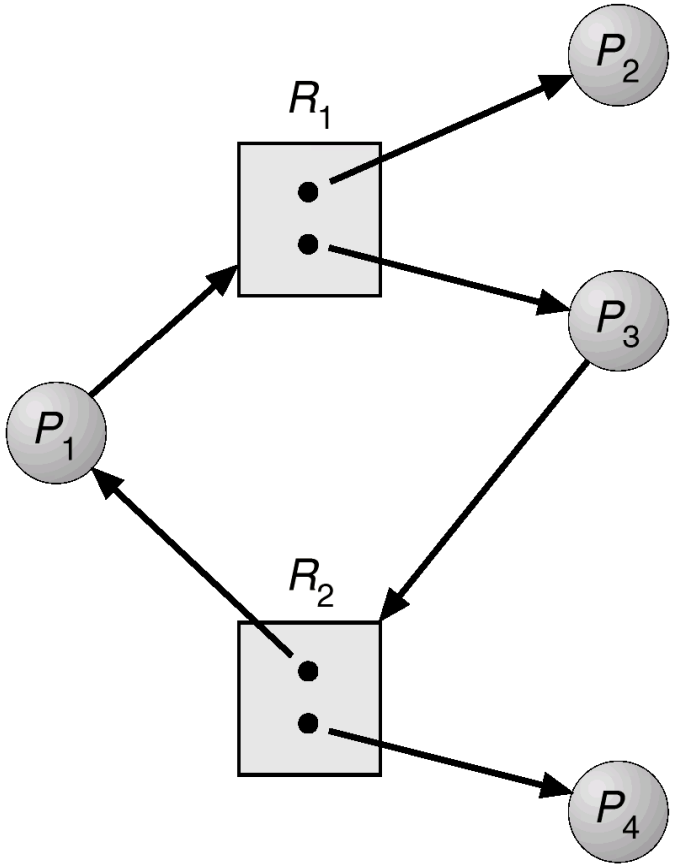
    - un processus utilise une ressource ou

      une ressource est allouée à un processus : 

Graphe d'allocation et Interblocages

Y a-t-il un deadlock?





Solutions à l'interblocage

① **Ignorer le blocage** (politique de l'autruche → ça n'arrivera pas

- ⇨ Approche adoptée par Unix ("don't worry, it will be fine").
- ⇨ Si cela arrive, relancer le système !

② **Détection-Guérison (solution optimiste)**

- ⇨ Laisser le blocage se produire puis après l'avoir détecté, de reprendre les ressources de certains.
- ⇨ Rompre le blocage par tuer un ou plus de processus, jusqu'à suppression des circuits (dans le graphe d 'allocation des ressources) -- Quelle est la victime?

Solutions à l'interblocage (suite)

③ ***Méthode préventive*** **(solution pessimiste)**

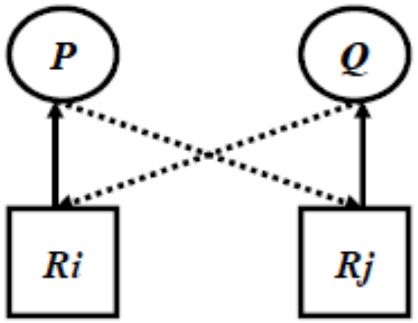
- Garantir qu'une situation d'interblocage ne se produira jamais
- Imposer des contraintes sur les demandes de manière que l'interblocage soit impossible
  - Demander toutes les ressources à l 'avance pas d 'attente
  - Exiger de connaître à l'avance les besoins et résultats
  - permettre la réquisition
  - Numéroter les ressources et les demandes doivent être faites selon l'ordre indiqué

④ ***Evitement de l'interblocage (solution*** **pessimiste*):***

- Faire en sorte que le deadlock ne puisse pas arriver:
  - Retarder une allocation possible si elle peut conduire à terme à un interblocage
  - Nécessité de connaître à l'avance les besoins futurs des processus

