Systèmes d'Exploitations Avancés

Chapitre V :

Ressources et interblocage

Amine DHRAIEF

Mastère professionnel en Modélisation, Bases de Données et Intégration des Systèmes

ESEN, Univ. Manouba

Introduction

- **L'existence d'un groupe de processus bloqués en attente de ressources, chacune d'elles étant allouée à l'un des processus du groupe**

Ressources

- **Pour s'exécuter, un processus a besoin d'un certain nombre d'éléments : des éléments matériels comme de la mémoire, un processeur, des périphériques (clavier et écran par exemple) et des éléments logiciels, fichiers, tables, etc.**
  - **Tous ces éléments matériels et logiciels sont appelés des ressources.**

- **On distingue des ressources à accès partagé (fichier en lecture seule) et des ressources à accès exclusif (par exemple une imprimante).**
  - **Avant d'utiliser une ressource, un processus doit la demander, explicitement ou implicitement.**
  - **Il doit la libérer après utilisation.**

Ressources

- **Les opérations de demande et de restitution d'une ressource sont appelées l'allocation et la libération.**
  - **Quand un processus demande l'allocation d'une ressource à accès exclusif déjà attribuée à un second processus, le système le bloque jusqu'à ce que la demande puisse être satisfaite, c'est à dire jusqu'à la libération de la ressource par le second processus.**

- **Une ressource est susceptible de réquisition (« préemption ») si le système peut la retirer au processus auquel elle était allouée pour l'affecter à un autre processus.**

- **L'unité centrale peut ainsi être réquisitionnée à faible coût ; il est plus compliqué, voire impossible, de réquisitionner un traceur ou une imprimante**

Ressources

- **Les ressources peuvent être de plusieurs types :**
  - **Reutilisables ou disponibles : Existent-elles après son utilisation ?**
  - **Reutilisables** :
    - Toutes les ressources physiques.
    - Quelques unes logiques (fichiers, mutex, verrous, etc.).
  - **Disponibles** :
    - Un processus engendre une ressource et un autre l'utilise.
    - Ressources associées à la communication et à la synchronisation comme les messages, les signaux, les sémaphores, etc.

Ressources

- **D'usage partagé :**
  - Est-ce que la ressource peut être utilisée par plusieurs processus en même temps ? Les ressources partagées n'affectent pas les interblocages.

- **Avec un ou multiples exemplaires :**
  - Existent-ils de 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 ?

Ressources

- **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**
  - **où le changement de contexte correspond à l'expropriation et l'état du processeur est copié à la Table de Contrôle de Processus et la mémoire virtuelle le remplacement est une expropriation et le contenu de la page doit être copié au swap**

- **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**

Exemples introductifs

- **L'interblocage n'est pas réduit à l'informatique. On peut en donner des exemples issus de la vie courante.**

- **Considérons par exemple un carrefour à quatre voies avec la règle de priorité à droite habituelle. Lorsque deux ou trois voitures abordent simultanément le carrefour, la règle permet aux voitures de passer l'une après l'autre.**

- **En revanche, si quatre voitures se présentent simultanément, le respect de la règle de priorité impose à chaque voiture d'attendre que celle de droite soit passé : le blocage est définitif.**

- **Dans ce cas, et en l'absence d'intervention extérieure (agent de police, fonte de la neige), le système est définitivement bloqué.**

Définition d'un interblocage

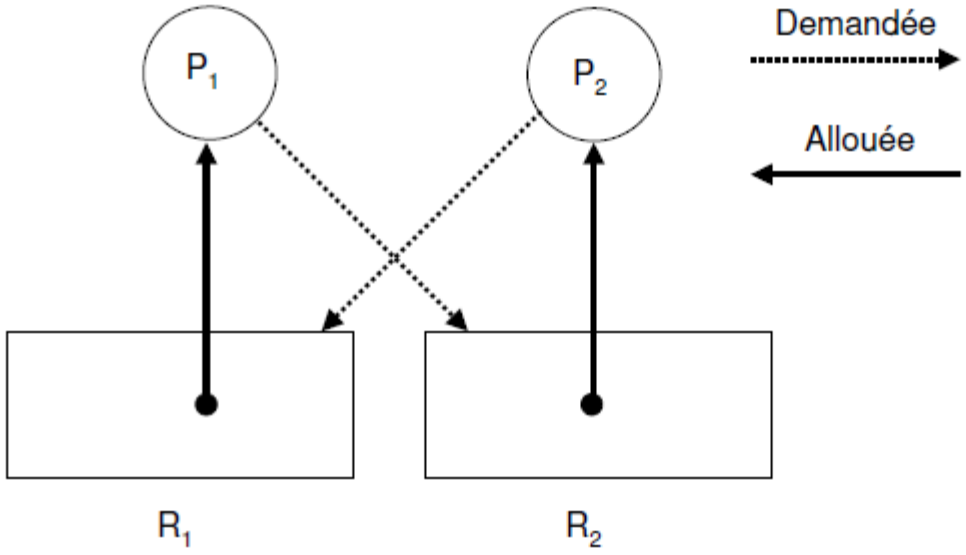
- **Des problèmes peuvent survenir, lorsque les processus obtiennent des accès exclusifs aux ressources.**

