UNIVERSITE A.MIRA-BEJAIA
FACULTE DES SCIENCES EXACTES
DEPARTEMENT INFORMATIQUE

SUPPORT DE COURS
SYSTEMES D'EXPLOITATIONS
INTERBLOCAGE (DEADLOCK)

Introduction

L'interblocage survient lorsque chaque processus d'un ensemble de processus détient une ressource requise par un autre processus du même ensemble. L'interblocage est un phénomène stable et persistant. (Exemple: Circulation routière : 4 flux de véhicules sur une intersection.)

Remarque: L'interblocage peut survenir pour une autre raison que l'accès aux ressources. Chaque processus attend de l'autre la réalisation d'une tache (un résultat).

Exemple: les sémaphores S_1 et S_2:

$P_1()\{$	$P_2()\{$
$\vdots$	$\vdots$
$P(S_1);$	$P(S_2);$
$\vdots$	$\vdots$
$P(S_2);$	$P(S_1);$
$\vdots$	$\vdots$
$\}$	$\}$

Lorsqu'un processus désire utiliser une ressource on écrit:

$P()\{$
$\vdots$
Requerir une ressource R ;
Utiliser la ressource R ;
Liberer la ressource R;
$\vdots$
$\}$

Si la ressource n'est pas disponible alors soit le processus attend jusqu'à libération de la ressource, soit le processus reçoit un message d'erreur et reprend.

Il existe deux sortes de ressources:

- Ressource préemptive: son retrait ne provoque pas de dommages au processus
- Ressource non-préemptive: Il y'a perte d'informations si la ressource est retirée au processus

Conditions d'interblocage

Définis par E.J.Coffman (1971): Si l'une des 4 conditions suivantes n'est pas vérifiées alors pas d'interblocage:

1. Exclusion mutuelle: les processus utilisent les ressources (ou leur instances) à un point d'accès
2. Détention et attente: les processus ne libèrent pas les ressources acquises alors qu'ils sont bloqués en attente d'autres ressources.

3. Pas de réquisition: Une ressource acquise ne peut être retirés à un processus
4. Attente circulaire: il existe une chaîne de 2 processus ou plus tel que chacun possède une ressource requise par le processus suivant de la chaîne.

Caractérisation de l'interblocage

Des représentations formelles sont nécessaire pour traiter l'interblocage (graphes, matrices...). Les représentations fondées sur les graphes sont les plus claires:

1. Trajectoire d'exécution: pour expliquer l'interblocage non pour l'éviter.
2. Graphes d'allocation:
   - Ressources critiques: la présence d'un circuit implique un interblocage,
   - Ressources multiples: la présence d'un circuit n'implique pas d' un interblocage (une composante connexe terminale).
3. Graphe des attentes: un arc orienté $P_i \rightarrow P_j \Rightarrow P_i$ attend une ressource détenu par P_j. La présence d'un circuit implique un interblocage. l'intérêt par rapport au précédent graphe: moins de sommets à visiter.

Il existe 4 stratégies de solutions aux interblocages.

1. ignorer le problème (autruche)
2. prèvenir l'interblocage avant qu'il ne survienne
3. détecter l'interblocage (le laisser survenir) et
4. corriger l'interblocage.

Algorithme de l'autruche

(solution optimiste) il suffit d'ignorer le problème dans le cas ou:

- les interblocages sont très rares
- le coût de la prévention est supérieur au coût des dommages.

les systèmes Windows et Unix ont adopté cette stratégie.

Prévention de l'interblocage

on parle de méthodes à priori ou pessimistes, parmi ces méthodes ou peut citer: les méthodes statiques (règles strictes pour empêcher l'interblocage) et les méthodes dynamiques (les requêtes d'accès aux ressources sont examinées au fur et a mesure).

Méthodes statiques il suffit que l'une des 4 conditions d'interblocage soit rendue fausse.

1. Exclusion mutuelle: utiliser des ressources virtuelles ou un spool pour manipuler des ressources critiques.
2. Détention et Attente: une requête pour une ressource est accepté uniquement si le processus n'a acquis aucune ressource ou s'il libère les ressources déjà acquises (ou les préempter) avant de le faire attendre (restitution des ressources précédentes).

3. Pas de réquisition:

   - allocation globale des ressources lors de la création des processus (allocation tout-ou-rien),
   - la préemption s'applique aux processus dont l'état est enregistré à des dates fixes pour pouvoir être restauré.

