Systèmes d'Exploitation - ENSIN6U3

Allocation des ressources

Leonardo Brenner [1] Jean-Luc Massat [2]

[1] Leonardo.Brenner@univ-amu.fr

[2] Jean-Luc.Massat@univ-amu.fr

Aix-Marseille Université

Faculté des Sciences

Table de matière

Définition des ressources

Définition d'une ressource

Une *ressource* est un objet utilisable par une tâche. Elle est caractérisée par :

- l'existence d'un *mode d'emploi* ;
- un état (*libre* ou *allouée*) ;
- l'existence d'un allocateur que répond aux requêtes.

Type de ressources

On distingue les ressources :

- banalisées : qui ont des occurrences multiples (imprimantes, canaux d'E/S) ;
- réquisitionnables : CPU, mémoire ;
- physiques ou logicielles ;
- partageables ou réentrantes : pour le code d'un programme.

Utilisation des ressources

Étapes d'utilisation

- Demande : représente l'étape de demande d'une ressource. Si la demande ne peut pas être satisfaite, on la met en attente ;
- Utilisation : le processus peut utiliser la ressource ;
- Libération : le processus libère la ressource demandée et allouée.

Objectif de l'allocation de ressources

Objectif

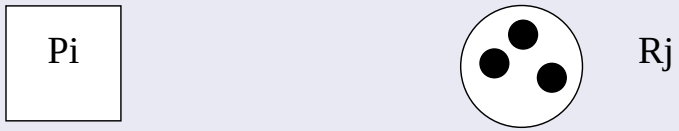
Les allocateurs de ressources doivent :

- être *équitables* (en respectant les priorités) ;
- éviter la *privation* (attente sans fin d'une ressource) ;
- éviter l'apparition d'un *interblocage* ;
- éviter une *congestion* en veillant à :
  - identifier une demande excessive de ressources ;
  - ne pas accepter de demandes quand le système est en surcharge.

Le graphe d'allocation de ressources

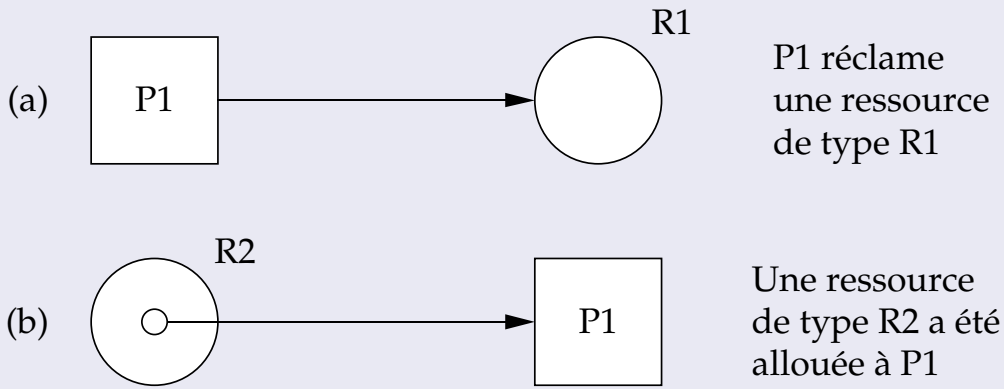
Graphe d'allocation

L'état des ressources est décrit au moyen d'un graphe de l'allocation de ressources.
Un processus P_i et une ressource R_j ayant 3 occurrences sont décrits par :



Demandes d'allocation

Les demandes d'allocation et les ressources allouées sont décrits par :



Les interblocages

Définition - Interblocage

Attente indéfinie pour un ou plusieurs processus d'une ou plusieurs ressources détenues par des processus en attente de ressources.