Prévention de l'interblocage

- ✓ Garantir qu'une situation de blocage ne se produira jamais
- ✓ Nécessité d'avoir un point de reprise (**checkpoint**)
  - ⇨ Travail depuis dernier point de reprise perdu



- ✓ Le blocage ne peut apparaître que si P demande la ressource Rj et Q demande la ressource Ri
  - ⇨ si i>j P ne peut demander Rj
  - ⇨ Si j>i Q ne peut demander Ri
  - ⇨ Les ressources sont uniques

- ✓ Exiger une connaissance a priori
- ✓ **Récapitulatif de la prévention**

Condition		*Approche*
Exclusion mutuelle	→	*Utiliser des ressources partageables*
Tenir et attendre	→	*Demander à l 'avance (preallaocation)*
Pas de réquisition	→	*Enlever des ressources (exiger de libérer avant de demander)*
Attente circulaire	→	*Ordonner les ressources*

Evitement de l'interblocage

- ✓ Avant toute allocation, évaluation du risque (futur) d'interblocage.
- ✓ Défini par Dijkstra, qui s'inspire de la restriction «Non tenir et demander » :
  - ⇨ N'autoriser une allocation que si **l'état résultant est sûre**; un état est dit sûre s'il existe une séquence d'autres états qui mènent les processus à avoir leurs ressources.

Evitement de l'interblocage –Algorithme du Banquier

- ✓ **Algorithme du « Banquier » avec plusieurs types de ressources**
  - ⇨ **Données** : 2 matrices, A et D (stockent l 'information sur les allocations de ressources et les besoins en ressources) et 3 vecteurs --RE/RP/RD-- qui désignent les ressources existantes/en possession/disponibles :
    - ↳ La matrice A enregistre les ressources allouées actuellement à chaque processus
    - ↳ La matrice D enregistre les ressources demandées pour chaque processus pour terminer
  - ⇨ **Comment peut-on connaître un état sécurisant?**
    1. Trouver une ligne r de D avec r ≤ RD; si r n'existe pas, le système se bloque (aucun processus ne va terminer son exécution)
    2. Marquer le processus, de la ligne r, qui s'est terminé → MAJ(RP, RD)
    3. Répéter 1, 2 jusqu'à marquage de tous les processus terminés → état sécurisant/deadlock.

Evitement du blocage -- Exemple

	R1	R2	R3	R4
P1	3	0	1	1
P2	0	1	0	0
P3	1	1	1	0
P4	1	1	0	1
P5	0	0	0	0

A

	R1	R2	R3	R4
P1	1	1	0	0
P2	0	1	1	2
P3	3	1	0	0
P4	0	0	1	0
P5	2	1	1	0

D

- ✓ RE = [6, 3, 4, 2]; RP = [5, 3, 2, 2]; RD = [1, 0, 2, 0]
- ✓ Est-ce que l'état résultant est sécurisant?
  - ⇨ Oui : P4; P1; P2; P3; P5
- ✓ Différer les demandes des processus qui mènent à des états non sécurisants
- ✓ ***Problème :*** les processus ne peuvent pas toujours connaître leurs besoins en ressources; le nombre de processus varie dynamiquement; les ressources peuvent être en échec.

CONCLUSION

- ✓ L'algorithme de ***détection d'interblocage*** est coûteux à exécuter
  - ⇨ L'interblocage peut être rompu soit par un abandon d'un processus soit réclamer quelques ressources; le rechargement est difficile
  - ⇨ Cette méthode est valable si la fréquence des blocages est faible, et que le déblocage reste facile
- ✓ ***La méthode préventive*** d'interblocage est assez restrictive pour une utilisation générale, mais peut être utilisée pour des cas spécifiques
  - ⇨ Nécessité d'une connaissance anticipée
- ✓ ***L'évitement du blocage*** nécessite une information qui est toujours non disponible
- ✓ Les algorithmes basés sur la réquisition des ressources et le fait de reprendre encore peut générer la ***famine*** (attente infinie par coalition)
  - ⇨ Changer le comportement des processus, ou changer la politique