4. Attente circulaire: on utilise la solution donnée par J.W.Havender (1968) qui est d'ordonner soit les ressources soit les processus.

   - Ordonner les ressources: à chaque ressource est assigné une priorité unique. un processus peut requérir une ressource R_i de priorité P_i si et seulement si $P_i > P_j$ tel que P_j est la plus haute priorité des ressources détenus par ce processus, sinon les ressources acquises de priorité supérieur à P_i doivent être rendues.

     Inconvénient: Soit un processus qui a besoin d'utiliser d'abord R_i puis R_j mais la priorité P_i est supérieur à P_j donc il va requérir R_j, la garder à son niveau même s'il en a pas besoin et requérir R_i et l'utiliser $\Rightarrow$ l'utilisation des ressources n'est pas optimale.

     Solution à l'inconvénient: les priorités basses sont données aux ressources les moins utilisées.

   - Ordonner les processus (approche transactionnelle): A chaque processus (transaction) est associé une priorité (en général le PID suffit). A chaque requête: si la ressource est libre la requête est satisfaite sinon les priorités des transactions sont comparés:

     (a) un processus peut demander un seul type (une instance) de ressource à la fois.

     (b) Si un processus demandeur requiert une ressource d'un processus détenteur de cette ressource alors la règle suivante est appliquée:

        – Si Pr(demandeur) < Pr(detenteur) $\Rightarrow$ le demandeur est bloqué en attente de la libération de la ressource.

        – Sinon $\Rightarrow$ la requête est refusée (transaction annulée)

   - Verrouillage à deux phases: les demandes de requêtes sont faites lors d'une phase d'acquisition. Le processus commence à travailler seulement s'il détient toutes les ressources dont il a besoin. A sa terminaison les ressources sont libérées une à une dans une phase de libération. Règle : aucune requête après une libération !

Méthodes dynamiques Les requêtes à risque seront refusées

- Notion d'état sain (fiable, sur):

  – L'état d'un système d'allocation est sain s'il existe une séquence d'exécution saine.

  – Une séquence d'exécution saine permet une suite d'états réalisables satisfaisants les demandes max de tous les processus.

  Conséquence: les processus déclarent leurs besoins max des ressources.

- Méthode de composition de séquence saine: Soit un état sain réalisé par la suite $< P_1 \dots P_n >$ il faut que P_1 puisse s'exécuter jusqu'à terminaison, libère les ressource qu'il occupait, puis même chose pour P_2 et ainsi de suite jusqu'à P_n. Donc il suffit de trouver un processus dont on peut satisfaire la demande maximale avec les ressources disponibles, puis virtuellement ce processus libère ses ressources qui deviennent disponibles, et recommencer la procédure.

  Remarque: il faudrait vérifier tous les ordonnancements possibles des n processus $\Rightarrow$ n ! possibilités au max.

1. Une création ou une libération d'une ressource maintient le système dans un état sain.
2. Lors d'une création d'un nouveau processus, l'ajouter à une séquence saine ne modifie pas l'état de la séquence.
3. Une augmentation de la déclaration des besoins max d'un système ne peut être autorisé que si elle laisse le systéme dans un état sain.

- Ressources mono-exemplaires: On utilise le graphe d'allocation. Au graphe d'allocation $\{R_i \rightarrow P_j : R_i$ est alloué à $P_j\}$ on ajoute un arc (en pointillé) de requête $\{P_i \rightarrow R_j : P_i$ requiert $R_j\}$. Un arc de requête est transformé en arc d'allocation ssi celui-ci ne génère pas de circuit dans le graphe, sinon la requête est refusée ou retardée.
- Ressources multi-exemplaires: l'algorithme du banquier est utilisé pour maintenir le système d'allocation dans un état sain.

  Données:

  - Nbre de processus: n
  - Nbre de ressources: m
  - Nbre de besoins max de ressources par processus: matrice Max[n,m]
  - Nbre de ressources disponibles a tout instant: vecteur disp[m]
  - Nbre de ressources alloués par processus: matrice Alloc[n,m]
  - Nbre de besoins courants en ressources par processus: Bes[n,m]= Max[n,m]-Alloc[n,m]

  L'algorithme du banquier: Soit une requête (nbre scalaire) