*Chapitre 4 : L’inter blocage*

* Modèles
* Prévention
* Evitement
* Détection/ Guérison

|  |
| --- |

1. **Introduction**

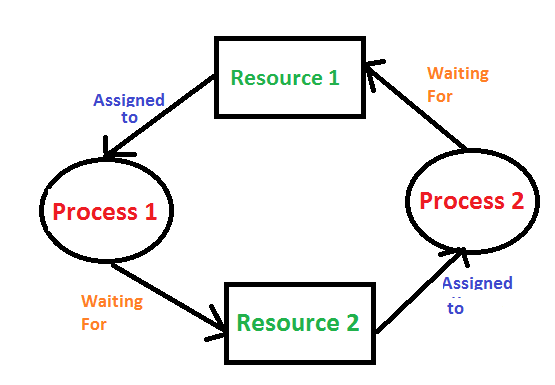
Un système d’interblocage est un système utilisé pour maintenir un état de fonctionnement sûr des machines, qui détecte les conditions anormales ou les séquences incorrectes et interrompt toute action ultérieure ou lance une action corrective.

**Exemple** :

Un système de verrouillage de vanne est un ensemble de verrouillage à clé piégée qui verrouille la vanne dans une ou deux positions (ouverte et/ou fermée) avec une clé piégée dans l'ensemble de verrouillage et une clé libre.

1. **Définition de l’interblocage**

Un ensemble de processus est en interblocage, si chaque processus attend un événement que seul un autre processus de l’ensemble peut provoquer.



*Figure 13 Schéma d'interblockage*

Le blocage est une situation où un ensemble de processus est bloqué parce que chaque processus détient une ressource et attend une autre ressource acquise par un autre processus.

Prenons un exemple, lorsque deux trains se rapprochent sur la même voie et qu'il n'y a qu'une seule voie, aucun des trains ne peut se déplacer une fois qu'ils sont l'un devant l'autre.

|  |
| --- |

*Figure 12 Problème de circulation routière*

Une situation similaire se produit dans les systèmes d'exploitation lorsqu'il y a deux processus ou plus qui détiennent certaines ressources et attendent des ressources détenues par d'autres. Par exemple, dans le diagramme ci-dessus, le processus 1 contient la ressource 1 et attend la ressource 2 qui est acquise par le processus 2, et le processus 2 attend la ressource 1

1. **Caractérisation de l’interblocage**

Une situation d’interblocage peut survenir si les quatre conditions suivantes se produisent simultanément (conditions nécessaires, mais non suffisantes) :

1. **Exclusion mutuelle :** Le système a des ressources non partageables (un seul processus à la fois peut s’en servir). Comme exemple : Processeur, une zone de mémoire, un périphérique, mais aussi un sémaphore, un moniteur, une section critique.
2. **Occupation et attente (hold and wait) :** Un processus détient au moins une ressource non partageable et qui attend d’acquérir des ressources supplémentaires détenus par d’autres processus.
3. **Pas de réquisition (préemption) :** Un processus qui a saisi une ressource non partageable, la garde jusqu’à ce qu’il aura complété sa tâche.
4. **Attente circulaire :** Il y a un cycle de processus tel que chaque processus pour compléter doit utiliser une ressource non partageable qui est utilisée par le suivant, et que le suivant gardera jusqu'à sa terminaison
5. **Méthodes de traitement de l’interblocage**

Trois méthodes principales sont possibles pour traiter le problème de l’interblocage :

* + - 1. Employer un protocole pour assurer que le système ne se trouvera jamais dans une situation d’interblocage (prévention).
      2. Permettre que le système se trouve en situation d’interblocage et le corriger ensuite.
      3. Ignorer complètement le problème d’interblocage et supposer que le système ne se trouvera jamais dans de telles situations.

Dans ce qui suit, ces trois techniques seront abordées.

1. **Prévenir les interblocages**

Pour prévenir un interblocage, il faut éviter la réalisation d’au moins une des conditions citées ci-dessous.