- **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**

Définition d'un interblocage



Définition d'un interblocage

- **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.**

Exemples

- **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**

Conditions nécessaires pour l'interblocage

- **Pour qu'une situation d'interblocage ait lieu, les quatre conditions suivantes doivent être remplies (Conditions de Coffman) :**
  I. **L'exclusion mutuelle: A un instant précis, une ressource est allouée à un seul processus.**
  II. **La détention et l'attente : Les processus qui détiennent des ressources peuvent en demander d'autres.**
  III. **Pas de préemption : Les ressources allouées à un processus sont libérées uniquement par le processus.**
  IV. **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.**

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.**

- **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.

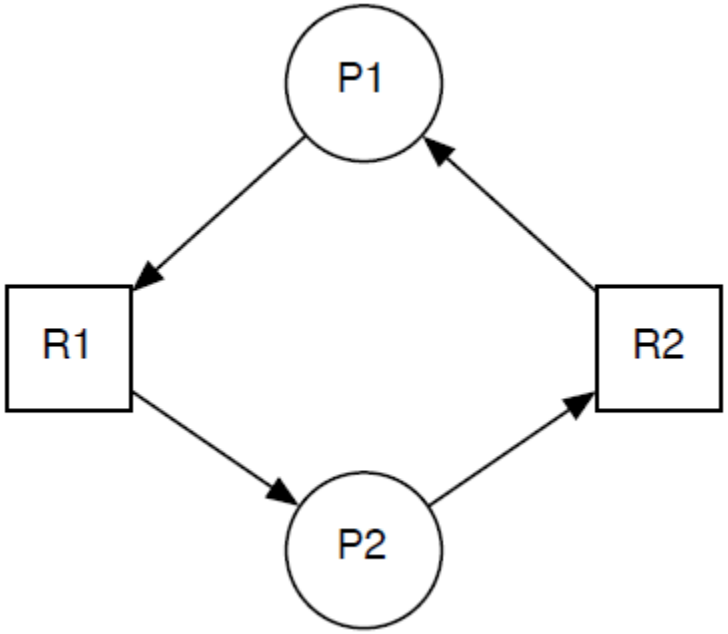
Graphe d'allocation des ressources

- **L'idée est d'associer à un interblocage un cycle dans un graphe. Les nœuds du graphe correspondent aux processus (dénotés par des cercles sur nos exemples) et aux ressources (dénotées par des carrés).**

- **Quand un processus possède une ressource, un arc relie le nœud processus au nœud ressource ; lorsqu'un processus demande une ressource, un arc relie la ressource au processus.**

- **Le graphe évolue donc à chaque demande, allocation ou libération de ressource**

Graphe d'allocation des ressources



Exemple

- Soient trois processus **A, B et C qui utilisent trois ressources R, S et T**

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

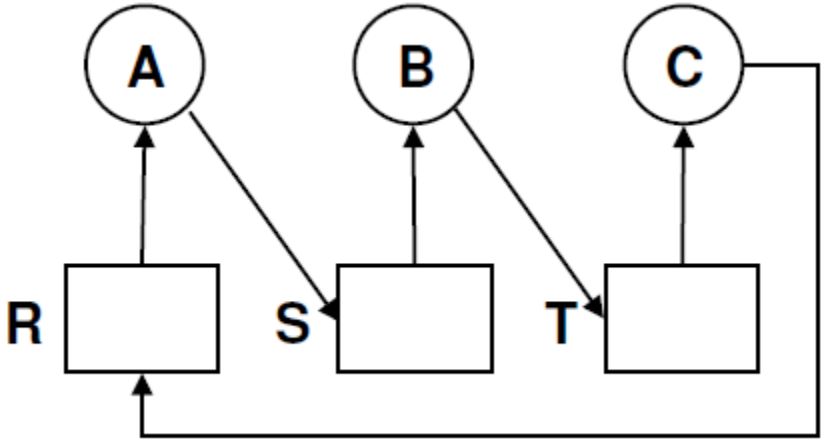
- Si les processus sont exécutés de façon séquentielle : **A suivi de B suivi C, il n'y pas d'interblocage.**

