III/ Les interblocages

1-Introduction

L'exécution d'un processus nécessite un ensemble de ressources (mémoire principale, disques, fichiers, périphériques, etc.) qui lui sont attribuées par le système d'exploitation. L'utilisation d'une ressource passe par les étapes suivantes :

– Demande de la ressource : Si l'on ne peut pas satisfaire la demande il faut attendre. La demande sera mise dans une table d'attente des ressources.

– Utilisation de la ressource : Le processus peut utiliser la ressource.

– Libération de la ressource : Le processus libère la ressource demandée et allouée.

Lorsqu'un processus demande un accès exclusif à une ressource déjà allouée à un autre processus, le système d'exploitation décide de le mettre en attente jusqu'à ce que la ressource demandée devienne disponible ou lui retourner un message indiquant que la ressource n'est pas disponible : réessayer plus tard.

Les ressources peuvent être de plusieurs types :

– Avec un ou multiples exemplaires : Existent-ils des multiples exemplaires d'une même ressource?

-preémptible ou non preémptible : Est-ce qu'on a le droit de retirer une ressource quand elle est utilisée par un processus?

La différence principale entre ressources préemptibles et non préemptibles est que les premières peuvent être retirées sans risque au processus qui les détient, tandis que les deuxièmes ne peuvent être retirées sans provoquer des problèmes. Comme exemples de ressources preémptibles nous avons le processeur. Les imprimantes et les scanners sont des exemples de ressources non preémptibles. Pour étudier le problème des interblocages, nous allons considérer uniquement les ressources non préemptibles.

2-Définition d'un interblocage

Des problèmes peuvent survenir, lorsque les processus obtiennent des accès exclusifs aux ressources. Par exemple, un processus P1 détient une ressource R1 et attend une autre ressource R2 qui est utilisée par un autre processus P2 ; le processus P2 détient la ressource R2 et attend la ressource R1. On a une situation d'interblocage (deadlock en anglais) car P1 attend P2 et P2 attend P1. Les deux processus vont attendre indéfiniment.

En général, un ensemble de processus est en interblocage si chaque processus attend la libération d'une ressource qui est allouée à un autre processus de l'ensemble. Comme tous les processus sont en attente, aucun ne pourra s'exécuter et donc libérer les ressources demandées par les autres. Ils attendront tous indéfiniment.

Exemples

Accès aux périphériques. Supposons que deux processus A et B veulent imprimer, en utilisant la même imprimante, un fichier stocké sur une bande magnétique. La taille de ce fichier est supérieure à la capacité du disque. Chaque processus a besoin d'un accès exclusif au dérouleur et à l'imprimante simultanément. On a une situation d'interblocage si :

– Le processus A utilise l'imprimante et demande l'accès au dérouleur.

– Le processus B détient le dérouleur de bande et demande l'imprimante.

Accès à une base de données. Supposons deux processus A et B qui demandent des accès exclusifs aux enregistrements d'une base de données. On arrive à une situation d'interblocage si :

- Le processus A a verrouillé l'enregistrement R1 et demande l'accès à l'enregistrement R2.
- Le processus B a verrouillé l'enregistrement R2 et demande l'accès à l'enregistrement R1.

Circulation routière. Considérons deux routes à double sens qui se croisent comme dans la **figure 1,** où la circulation est impossible. Un problème d'interblocage y est présent.

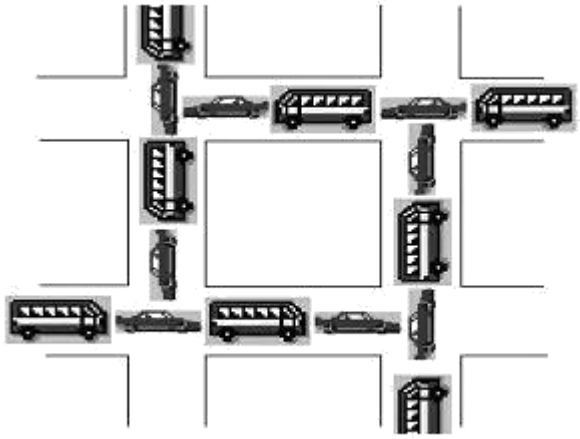


FIG.1 – Problème de circulation routière.