* + **Exclusion mutuelle**

Réduire le plus possible l’utilisation des ressources partagées et sections critiques. En général, ceci n’est pas possible puisque certaines ressources sont intrinsèquement non partageables.

* + **Occupation et attente**

Un processus qui demande de nouvelles ressources ne devrait pas en retenir d’autres (les demander toutes ensemble). Une situation de famine se caractérise par l’attente d’une ressource non satisfaite pendant un certain temps.

* + **Pas de réquisition**

Si un processus qui demande d’autres ressources ne peut pas les avoir, il doit être suspendu, ses ressources doivent être rendues disponibles. Cela peut s’appliquer aux ressources dont l’état peut être facilement sauvegardé et restauré (registre UC ou zones mémoires)

* + **Attente circulaire**

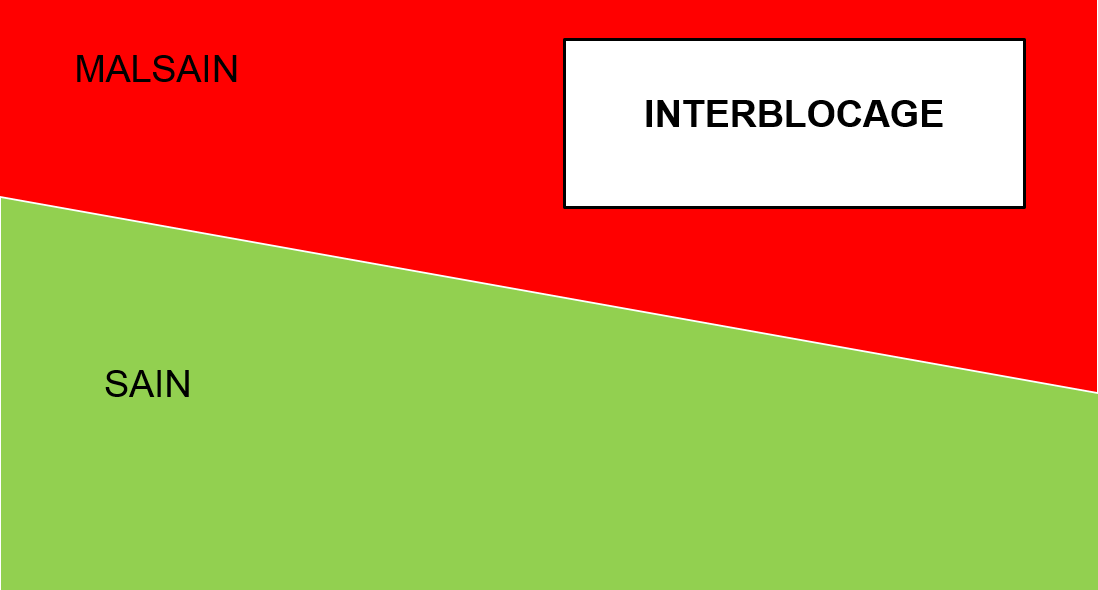
Imposer un ordre total sur les ressources. Un processus doit demander les ressources dans cet ordre (comme exemple : tout processus doit toujours demander une imprimante avant de demander une unité de disque)

L’utilisation d’une autre méthode, à savoir éviter les interblocages en demandant des informations supplémentaires sur la façon dont les ressources vont être requises. Cette méthode est présentée dans la section suivante.

1. **Éviter les interblocages**

Pour éviter un éventuel futur interblocage, il faut savoir quelles sont les ressources disponibles, les ressources allouées à chaque processus et les futures requêtes de chaque processus. Sur la base de ces informations, un examen dynamique est effectué sur l’état d’allocation des ressources afin d’assurer qu’il ne puisse jamais exister une condition d’attente circulaire. Ainsi, Il faut éviter d’accepter toute requête qui fait passer un système d’un état sain vers un état malsain.

Puisque l’ensemble des états d’interblocage est un sous ensemble des états malsains (voir figure ci-dessous). Un état est dit sain s’il existe à partir de cet état une séquence saine.