Exemple

- 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**

- **Montrer qu'on atteint la situation d'interblocage**

Exemple



Réduction du graphe d'allocation des ressources

- **Un graphe réduit peut-être utilisé pour déterminer s'il existe ou non un interblocage.**
  - **Si le graphe peut être réduit pour tous les processus, il n'y a pas d'interblocage ;**
  - **Si le graphe ne peut être réduit pour tous les processus, alors les processus irréductible constituent l'ensemble des processus interbloqués**

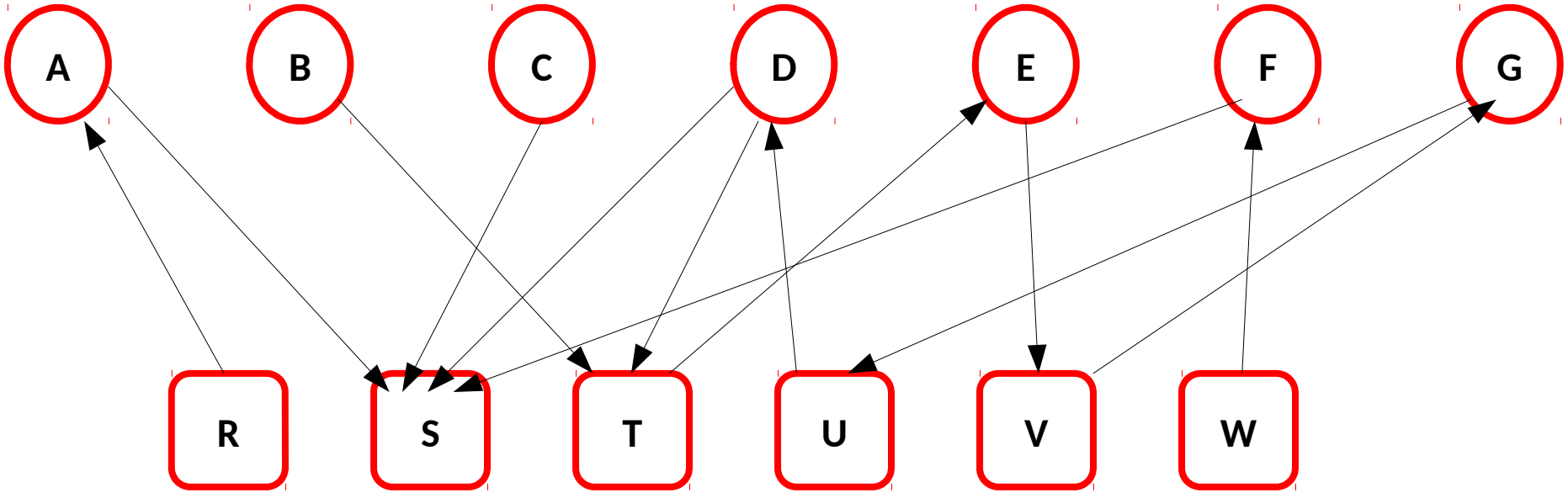
- **Pour la réduction d'un graphe d'allocation des ressources, les flèches associées à chaque processus et à chaque ressource doivent être vérifiées.**
  i. **Si une ressource possède seulement des flèches qui sortent (il n'y a pas des requêtes), on les efface.**
  ii. **Si un processus possède seulement des flèches qui pointent vers lui, on les efface.**
  iii. **Si les demandes en ressources d'un processus peuvent être servies, on réduit le graphe en enlevant les arcs de/vers ce processus ;**