3-Conditions nécessaires pour l'interblocage

Pour qu'une situation d'interblocage ait lieu, les quatre conditions suivantes doivent être remplies (Conditions de Coffman) :

– **L'exclusion mutuelle**. A un instant précis, une ressource est allouée à un seul processus.
– **La détention et l'attente.** Les processus qui détiennent des ressources peuvent en demander d'autres.
– **Pas de préemption**. Les ressources allouées à un processus sont libérées uniquement par le processus.
– **L'attente circulaire**. Il existe une chaîne de deux ou plus processus de telle manière que chaque processus dans la chaîne requiert une ressource allouée au processus suivant dans la chaîne.

Par exemple, dans le problème de circulation de la figure 1 le trafic est impossible. On observe que les quatre conditions d'interblocage sont bien remplies :

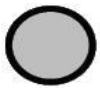
– *Exclusion mutuelle* : Seulement une voiture occupe un endroit particulier de la route à un instant donné.
– *Détention et attente* : Aucune voiture ne peut faire marche arrière.
– *Pas de préemption* : On ne permet pas à une voiture de pousser une autre voiture en dehors de la route.
– *Attente circulaire* : Chaque coin de la rue contient des voitures dont leur mouvement dépend des voitures qui bloquent la prochaine intersection.

4-Graphe d'allocation des ressources

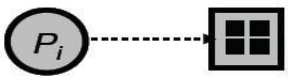
Le graphe d'allocation des ressources est un graphe biparti composé de deux types de nœuds et d'un ensemble d'arcs :

– Les processus qui sont représentés par des cercles.
– Les ressources qui sont représentées par des rectangles. Chaque rectangle contient autant de points qu'il y a d'exemplaires de la ressource représentée.
– Un arc orienté d'une ressource vers un processus signifie que la ressource est allouée au processus.
– Un arc orienté d'un processus vers une ressource signifie que le processus est bloqué en attente de la ressource.

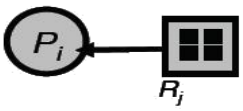
Ce graphe indique pour chaque processus les ressources qu'il détient ainsi que celles qu'il demande.

Processus  Ressource possédant 4 exemplaires (instances) 

Pi demande un exemplaire de Ri, bloqué en attente de la ressource



Pj détient un exemplaire de Rj ou Rj est affecté à Pj



Exemple **:** Soient trois processus A, B et C qui utilisent trois ressources R, S et T comme illustré à la figure 2 :

A	B	C
Demande R	Demande S	Demande T
Demande S	Demande T	Demande R
Libère R	Libère S	Libère T
Libère S	Libère T	Libère R

FIG. 2 – Besoins de trois processus.

Si les processus sont exécutés de façon séquentielle : A suivi de B suivi C, il n'y pas d'interblocage. Supposons maintenant que l'exécution des processus est gérée par un ordonnanceur du type circulaire. Si les instructions sont exécutées dans l'ordre :

1. A demande R
2. B demande S
3. C demande T
4. A demande S
5. B demande T
6. C demande R

On atteint une situation d'interblocage, fait qui est montré sur la figure 3.