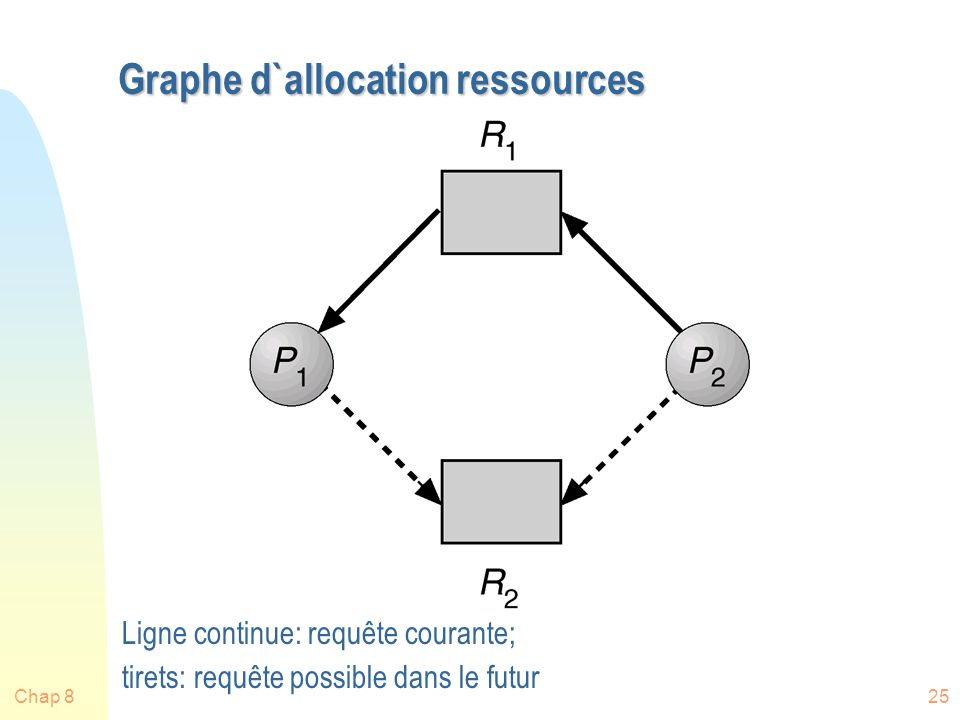
Sinon, l’état est dit malsain. Si à partir d’un état donné, l’ensemble des processus du système peuvent terminer normalement dans un ordre donné leur exécution, alors cette succession de terminaisons est dite séquence saine.

Pour traiter le problème d’interblocage d’une façon effective, nous allons définir, dans la section suivante, la notion de graphe d’allocation de ressources.

1. **Graphe d’allocation de ressources**
   * **Définitions du graphe**

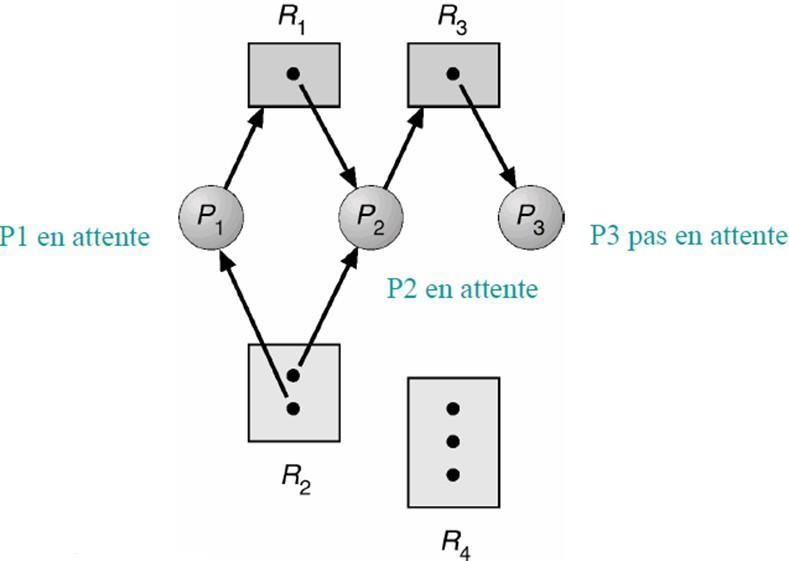
* G = (V, E) où V : ensemble de sommets et E : ensemble d’arrêtes.
* V est partitionné en deux sous-ensembles :
  + - P = P1, P2, . . . , Pn, l’ensemble de tous les processus.
    - R = R1, R2, . . . , Rm, l’ensemble de tous les types de ressources.
* E contient :
  + - Arrête requête : arête dirigée de Pi vers Rk (Pi a besoin d’un exemplaire de Rk)
    - Arrête affectation : arête dirigée de Ri vers Pk (Pi détient un exemplaire de Rk)
    - Arête de réclamation : arête dirigée de Pi vers Rk (Pi peut demander dans le futur un exemplaire de Rk)