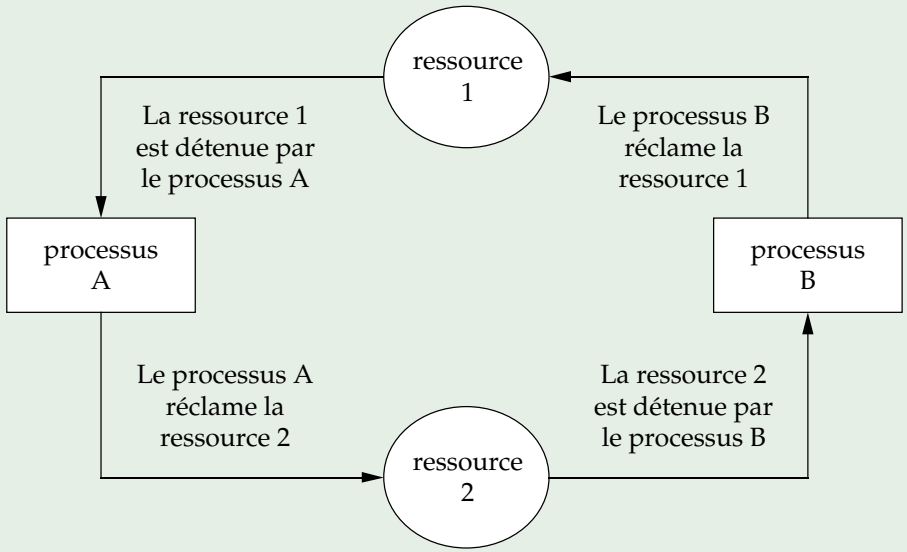
Conditions d'apparition

- **Exclusion mutuelle** : les ressources ne sont pas partageables ;
- **Détention et attente** : chaque processus utilise simultanément plusieurs ressources ;
- **Pas de préemption** : les ressources ne sont pas réquisitionnables ;
- **Attente circulaire** : il existe un ensemble de processus $(p_0, ..., p_n)$ tel que :
  - p_i attend p_{i+1} pour $0 \leq i < n$,
  - p_n attend p_0.

Un exemple d'interblocage

Deux processus en interblocage

Un interblocage représenté par un graphe d'allocation des ressources :



P_a

(1) prendre R_1

⋮

(3) prendre R_2

⋮

P_b

⋮

(2) prendre R_2

⋮

(4) prendre R_1

Traitement des interblocages

Comment traiter les interblocages

- *Ignorer les problèmes* : politique de l'autruche ;
- La *prévention* : On impose des règles strictes qui évitent l'apparition d'un interblocage ;
- L'*évitement* : On surveille l'apparition d'un interblocage en imposant des contraintes moins strictes ;
- La *détection* : Un algorithme à la charge de détecter les interblocages ;
- La *guérison* : Suppression éventuelle de l'un des processus.

Ignorer les problèmes

Politique de l'autruche

Cette stratégie consiste à ignorer complètement les possibilités d'interblocages. C'est à les utilisateurs de coder correctement leurs applications et de surveiller à la bonne exécution.

Critiques

+ Solution adoptée par la plupart des systèmes d'exploitation courants ;
+ Très rapide et légère, il n'y a rien à implémenter ;
- Aucune garantie sur le bon fonctionnement du système.

Prévention des interblocages (1/5)

Annonce des demandes

Annoncer les demandes de ressources nécessaires avant de démarrer un processus. Le processus ne démarre que si tous les ressources demandées sont disponibles.

Critiques

+ solution parfaite ;
- très mauvaise utilisation des ressources ;
- programmes difficiles à écrire (on ne connaît pas les ressources en avance) ;
- risque de famine.

Prévention des interblocages (2/5)

Libération et re-allocation des ressources

À chaque demande d'allocation d'une ressource supplémentaire, il faut libérer toutes les ressources détenues et les redemander en y ajoutant la ressource supplémentaire.

Critiques

- très mauvaise utilisation des ressources (perte au milieu de l'utilisation) ;

Prévention des interblocages (3/5)

Virtualisation de ressources

Diminuer le nombre de ressources qui ne peuvent pas être réquisitionnées.

- interdire l'accès direct à la ressource,
- remplacer la ressource **réelle** par une ressource **virtuelle**.

Exemple de virtualisation de ressources

- processeur $\longrightarrow$ processus ;
- mémoire $\longrightarrow$ mémoire virtuelle ;
- imprimante $\longrightarrow$ queue d'impression ;
- écran $\longrightarrow$ double-buffer, display ;
- machine $\longrightarrow$ machine virtuelle.

Prévention des interblocages (4/5)

Ressources réquisitionnables

Pour supprimer la contrainte de non réquisition. On peut utiliser le protocole suivant :

- un processus détenant certaines ressources en demande une autre
- si la ressource peut être allouée immédiatement, alors le processus continue avec la ressource demandée ;
- sinon, toutes les ressources actuellement allouées au processus sont réquisitionnées.

