



Systèmes
d'Exploitation

Didier Verna
EPITA

Théorie

Traitement

Prévention

Évitement

Détection

Correction

Systèmes d'Exploitation

Interbloquages

Didier Verna

didier@lrde.epita.fr

<http://www.lrde.epita.fr/~didier>



Problématique

Extrait de la législation du Kansas (1900)

Systèmes
d'Exploitation

Didier Verna
EPITA

Théorie

Traitement

Prévention

Évitement

Détection

Correction

« Quand deux trains approcheront du croisement, les deux devront s'arrêter complètement, et aucun d'eux ne redémarrera jusqu'à ce que l'autre soit reparti. »



Table des matières

Systèmes
d'Exploitation

Didier Verna
EPITA

Théorie

Traitement

Prévention

Évitement

Détection

Correction

1 Théorie

2 Traitement des interbloquages

- Prévention des interbloquages
- Évitement des interbloquages
- Détection des interbloquages
- Correction des interbloquages



■ Hypothèses

- ▶ Plusieurs types de ressource (CPU, mémoire, disque, imprimantes *etc.*)
- ▶ Une ou plusieurs instances de chaque ressource
- ▶ Allocation indifférente de n'importe quelle instance d'un type donné
- ▶ Utilisation mutuellement exclusive d'une instance de ressource

■ Classes de ressources

- ▶ **Ressources réquisitionnables** : récupération de la ressource pendant son utilisation, sans effet indésirable (ex. mémoire)
- ▶ **Ressources non réquisitionnables** : graveurs, imprimantes *etc.*
- ▶ Les interbloquages concernent surtout les ressources non réquisitionnables



Séquence d'utilisation

Allocation \Rightarrow utilisation \Rightarrow libération

Systèmes
d'Exploitation

Didier Verna
EPITA

Théorie

Traitement

Prévention
Évitement
Détection
Correction

■ En cas d'échec de l'allocation

- ▶ Processus bloqué (puis réveillé) automatiquement, ou code d'erreur en retour
- ▶ Dans tous les cas, on peut considérer le processus comme « bloqué »

■ Exemple d'interblocage

Processus 1 :

```
wait (sem1);  
wait (sem2);
```

Processus 2 :

```
wait (sem2);  
wait (sem1);
```



Conditions d'interblocage

Nécessaires mais non suffisantes

Systèmes
d'Exploitation

Didier Verna
EPITA

Théorie

Traitement

Prévention
Évitement
Détection
Correction

- 1 **Exclusion mutuelle** : chaque ressource est soit libre, soit occupée par un unique processus
- 2 **Occupation et attente** : chaque processus occupant une ressource peut en attendre une autre
- 3 **Pas de réquisition** : seul le processus occupant une ressource peut la libérer
- 4 **Attente circulaire** : boucle d'attente de ressources occupées par les processus

Remarque : attente circulaire \implies occupation et attente



Graphe d'allocation de ressources

Systèmes
d'Exploitation

Didier Verna
EPITA

Théorie

Traitement

Prévention
Évitement
Détection
Correction

- **Graphe orienté**
- **Deux types de noeuds**
 - ▶ processus P_i
 - ▶ ressources R_j
- **Deux types d'arcs**
 - ▶ Arcs de requête $P_i \rightarrow R_j$
 - ▶ Arcs d'affectation $R_j \rightarrow P_i$
- Pour qu'il y ait interblocage, le graphe d'allocation de ressources doit nécessairement contenir un cycle.
Cette condition devient suffisante s'il n'existe qu'une seule instance de chaque ressource.



Exemple

... et contre-exemple

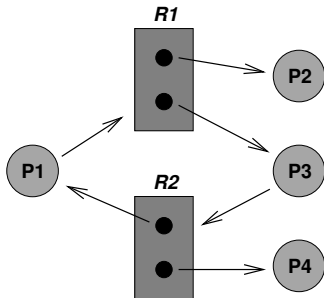
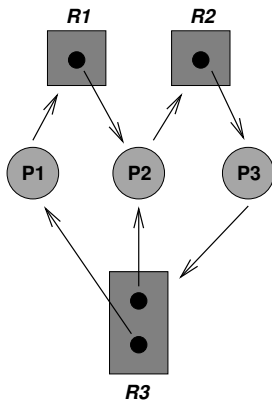
Systèmes
d'Exploitation

Didier Verna
EPITA

Théorie

Traitement

Prévention
Évitement
Détection
Correction





Traitement des interbloquages

Systèmes
d'Exploitation

Didier Verna
EPITA

Théorie

Traitement

Prévention

Évitement

Détection

Correction

■ Empêcher les interbloquages

- ▶ **Prévention** : s'assurer que l'une des conditions nécessaires n'est jamais remplie
- ▶ **Évitement** : disposer à l'avance d'informations sur l'utilisation des ressources par un processus, et décider dynamiquement de l'allocation