Réduction du graphe d'allocation des ressources

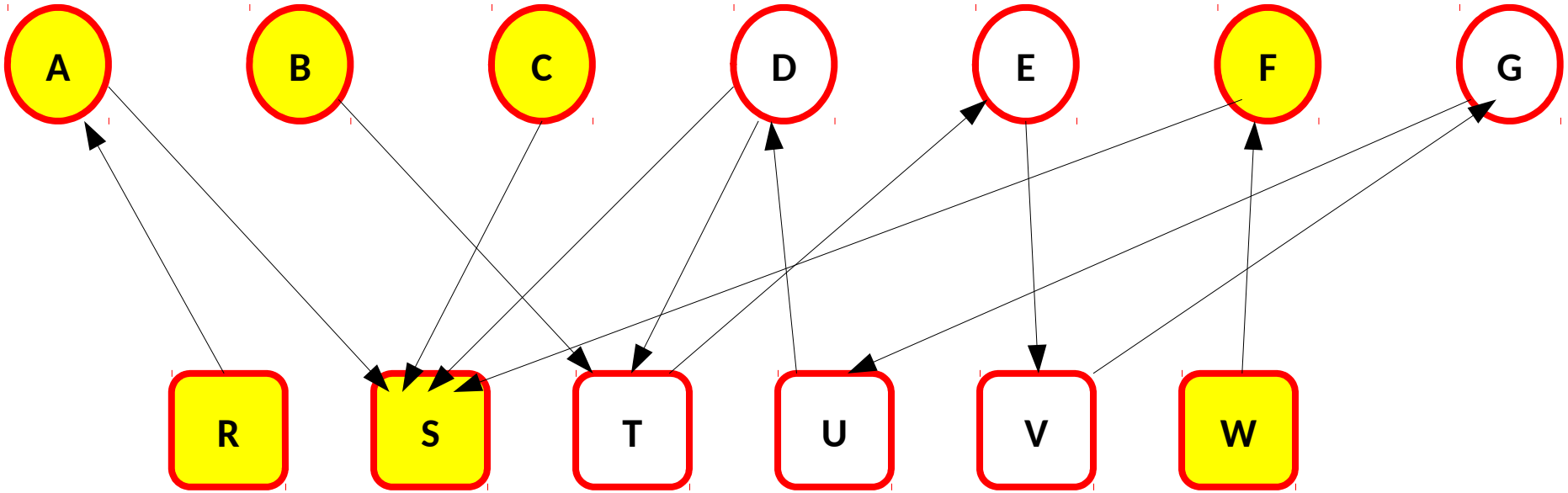
- **Considérons un système ayant sept processus, A à G, et six ressources R à W. L'attribution des ressources est la suivante :**
  - A détient R et demande S ;
  - B demande T ;
  - C demande S ;
  - D détient U et demande S et T ;
  - E détient T et demande V;
  - F détient W et demande S ;
  - G détient V et demande U.

- **Construire le graphe d'allocation des ressources ?**
- **Y a-t-il un interblocage?**
- **Si oui, quels sont les processus concernés ?**