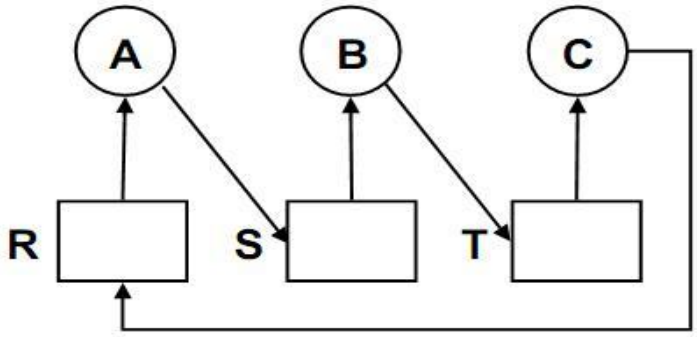


FIG. 3 – Situation d'interblocage de trois processus.

4.1 Réduction du graphe d'allocation des ressources

Un graphe réduit peut être utilisé pour déterminer s'il existe ou non un interblocage. Pour la réduction d'un graphe d'allocation des ressources, les flèches associées à chaque processus et à chaque ressource doivent être vérifiées.

- Si une ressource possède seulement des flèches qui sortent (il n'y a pas des requêtes), on les efface.
- Si un processus possède seulement des flèches qui pointent vers lui, on les efface.
- Si une ressource a des flèches qui sortent, mais pour chaque flèche de requête il y a une ressource disponible dans le bloc de ressources où la flèche pointe, il faut les effacer.

Exemple

Considérez quatre processus P1,P2….,P4 qui utilisent des ressources du type R1,R2 et R3. Le tableau suivant montre l'allocation courante et le nombre maximaux d'unités de ressources nécessaires pour l'exécution des processus. Le nombre de ressources disponibles est A= [0,0,0]. Le graphe d'allocation des ressources pour l'état courant est montré sur la figure 4.

Processus	R_1	R_2	R_3	R_1	R_2	R_3
P_1	3	0	0	0	0	0
P_2	1	1	0	1	0	0
P_3	0	2	0	1	0	1
P_4	1	0	1	0	2	0

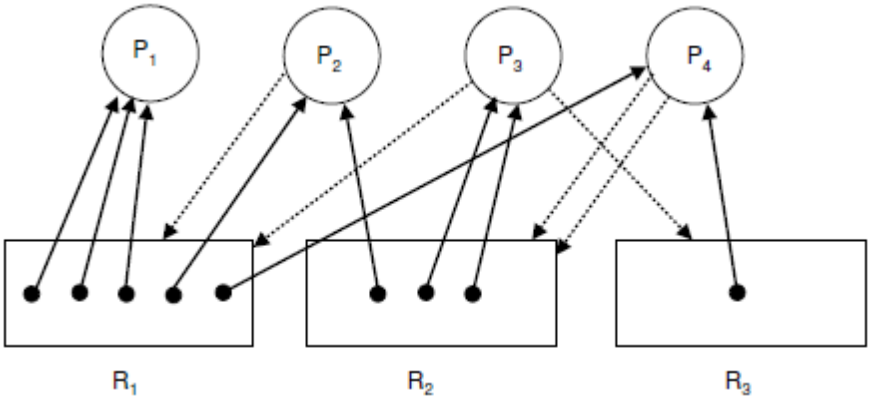


FIG. 4 - Graphe d'allocation des ressources

P1 possède seulement des flèches qui pointent vers lui, alors on les efface.
Les requêtes de R1 effectuées par P2 peuvent donc être allouées, et P2 aura seulement des flèches qui pointent vers lui. On les efface aussi. On efface la flèche de R1 vers P3 (qui a changé de direction) car R1 n'a que des flèches sortant. Le graphe réduit final est montré sur la figure 5.