Voir ce schéma explicite.

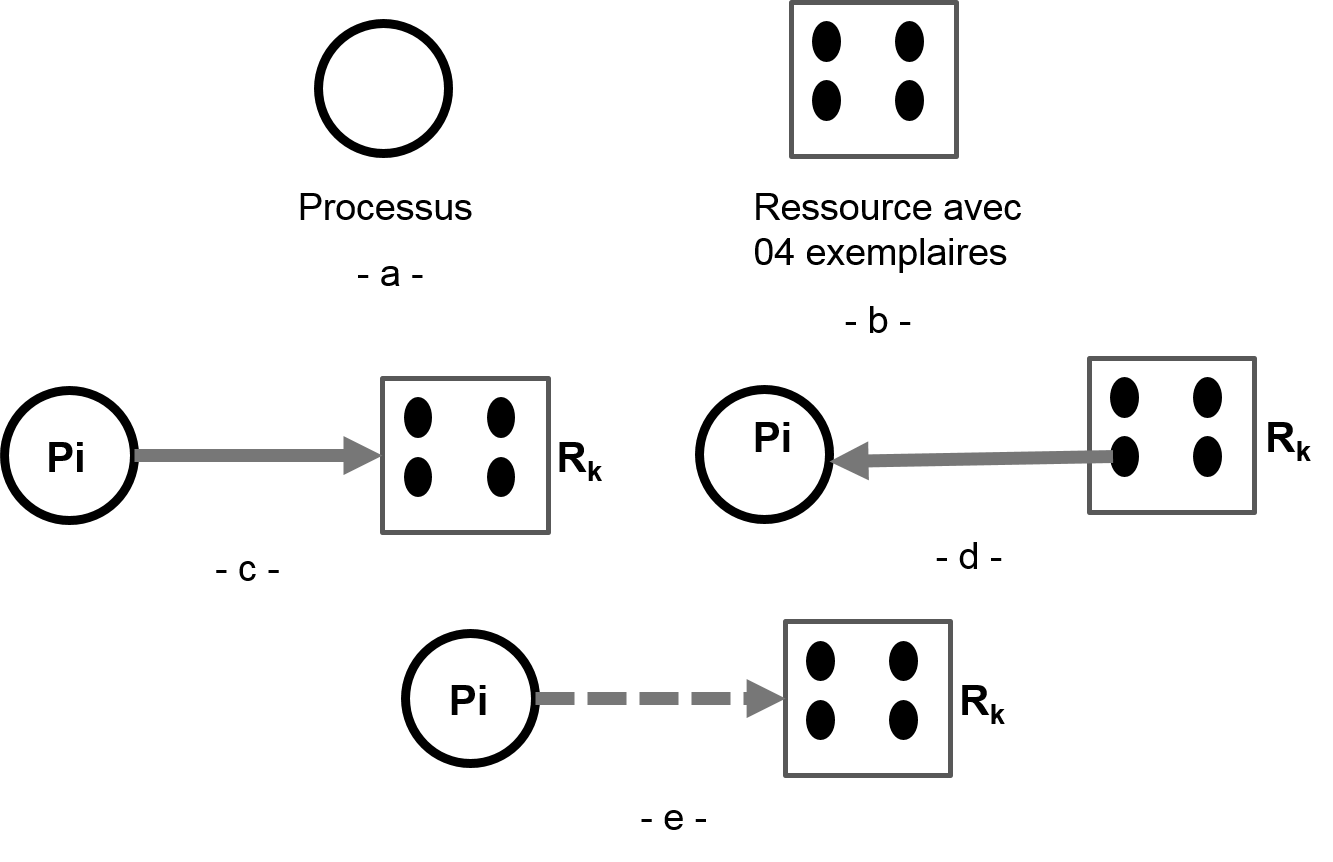


*Figure 13 Schéma explicite d’un graphe d’allocation de ressources*

* + **Algorithme pour éviter l’interblocage**

Il s’agit de construire un graphe d’allocation de ressources et voir s’il y a une manière dont tous les processus peuvent terminer. Dans le cas d’une ressource par type, l’algorithme cherche des circuits dans le graphe (algorithme d’ordre n2 , avec n = nombre de processus). Par contre, cet algorithme n’est pas applicable dans le cas de plusieurs ressources par type. 

* + **Algorithme du banquier**

Cet algorithme pourrait s’appliquer dans un système bancaire pour s’assurer que la banque ne prête jamais son liquide disponible de telle sorte qu’elle ne puisse plus satisfaire tous ses clients. 

Il est applicable à un système d’allocation de ressources avec possibilité de plusieurs instances pour chaque type de ressource. En entrant dans le système, tout