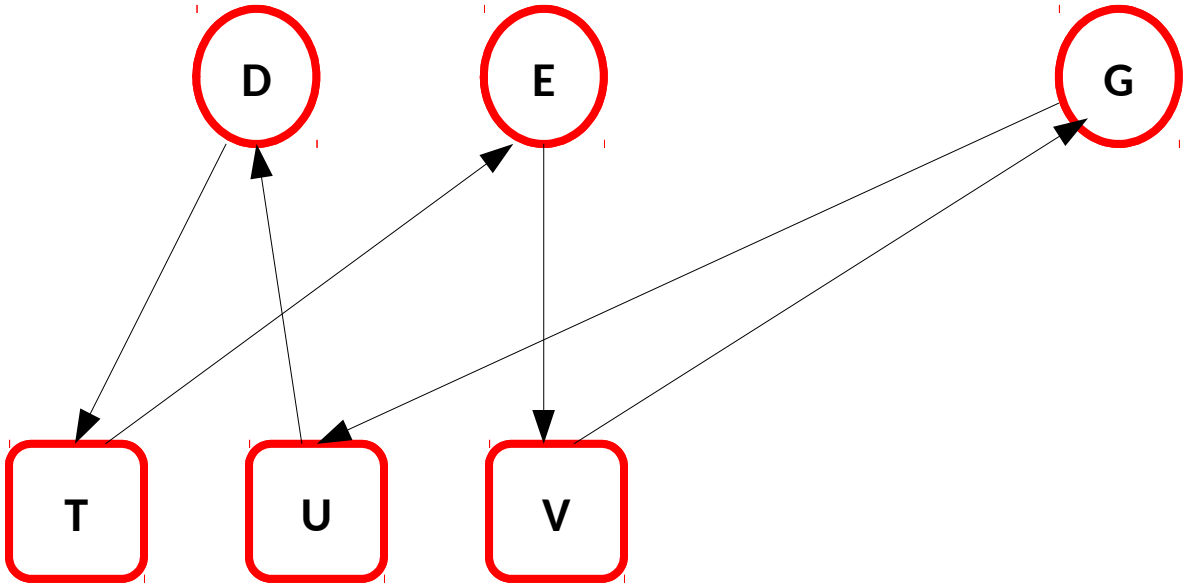
le graphe d'allocation des ressources



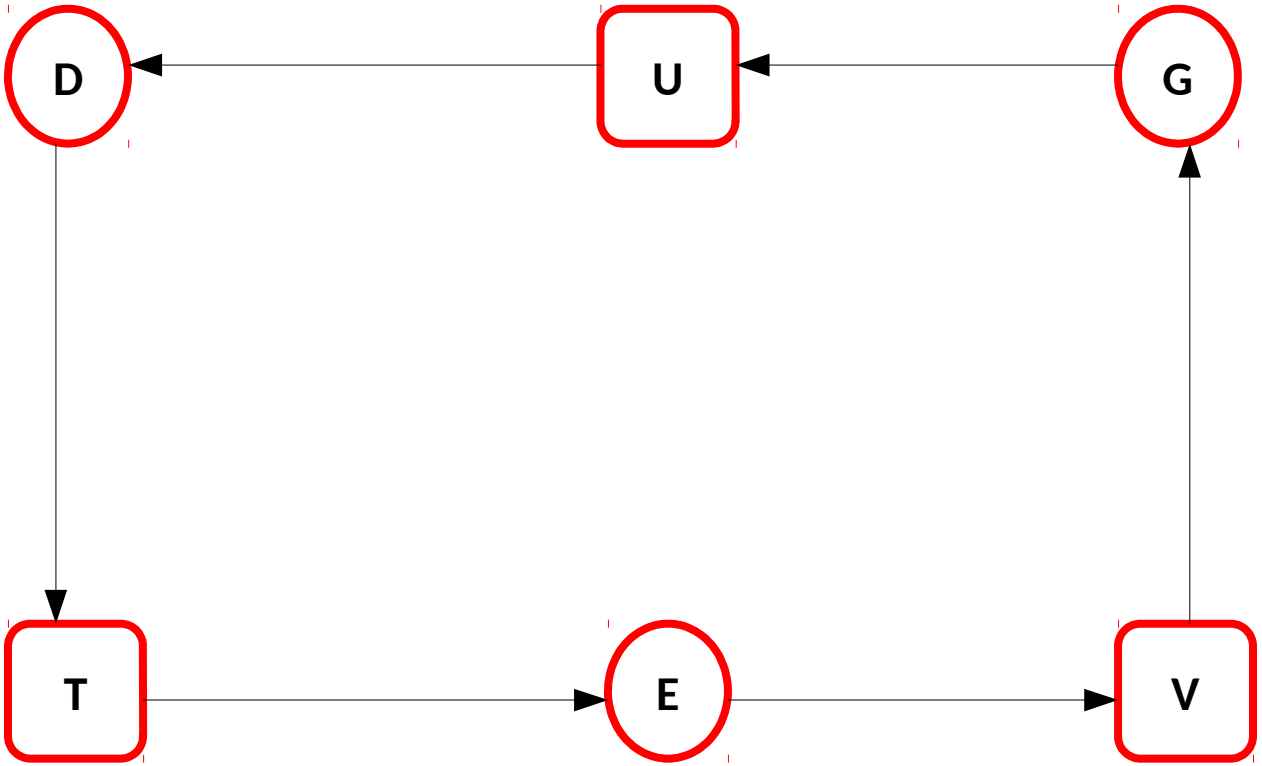
le graphe d'allocation des ressources



Le cycle



Le cycle



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

Traitement des interblocages

- **Les détecter et y remédier:** On tente de traiter les interblocages, en détectant les processus inter-bloqués 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.

LA DÉTECTION ET LA REPRISE

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.**

La détection et la reprise

- **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).**

Algorithme de détection des interblocages

- **Soient (n) le nombre de processus P[1], P[2],…,P[n] et (m) le nombre de ressources E[1], E[2], …., E[m] qui existent dans un système.**

- **L'algorithme de détection des interblocage utilise les matrices et vecteurs suivants:**
  - **Matrice C des allocation courante d'ordre (n*m): L'élément C[i,j] désigne le nombre de ressource de type E[j] détenu par le processus P[i].**
  - **Matrice R des demandes de ressources d'ordre (n*m): L'élément R[i,j] est le nombre des ressources de type E[j] demandées par le processus P[i] pour pouvoir poursuivre son exécution**

Algorithme de détection des interblocages

- **<u>Vecteur A d'ordre m:</u> L'élément A[j] est le nombre de ressources de type E[j] disponibles (non attribuées).**