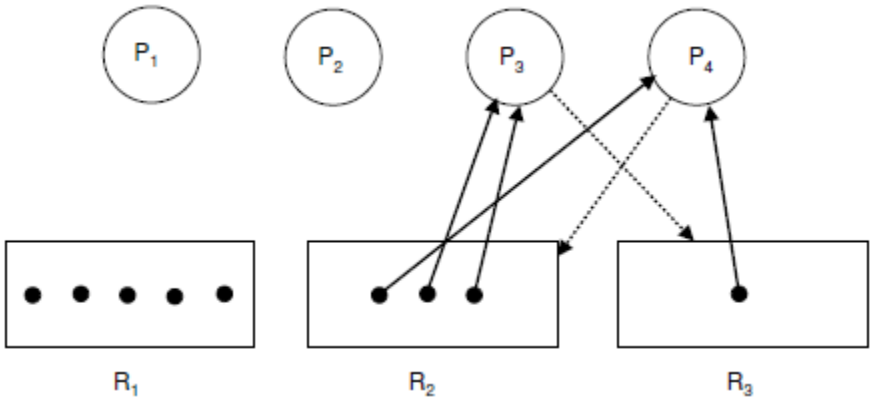


FIG. 5 - Graphe réduit d'allocation

5- Traitement des interblocages

Comme nous l'avons mentionné, les situations d'interblocage peuvent se produire dans un système. La question qui se pose est donc : doit-il prendre en compte ce problème ou l'ignorer?

Ignorer complètement les problèmes. On peut faire l'autruche et ignorer les possibilités d'interblocages. Cette « stratégie » est celle de la plupart des systèmes d'exploitation courants car le prix à payer pour les éviter est élevé.

Les détecter et y remédier. On tente de traiter les interblocages, en détectant les processus interbloqués et en les éliminant.

Les éviter. En allouant dynamiquement les ressources avec précaution. Le système d'exploitation peut suspendre le processus qui demande une allocation de ressource s'il constate que cette allocation peut conduire à un interblocage. Il lui attribuera la ressource lorsqu'il n'y aura plus de risque.

Les prévenir. En empêchant l'apparition de l'une des quatre conditions de leur existence.

6- La détection et la reprise

Dans ce cas, le système ne cherche pas à empêcher les interblocages. Il tente de les détecter et d'y remédier. Pour détecter les interblocages, il construit dynamiquement le graphe d'allocation des ressources du système qui indique les attributions et les demandes de ressources. Dans le cas des ressources à exemplaire unique, il existe un interblocage si le graphe contient au moins un cycle.
Dans le cas des ressources à exemplaires multiples, il existe un interblocage si le graphe contient au moins un cycle terminal (aucun arc ne permet de le quitter).
Le système vérifie s'il y a des interblocages :

- A chaque modification du graphe suite à une demande d'une ressource (coûteuse en termes de temps processeur).
- Périodiquement ou lorsque l'utilisation du processeur est inférieure à un certain seuil (la détection peut être tardive).

6.1 Algorithme de détection des interblocages

Soient n le nombre de processus P_1,P_2…. P_n et m le nombre de types de ressources E_1, E_2, ….E_m qui existent dans un système. L'algorithme de détection des interblocages utilise les matrices et vecteurs suivants :
– Matrice C des allocations courantes d'ordre (n*m). L'élément C[i,j] désigne le nombre de ressources de type Ej détenues par le processus .
– Matrice R des demandes de ressources d'ordre (n*m). L'élément R[i,j] est le nombre de ressources de type Ej qui manquent au processus pour pouvoir poursuivre son exécution.
– Vecteur A d'ordre m. L'élément A[j] est le nombre de ressources de type disponibles (non attribuées).
– Vecteur E d'ordre m. L'élément E[j] est le nombre total de ressources de type j existantes dans le système.

1. Rechercher un processus P_i non marqué dont la rangée R[i] est inférieure ou égale à A.
2. Si ce processus existe, ajouter la rangée C[i] à A, marquer le processus et revenir à l'étape 1.
3. Si ce processus n'existe pas, les processus non marqués sont en interblocage. L'algorithme se termine.