processus doit déclarer le nombre maximal d’instances de chaque type de ressource dont il a besoin (il ne doit pas excéder le nombre total de ressources).

Quand un processus requiert un ensemble de ressources, le système doit déterminer si cette allocation laissera le système dans un état sain.

* Trouver un processus Pi non marqué dont la rangée i de R est inférieure à A
* Si un tel processus n’existe pas, alors l’état est non sûr (il y a interblocage). L’algorithme se termine.
* Sinon, ajouter la rangée i de C à A, et marquer le processus.
* Si tous les processus sont marqués, alors l’état est sûr et l’algorithme se termine, sinon aller à l’étape une.

L’algorithme du banquier permet bien d’éviter les interblocages, mais il est peu utilisé en pratique, car on ne connaît pas toujours à l’avance les besoins en ressources des processus.

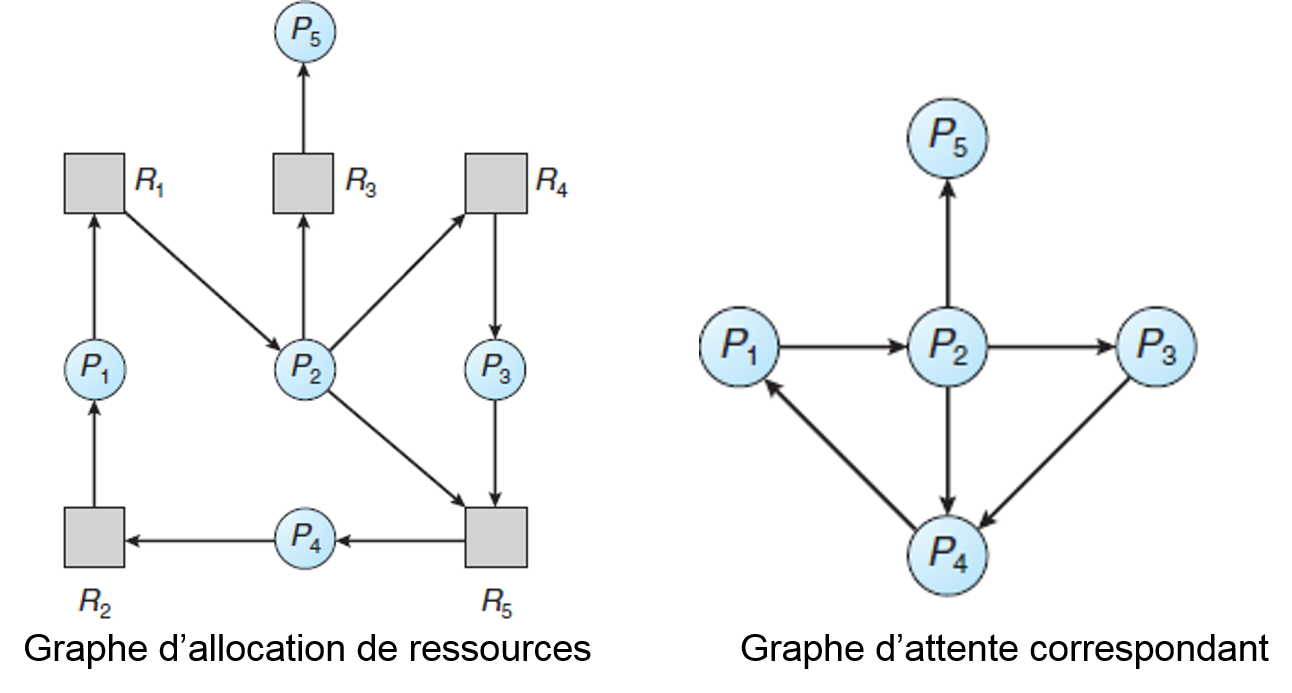
1. **Détecter et corriger les interblocages**