- **<u>Vecteur E d'ordre m:</u> 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 inférieur 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**

- **On peut démontrer mathématiquement que s'il existe au moins une séquence sûre, alors il existe un nombre infini de séquences sûres. Dans les états sûrs, le système d'exploitation possède le contrôle sur les processus. Dans un état non sûr le contrôle dépends du comportement des processus.**

Exemple 1

- **Considérons un système où nous avons 3 (P1,P2,P3) processus en cours et qui dispose de 4 types de ressources : 4 dérouleurs de bandes, 2 scanners, 3 imprimantes et 1 lecteur de CD ROM.**

- **Les ressources détenues et demandées par les processus sont indiquées par les matrices C et R.**

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

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

Exemple 1

- **Nombre total de ressources : E = [4,2,3,1]**
- **Ressources disponibles A = [2,1,0,0]**
- **Ressources détenus par (P3) C3 = [0,1,2,0]**
- **Ressource manquantes à (P3) R3 = [2,1,0,0]**

→ P3 peut s'exécuter (comparer R3 à A)

→ P3 libère ses ressources (C3) → A [2,2,2,0]

Exemple 1

- **Nombre total de ressources : E = [4,2,3,1]**
- **Ressources disponibles A = [2,2,2,0]**
- **Ressources détenus par (P2) C2 = [2,0,0,1]**
- **Ressource manquantes à (P2) R2 = [1,0,1,0]**

→ P2 peut s'exécuter (comparer R2 à A)

→ P2 libère ses ressources (C2) → A [4,2,2,1]

Exemple 1

- **Nombre total de ressources : E = [4,2,3,1]**
- **Ressources disponibles A [4,2,2,1]**
- **Ressources détenus par (P1) C1 = [0,0,1,0]**
- **Ressource manquantes à (P1) R1 = [2,0,0,1]**

→ P1 peut s'exécuter (comparer R1 à A)

→ P1 libère ses ressources (C1) → A [4,2,3,1]

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 :**

1) **Retirer temporairement une ressource à un processus pour l'attribuer à un autre.**

2) **Restaurer un état antérieur (retour en arrière) et éviter de retomber dans la même situation.**

3) **Supprimer un ou plusieurs processus.**

L'ÉVITEMENT DES INTERBLOCAGES

L'évitement des interblocages

- **Lorsqu'un processus demande une ressource, le système doit déterminer si l'attribution de la ressource est sûre.**
  - **Si c'est le cas, il lui attribue la ressource.**
  - **Sinon, la ressource n'est pas accordée.**

- **Un état est sûr si tous les processus peuvent terminer leur exécution (il existe une séquence d'allocations de ressources qui permet à tous les processus de se terminer).**

Les états sûrs et non sûrs

- **Exemple:**
  - une ressource partagée (10 instances)
  - Trois processus A, B, C
  - A possède 3 instances, au finale peut avoir besoin de 9
  - B possède 2 instances, pourra avoir besoin ultérieurement de 4
  - C possède 2, pourra avoir besoin de 5 supplémentaires

Les états sûrs et non sûrs

A	3	9
B	2	4
C	2	7
Libres= 3		

A	3	9
B	4	4
C	2	7
Libres= 1		

A	3	9
B	0	-
C	2	7
Libres= 5		

A	3	9
B	0	-
C	7	7
Libres= 0		

A	3	9
B	0	-
C	0	-
Libres= 7		

A	9	9
B	0	-
C	0	-
Libres= 1		

L'état est sûr

Les états sûrs et non sûrs

A	3	9
B	2	4
C	2	7
Libres= 3		

(1) A demande et obtient une autre ressource

A	4	9
B	2	4
C	2	7
Libres= 2		

A	4	9
B	4	4
C	2	7
Libres= 0		

(2) B s'exécute

A	4	9
B	-	-
C	2	7
Libres= 4		

(3) Blocage

L'état est non sûr

La demande de A n'aurait pas dû être satisfaite

Les états sûrs et non sûrs

- **La différence entre un état sûr et état non sûr**
  - **Dans un état sûr le système peu garantir que tout les processus s'achèveront**
  - **Dans un état non sûr, cette garantie ne peut pas être « garantie »**

L'algorithme du banquier pour une ressource unique

- **En 1965, Dijkstra a proposé un algorithme d'ordonnancement qui permet d'éviter les interblocages, et appelé algorithme du banquier.**

- **Il s'inspire de la manière dont un banquier accorde des crédits à un groupe de clients.**