Critiques

- Lenteur
- Risque de famine

Prévention des interblocages (5/5)

Classement des ressources

Classer les ressources suivant un ordre, et respecter cet ordre lors des demandes de ressources.

Critiques

+ La condition d'attente circulaire est impossible donc pas d'interblocage possible ;
- Cet ordre doit respecter une certaine logique, ... mais laquelle ?
- La contrainte est très lourde ;
- La portabilité des applications est médiocre ;
- Que faire quand on ajoute une nouvelle ressource ?

Évitement des interblocages

Politique d'évitement : Prudence

On évite les interblocages en adoptant un comportement *prudent*. La méthode la plus connu est l'*algorithme des banquiers*.

Principe

Cet algorithme consiste à examiner chaque nouvelle requête pour voir si elle conduit à un état **sain**. Si c'est la cas, la ressource est allouée, sinon la requête est mise en attente.

- le S.E. connaît les demandes maximales ;
- les allocations/libérations sont libres.

État sain

Un système est dans un état sain s'il existe une séquence saine. Une séquence de processus $P_1, ..., P_n$ est une séquence saine pour l'état d'allocation courant si, pour chaque P_i, les requêtes des ressources de P_i peuvent être satisfaites par les ressources disponibles, plus les ressources détenues par tous les P_j, avec $j < i$.

L'algorithme des banquiers (1/2)

Structures de données

- M : nombre de classes de ressources ;
- N : nombre de processus ;
- $dispo_j$ pour $j \in \{1, ..., M\}$;
- $max_{i,j}$ pour $i \in \{1, ..., N\}, j \in \{1, ..., M\}$;
- $alloc_{i,j}$ pour $i \in \{1, ..., N\}, j \in \{1, ..., M\}$.

Conditions d'exécution

Un processus P_i peut s'exécuter ssi

$$\forall j \in \{1, ..., M\}, (max_{i,j} - alloc_{i,j}) \leq dispo_j$$

Un ordre d'exécution $P_{k_1}, ..., P_{k_n}$ est dit *sain* ssi les processus peuvent s'exécuter dans cet ordre **les uns après les autres**.
Si,dans un système d'allocation de ressources, il existe un ordre d'exécution sain, alors il n'y a pas d'interblocages.

L'algorithme des banquiers (2/2)

Algorithme d'allocations

Algorithme d'allocation de R_j à P_i :

1. $dispo_j > 0$?
2. les annonces sont-elles respectées ?
3. si R_j est allouée à P_i, l'état est-il sain ?
4. si la réponse est négative, suspendre le processus P_i.

Critiques

+ Seule les requêtes laissant le système dans un état sain sont honorées ;
- Présuppose le nombre de ressources invariant ;
- Il faut connaître le nombre maximale de ressources que le processus aura besoin.

Détection des interblocages (1/3)