Exemple:.
Considérons un système où nous avons trois processus en cours et qui dispose de 4 types de ressources : 4 dérouleurs de bandes, 2 scanners, 3 imprimantes et un lecteur de CD ROM. Les ressources détenues et demandées par les processus sont indiquées par les matrices C et R.

$$E = [4,2,3,1]; A = [2,1,0,0]$$

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{bmatrix}; R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{bmatrix}$$

Seul le processus P3 peut obtenir les ressources dont il a besoin. Il s'exécute jusqu'à la fin puis libère ses ressources, ce qui donne : A= [2,2,2,0]. Ensuite P2 peut obtenir les ressources dont il a besoin. Il s'exécute jusqu'à la fin et libère ses ressources, ce qui donne : A= [4,2,2,1] . Enfin, P1 peut obtenir les ressources dont il a besoin. Il n'y a pas d'interblocage.

Exercice
Considérons un système ayant quatre processus, A, B. C et D, et trois types ressources R, S et T, à plusieurs exemplaires, avec 3R, 2S et 2T. L'attribution des ressources est la suivante :
– A détient une ressource de type R et demande une ressource de type S
- B détient 2 ressources de type S et demande une ressource de type R et une ressource de type T
- C détient 1 ressource de type R et demande une ressource de type S
- D détient 2 ressources de type T et demande une ressource de type R
Construire le graphe d'allocation des ressources. Y a-t-il un interblocage ? Si oui, quels sont les processus concernés?

7-La reprise des interblocages

Lorsque le système détecte un interblocage, il doit le supprimer, ce qui se traduit généralement par la réalisation de l'une des opérations suivantes :

- Retirer temporairement une ressource à un processus pour l'attribuer à un autre.
- Restaurer un état antérieur (retour en arrière) et éviter de retomber dans la même situation.
- Supprimer un ou plusieurs processus.

8-L'évitement des interblocages (allocation des ressources avec précaution)

a) Etat Sain

Un état est dit sain, si le système peut allouer les ressources pour chaque tâche dans un ordre quelconque tout en évitant un interblocage ; formellement, un système est dans un état sain, s'il existe une suite saine. Une suite de tâches <P1, P2, …,Pn> est une suite saine pour l'état courant d'allocation si, pour chaque Pi, les requêtes que Pi peut faire peuvent être satisfaites avec les ressources disponibles plus les ressources détenues par toutes les tâches Pj, avec j < i.

Dans cette situation, si les ressources nécessaires pour Pi ne sont pas immédiatement disponibles alors Pi doit attendre que tous les Pj aient terminé ; dès lors, Pi peut obtenir toutes ses ressources, elle achève son exécution, libère les ressources allouées et termine. Quand Pi termine, Pi+1 peut obtenir les ressources nécessaires et ainsi de suite. Si une telle suite n'existe pas alors le système est dit non sain.

b) Algorithme du banquier

Comment déterminer si un état est sûr (sain) ou non sûr ? Djikstra a proposé en 1965 un algorithme d'ordonnancement, appelé l'Algorithme du banquier qui permet de répondre à cette question.
Son nom provient de l'analogie avec le comportement d'un banquier qui doit garder suffisamment de ressources pour pouvoir satisfaire l'ensemble des requêtes de ses clients, quitte à refuser certaines demandes qui ne lui permettraient pas par la suite de satisfaire les autres.

Lorsqu'une tâche entre dans le système, elle déclare le nombre maximum d'instances qu'elle peut être amenée à demander. Ce nombre évidemment ne peut dépasser le nombre de ressources disponibles dans le système.

Lorsqu'un utilisateur demande un ensemble de ressources, le système doit déterminer si cette allocation laisse le système dans un état sain ; si oui, les ressources sont allouées, si non, la tâche est mise en attente tant que toutes les ressources nécessaires ne sont pas disponibles (libérées par les autres tâches).