Si un système n’emploie pas d’algorithme pour prévenir ou pour éviter les interblocages, il doit, en cas de situation d’interblocage, détecter cet interblocage ensuite le corriger.

* + **Détecter les interblocages**

**Instance unique pour chaque type de ressource**

Dans ce cas, le système doit assurer la maintenance du graphe d’attente et appeler périodiquement un algorithme qui cherche un circuit dans ce graphe.

Le graphe d’attente est obtenu à partir du graphe d’allocation de ressources en éliminant les nœuds représentant les ressources. Ainsi, le graphe d’attente représente un sous graphe du graphe d’allocation de ressources.

*Figure 17 Exemple d’un graphe d’attente*

1. **Corriger les interblocages**

Deux alternatives sont possibles pour corriger un interblocage détecté :

* Manuellement en informant l’opérateur.
* Automatiquement par le système.

Dans le cas d’une correction automatique, le système peut procéder par l’une des deux possibilités suivantes :

* Terminaison de processus : Dans ce cas, il est possible de :
  + Avorter tous les processus en interblocage, ce qui va rompre le circuit d’interblocage, mais à un cout très élevé. En effet, tous les calculs partiels effectués par ces processus seront perdus et recalculés plus tard.
  + Avorter un processus a` la fois jusqu’à ce que le circuit d’interblocage soit éliminé. Ceci provoque une surcharge considérable puisque après chaque avortement, on doit appeler l’algorithme de détection de circuit.
* Réquisition de ressources : Enlever des ressources à un processus et les affecter à un autre pour défaire le circuit d’interblocage. Cette réquisition nous oblige à réfléchir sur :
  + La sélection d’une victime en précisant quelles ressources seront réquisitionnées et à partir de quels processus. On peut prendre comme facteur de coût le nombre de ressources détenues et le temps d’exécution.
  + Le retour en arrière puisqu’une perte de ressources implique forcément un redémarrage du processus à partir d’un état sain (état initial).
  + Comment garantir que les ressources ne seront pas réquisitionnées a` partir du même processus, en d’autres termes comment éviter la famine.

1. **Conclusion**

Une situation d’interblocage se produit si un ou plusieurs processus attendent indéfiniment un évènement qui ne peut être provoqué que par l’un des processus en attente. Le concepteur du système d’exploitation peut adopter l’une des approches suivantes vis à vis du problème d’interblocage :

* Prévenir ou éviter
* Détecter et corriger
* Ignorer

Aucune de ces approches n’est adéquate à elle seule pour tous les types de problème d’allocation de ressources dans les SE. Ainsi, ces approches peuvent être combinées, en permettant d’utiliser séparément une approche optimale pour chaque classe de ressources dans le SE.