■ Corriger les interbloquages : requiert un algorithme de détection et un algorithme de correction

■ Ignorer les interbloquages : prétendre que ceux-ci n'arrivent jamais (UNIX, Windows)



Prévention des interbloquages

Systèmes
d'Exploitation

Didier Verna
EPITA

Théorie

Traitement

Prévention

Évitement

Détection

Correction

- **Exclusion mutuelle** : spooling.
Intrinsèquement impossible pour certaines ressources.
- **Occupation et attente** : forcer les processus à demander toutes leurs ressources avant l'exécution, ou bien une seule à la fois. Inefficace et risque de famine.
- **Pas de réquisition** : si un processus requiert une ressource occupée, toutes ses ressources sont implicitement libérées et le processus passe en attente. Pas utilisable pour certains types de ressources (imprimantes *etc.*).
- **Attente circulaire** : une seule ressource à la fois, ou imposer un ordre total sur l'allocation des ressources.



Évitement des interbloquages

S'assurer que le système reste dans un « état sain »

Systèmes
d'Exploitation

Didier Verna
EPITA

Théorie

Traitement

Prévention

Évitement

Détection

Correction

- Est-il possible de *prévoir* les interbloquages ?
⇒ Oui.
- De quelle information a t'on besoin ?
⇒ Du nombre maximum de requêtes sur une ressource pour chaque processus.



Trajectoires de ressources

Systèmes
d'Exploitation

Didier Verna
EPITA

Théorie

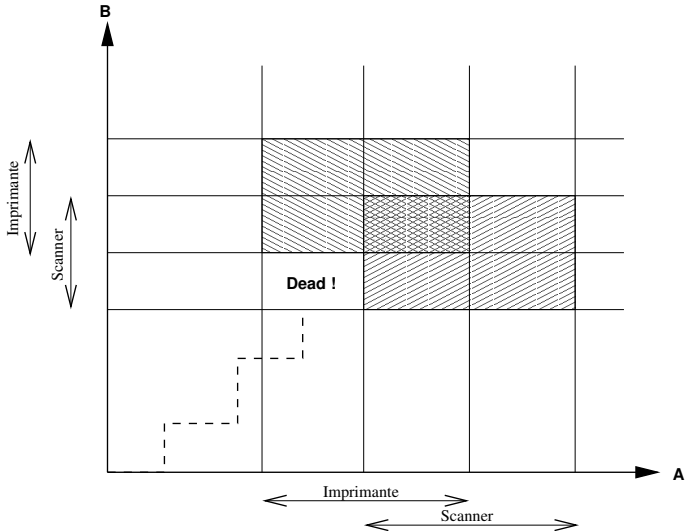
Traitement

Prévention

Évitement

Détection

Correction





États sains / malsains

Systèmes
d'Exploitation

Didier Verna
EPITA

Théorie

Traitement

Prévention

Évitement

Détection

Correction

Un état est « sain » si on peut trouver un ordonnancement garantissant la complétion des processus.

Proc / Used / Max :

A	3	9	3	9	3	9	3	9	3	9
B	2	4	4	4	0	-	0	-	0	-
C	2	7	2	7	2	7	7	7	0	-
∅	3		1		5		0		7	

A	3	9	4	9	4	9	4	9
B	2	4	2	4	4	4	0	-
C	2	7	2	7	2	7	2	7
∅	3		2		0		4	

Remarque : un état malsain ne conduit pas *forcément* à un interblocage (on ne sait pas conclure).



Algorithme du graphe d'allocation

Ssi chaque ressource n'existe qu'en une seule instance

Systèmes
d'Exploitation

Didier Verna
EPITA

Théorie

Traitement

Prévention

Évitement

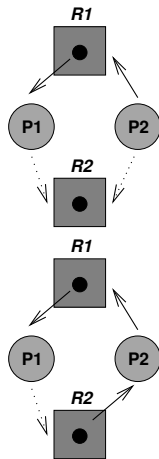
Détection

Correction

■ arc de réclamation

$(P_i \rightarrow R_j)$: indique que P_i demandera R_j dans le futur.

- Réclamation des ressources *a priori*.
- Une libération de ressource transforme un arc d'affectation en arc de réclamation.
- La transformation d'un arc de réclamation en arc d'affectation ne doit pas introduire de cycle (« état malsain »).