Principe

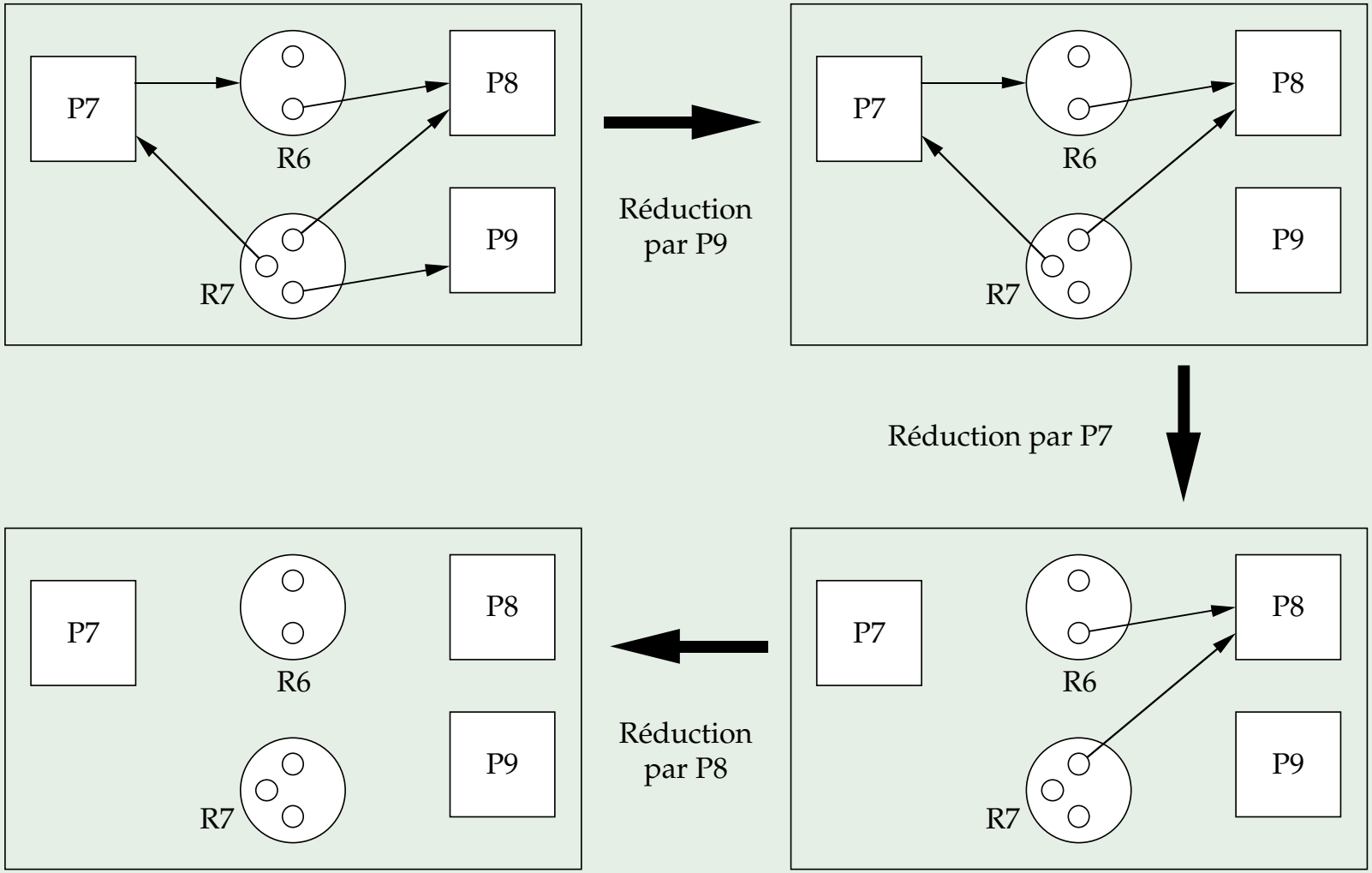
- Le système ne cherche pas à empêcher les interblocages ;
- À pour but de détecter les attentes circulaires ;
- Coût exploitation non négligeable.

Différents types de vérification

- À chaque modification du graphe suite à une demande d'une ressource ;
- Périodiquement ;
- Faible utilisation du processeur.

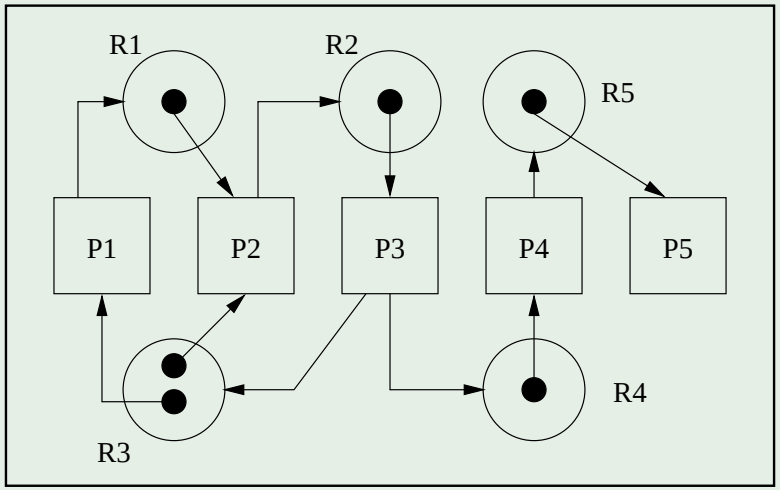
Détection des interblocages (2/3)