- **L'algorithme vérifie si le fait d'accorder cette requête conduit à un état non sûr.**
  - **Si tel est le cas la requête est refusée**
  - **Dans le cas contraire, elle est accordée**

L'algorithme du banquier pour une ressource unique

- **L'idée de l'algorithme du banquier repose sur un contrat passé entre chaque processus et le système.**

- **Chaque processus doit annoncer au départ le nombre maximum de ressources dont il aura besoin pendant son exécution.**

- **Il s'engage alors à ne pas dépasser cette demande maximum ainsi qu'à libérer au bout d'un temps fini toutes les ressources qui lui ont été allouées.**

- **De son côté, le système s'engage à satisfaire au bout d'un temps fini toute demande valide (restant dans les limites de l'annonce) effectuée par un processus.**

L'algorithme du banquier pour une ressource unique

- **Quatre client A,B,C,D ont reçu chacun un crédit d'un certain montant en unité**
- **Le banquier sait que les clients n'auront pas tous besoin immédiatement de leur crédit maximum**
- **Il leur a donc réserve seulement 10 unit au lieu de 22**

L'algorithme du banquier pour une ressource unique

A	0	6
B	0	5
C	0	4
D	0	7
Libres= 10		

(a) Init

A	1	6
B	1	5
C	2	4
D	4	7
Libres= 2		

(b) Sûr

A	1	6
B	2	5
C	2	4
D	4	7
Libres= 1		

(c) Non sûr

- (b) est sûr car: C satisfait → C libère 4 ressources → B → D → A
- (c) non sûr: B demande un ressource → aucun client n'est satisfait

L'algorithme du banquier pour une ressource unique

- **L'algorithme du banquier consiste à examiner chaque nouvelle requête pour voir si elle conduit à un état sûr**

- **Le cas échéant la ressource est alloué, sinon elle est mise en attente**

- **Afin de voir si un état est sûr,**
  - **le banquier vérifie s'il possède suffisamment de ressource pour satisfaire un client**
  - **Si tel est le cas, on suppose que ces crédit seront remboursé, on examine le cas du client le plus proche de la limite et ainsi de suite.**
  - **Si tous les crédit sont finalement remboursés, l'état est sûr et la requête initiale peut être raccordée.**

L'algorithme du banquier pour plusieurs ressources

	Ressources attribuées C			
	R1	R2	R3	R4
A	3	0	1	1
B	0	1	0	0
C	1	1	1	0
D	1	1	0	1
E	0	0	0	0
Total	5	3	2	2

	Ressources demandées R			
	R1	R2	R3	R4
A	1	1	0	0
B	1	1	0	0
C	3	1	0	0
D	0	0	1	0
E	2	1	1	0
Total	6	4	3	2

Ressources existantes : E = (6 4 3 2)
Ressources détenus P = (5 3 2 2)
ressources disponibles dispo A = E – P = (1 1 1 0)

L'algorithme du banquier pour plusieurs ressources

- **5 processus sont actifs (A, B, C, D, E) et il existe 4 catégories de périphériques (R1,R2,R3,R4)**

1. **Rechercher une rangée R dont les demandes en ressources non satisfaites son inférieur ou égale à A.**

2. **Supposer que le processus associé à C obtient les ressources et se termine. Supprimer sa ligne et actualiser A.**

3. **Répéter 1 et 2 jusqu'à ce que tous les processus soient terminés (l'état initial était donc sûr) ou jusqu'à un interblocage (l'état initial n'était pas sûr)**

PRÉVENTION DE INTERBLOCAGES

Approches de prévention des interblocages

Condition	Approche
Exclusion mutuelle	Tout traiter en différer (spoule)
Détention et attente	Demander toutes les ressources dès le départ
Non-préemption	Retier les ressource
Attente circulaire	Ordonner les ressources numériquement

S'attaquer à la condition de l'exclusion mutuelle

- **Si une ressource n'est jamais attribué de manière exclusive, nous n'aurons jamais d'interblocage**

- **Certains périphériques (tel que l'imprimante) peuvent être spoolés (traités en différé)**
  - seul le démon d'imprimante peut directement utiliser l'imprimante
  - cela élimine les interblocages

- **Tous les périphériques ne peuvent être spoulés.**

- **Principe:**
  - **éviter d'attribuer une ressource lorsque cela n'est pas absolument nécessaire**
  - **le plus petit nombre possible de processus peuvent réclamer la ressource**