Cet algorithme nécessite les structures de données suivantes :

- Ressource[r] nombre total de ressources de la classe r
- Annonce[p,r] nombre maximal de ressources de la classe r nécessaires au processus p
- Alloc[p,r] nombre de ressources de la classe r couramment allouées au processus p
- Dispo[r] nombre total de ressources de la classe r couramment disponibles

$$Dispo[r] = Ressource[r] - \sum_{p=0}^{p=N-1} Alloc\,[p,r]$$

- **Suite Fiable**

Soit E0 un état réalisable, Cherchons un processus *pa* qui, s'il était exécuté seul à partir de l'état E0, en utilisant la totalité de son annonce, pourrait aller jusqu'à son terme et donc libérer toutes les ressources qu'il possède.

- Un tel processus doit donc vérifier :

$$Annonce\,[a,*] - Alloc[a,*] <= Dispo$$

```
Fonction nouvelEtatFiable: booleen;

DispCourant: Tableau [Ressource] de Entier;
Reste : ListedeProcessus;
Trouvé, Possible : booleen;

Début
DispCourant := Dispo;
Reste := [0..N-1]; // tous les processus
Possible := vrai;
Tant que Possible=vrai alors
    Chercher(p) dans Reste tel que

    Annonce[p,*]- Alloc[p,*] <= DispCourant Si

    Trouvé Alors

        DispCourant +:= Alloc[p,*];

        Reste -:= [p]

    FinSi
    Sinon Possible := false

FinTantque

nouvelEtatFiable :=(Reste=[]);

Fin
```

Exemple :

A un instant t donné, l'état E1 du système peut être le suivant :

	R0	R1	R3
P0	7	5	3
P1	3	2	2
P2	9	0	2
P3	2	2	2
P4	4	3	3

Annonce

	R0	R1	R3
P0	0	1	0
P1	2	0	0
P2	3	0	2
P3	2	1	1
P4	0	0	2

Alloc

R0	R1	R2
10	5	7

Ressources

R0	R1	R2
3	3	2

Dispo

Cet état est-il un état fiable ?

Solution :

On peut calculer la matrice : Annonce – Alloc

	R0	R1	R2
P0	7	4	3
P1	1	2	2
P2	6	0	0
P3	0	1	1
P4	4	3	1

On peut déduire la suite fiable possible : (p1, p3, p4, p2, p0)

9-La prévention des interblocages

Pour prévenir les interblocages, on doit éliminer une des quatre conditions nécessaires à leur apparition.

- **La condition d'exclusion mutuelle** : le fait de ne pas réserver exclusivement les ressources permet d'éliminer cette condition. Par exemple, pour les imprimantes, les processus « spoolent » leurs travaux dans un répertoire spécialisé et un démon d'impression les traitera, en série, l'un après l'autre.
- **La condition de détention et d'attente** : on peut exiger que tous les processus commandent toutes leurs ressources avant de commencer à les utiliser. Cette solution, en plus de mal utiliser les ressources exigent de connaître par avance les ressources nécessaires. En tel cas, un algorithme du banquier ferait l'affaire.

- **Pas de préemption**. La troisième condition n'est pas raisonnablement traitable pour la plupart des ressources sans dégrader profondément le fonctionnement du système. On peut cependant l'envisager pour certaines ressources dont le contexte peut être sauvegardé et restauré.

- **L'attente circulaire**. Enfin, on peut résoudre le problème de l'attente circulaire en numérotant les ressources et en n'autorisant leur demande, par un processus, que lorsqu'elles correspondent à des numéros croissants ou en accordant aux processus une seule ressource à la fois (s'il a besoin d'une autre ressource, il doit libérer la première). Par exemple :
  - F(CD-ROM)=1
  - F(imprimante)=2
  - F(plotter)=3
  - F(rubban)=4

Ainsi on garantit qu'il n'aura pas de cycles dans le graphe des ressources. On peut exiger seulement que aucun processus ne demande une ressource dont le numéro est inférieur aux ressources déjà allouées. Mais ceci n'est pas non plus la panacée, car, en général, le nombre potentiel de ressources est si important qu'il est très difficile de trouver la bonne fonction F pour les numéroter.