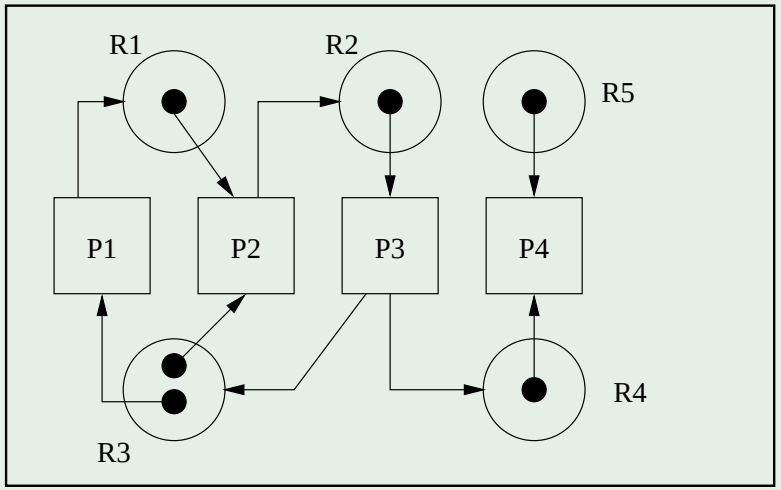
Détection par graphe d'allocation

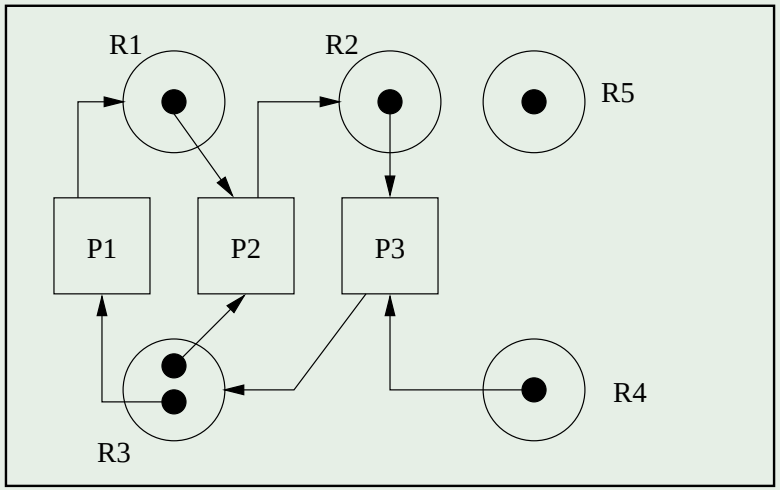


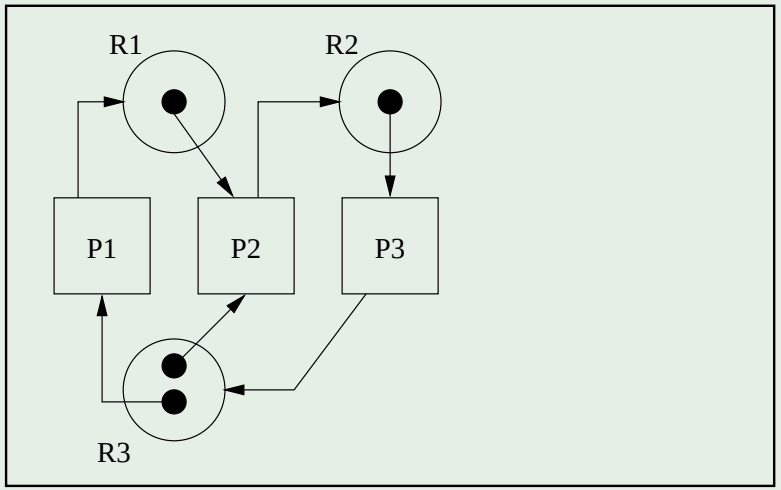
Détection des interblocages (3/3)

Détection par graphe d'allocation









Guérison des interblocages

Comment guérir des interblocages

- Retirer temporairement une ressources à un processus ;
- Restaurer un état antérieur du système ;
- Supprimer un ou plusieurs processus.

Problèmes

- Quel processus supprimer ? ou retirer les ressources ?
- Comment revenir à un état sain ?