S'attaquer à la condition de détention et d'attente

- **Si nous pouvons empêcher que des processus qui détiennent des ressources attendent d'autre ressources, nous pouvons éviter l'interblocage.**
  - **Exiger de tous les processus qu'ils demandent des ressources dont ils ont besoin avant de commencer à s'exécuter.**
  - **Si elles sont disponibles, le processus se verra attribuer tout ce dont il a besoin et pourra s'exécuter jusqu'au bout.**
  - **Si une ou plusieurs ressources sont indisponibles, rien ne pourra être attribué et le processus se contentera d'attendre.**

- **Plusieurs processus ne savent pas de combien de ressources ils vont avoir besoin avant de commencer de s'exécuter.**
  - **S'ils le savent → algorithme du banquier**

S'attaquer à la condition de non-préemption

- **Cette option est difficilement réalisable**

- **Considérer un processus utilisant une imprimante**
  - **réquisitionner l'imprimante au milieu de la tâche !!??**
  - **Solution dans ce cas: utiliser**

    le disque et le démon d'impression

S'attaquer à la condition de l'attente circulaire

- Établir une règle qui stipule qu’un processus est à tout moment attribué à une seule ressource

- S’il a besoin d’une seconde ressource, il doit libérer la première

- Irréaliste

S'attaquer à la condition de l'attente circulaire

- **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**

- **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

La privation de resources (starvation, famine)

- Le terme de famine (« starvation ») est issu de l'exemple classique des philosophes aux spaghetti dans lequel deux philosophes qui mangent en alternance peuvent empêcher de manger le philosophe qui se trouve entre eux

- Prévenir la famine consiste à garantir que toute demande de ressource sera satisfaite au bout d'un temps fini.

- Les méthodes classiques de prévention de la famine consistent d'une part à augmenter la priorité des demandes en fonction du temps pendant laquelle une ressource a été attendue et, d'autre part, à introduire un ordre total entre les demandes de ressources et à utiliser cet ordre dans les algorithmes d'allocation

EXERCICES

Exercice I

- Pour **traiter une image de taille T unités**, un système **composé de N processeurs** à mémoire partagée, **crée N processus**. Chaque processus s'exécute sur un processeur et se charge de traiter une partie de l'image. Pour faire son traitement, un processus a besoin d'une unité de mémoire par unité d'image qu'il a à traiter, mais demande de la mémoire unité par unité au fur et à mesure de ses besoins. Si une demande ne peut être satisfaite, le processus demandeur est mis en attente (attente active). La libération des unités de mémoire allouées à un processus aura lieu lorsqu'il aura fini le traitement de toute sa partie. **La mémoire disponible pour répondre aux demandes d'allocation des différents processus est de taille M unités**.

Exercice I

- **Question 1: Algorithme du banquier**

- Pour éviter les interblocages, le système utilise l'algorithme du banquier. Supposez que N =4, T = 16, M = 8 et qu'à l'état courant les processus P1, P2, P3 et P4 ont respectivement traité 1 sur 3 unités, 1 sur 4 unités, 1 sur 4 unités et 3 sur 5 unités (par exemple, **P1 : 1 sur 3 unités** signifie que le processus P1 est chargé de traiter 3 unités de l'image et a seulement traité 1 unité).

1) **Est-il possible d'atteindre une situation d'interblocage, si T≤M ? Si T>M ? Justifiez votre réponse.**
2) **Vérifiez, en utilisant l'algorithme du banquier, si l'état courant est certain (sûr ou sauf).**
3) **Le processus P3 demande une unité de mémoire. Doit-on la lui accorder ? Attention, vous devez répondre à cette question en utilisant l'algorithme du banquier.**

Correction

- **Question 1 :**

1) Si T ≤ M, il ne peut pas y avoir d'interblocage car il y a suffisamment d'espace mémoire pour satisfaire toutes les demandes d'allocation des processus.

Par contre, si T>M, on peut atteindre une situation d'interblocage : il n'a plus d'espace mémoire disponible et tous les processus ont encore besoin d'allouer de l'espace mémoire.

2) E= 8 - A = 8 – (1+1+1+3) = 2.

C = (1, 1, 1,3) R = (2 3 3 2)

L'état est sûr car l'ordonnancement suivant permet de terminer tous les processus :

P1 ; A = 3 ; P4 ; A = 6 ; P2 ; A=7 ; P3 ; A = 8.

3) A = 1 ; C = (1, 1, 2, 3) R = (2 3 2 2)

L'état n'est pas sûr car on ne peut satisfaire aucune requête. La demande est refusée.

THE END