Algorithme du banquier

Dijkstra, 1965

Systèmes
d'Exploitation

Didier Verna
EPITA

Théorie

Traitement

Prévention

Évitement

Détection

Correction

■ Description

- ▶ Utilisable dans le cas général
- ▶ Réclamation de ressources *a priori*
- ▶ Complexité en $O(RP^2)$
- ▶ Composition en deux algorithmes : requête de ressource et détermination d'état sain

■ Structures de données nécessaires

- ▶ free [R] : vecteur de disponibilité des ressources
- ▶ used [P, R] : matrice d'occupation des ressources
- ▶ request [P, R] : matrice de requête de ressources



Algorithme de requête de ressources

Systèmes
d'Exploitation

Didier Verna
EPITA

Théorie

Traitement

Prévention

Évitement

Détection

Correction

- (1) **si** request $[P_i] \leq$ need $[P_i]$:
- (2) aller en 5.
- (3) **sinon:**
- (4) erreur (demande supérieure à la réclamation initiale).
- (5) **si** request $[P_i] \leq$ free:
- (6) aller en 9.
- (7) **sinon:**
- (8) bloquer P_i .
- (9) Faire tourner l'algorithme de détermination d'état avec :
- (10) free \leftarrow free - request $[P_i]$
- (11) used $[P_i] \leftarrow$ used $[P_i]$ + request $[P_i]$.
- (12) Si l'état calculé est sain, faire la mise à jour.



Algorithme de détermination d'état

Systèmes
d'Exploitation

Didier Verna
EPITA

Théorie

Traitement

Prévention

Évitement

Détection

Correction

- (1) Chercher P_i non marqué tel que $\text{request} [P_i] \leq \text{free}$.
- (2) Si P_i n'existe pas, aller en 6.
- (3) $\text{free} \leftarrow \text{free} + \text{used} [P_i]$.
- (4) Marquer P_i .
- (5) Aller en 1.
- (6) S'il reste un processus non marqué, alors le système est en état malsain.



Détection des interblocages

Systèmes
d'Exploitation

Didier Verna
EPITA

Théorie

Traitement

Prévention

Évitement

Détection

Correction

- Ni prévention, ni évitement
- Des interblocages peuvent donc se produire
- Besoin d'un algorithme de détection
- Besoin d'un algorithme de correction

⇒ Surcharge du système : maintenance de l'information nécessaire, exécution des algorithmes *etc.*



Algorithme du graphe d'attente

Ssi chaque ressource n'existe qu'en une seule instance

Systèmes
d'Exploitation

Didier Verna
EPITA

Théorie

Traitement

Prévention

Évitement

Détection

Correction

- Obtenu à partir du graphe d'allocation de ressources
- Suppression des noeuds de type ressource
- Rabattement des arcs : $P_i \rightarrow P_j$ indique que P_i attend une ressource occupée par P_j

⇒ Un interblocage se produit s'il existe un cycle dans le graphe d'attente. Un algorithme de détermination de cycle est en $O(P^2)$.



Structures de données nécessaires

- $free [R]$: vecteur de disponibilité des ressources
- $used [P, R]$: matrice d'occupation des ressources
- $request [P, R]$: matrice de requête de ressources

- (1) Chercher P_i non marqué tel que $request [P_i] \leq free$.
- (2) **si** P_i n'existe pas:
- (3) aller en 7.
- (4) $free \leftarrow free + used [P_i]$.
- (5) Marquer P_i .
- (6) Aller en 1.
- (7) **si** il reste un processus non marqué:
- (8) il est en situation d'interblocage.



Deux paramètres à prendre en compte

- Fréquence probable des interbloquages
- Nombre de processus impliqués

Déclenchement possible

- À chaque refus d'allocation d'une ressource (extrêmement coûteux)
- À intervalles réguliers
- Quand l'utilisation du processeur tombe en dessous d'une certaine limite
- *etc.*



Correction des interbloquages

Trois alternatives possibles

Systèmes
d'Exploitation

Didier Verna
EPITA

Théorie

Traitement

Prévention

Évitement

Détection

Correction

- **Intervention manuelle** : le système ne fait qu'informer l'opérateur de l'interbloquage
- **Réquisition de ressources** : saisie et redistribution arbitraire des ressources vers d'autres processus (risque de famine)
- **Terminaison de processus** : pas toujours facile (ex. processus en cours de modification de fichier, en cours d'impression *etc.*)



Réquisition de ressources

Trois problèmes à résoudre

Systèmes
d'Exploitation

Didier Verna
EPITA

Théorie

Traitement

Prévention

Évitement

Détection

Correction

- **Sélection d'une victime** : quelles ressources et quels processus doivent être réquisitionnés ?
- **Retour en arrière** : retour à un état sain (« checkpoints », coûteux pour le système) ou redémarrage complet (coûteux pour l'utilisateur)
- **Famine** : technique de vieillissement (ex. inclure le nombre de retours en arrière dans le coût)



■ Choix possibles

- ▶ **Terminaison totale** : coût élevé pour l'utilisateur (les processus avortés devront être relancés)
- ▶ **Terminaison partielle** : un par un jusqu'à ce que le cycle d'interblocage soit rompu. Coût élevé pour le système (appels multiples à l'algorithme de détection)
- ▶ **Terminaison externe** : tuer un processus non interbloqué (parce qu'il occupe des ressources intéressantes)

■ Critères de choix pour la terminaison

- ▶ priorité du processus
- ▶ âge du processus, temps restant d'exécution
- ▶ nombre et type de ressources utilisées
- ▶ *etc.*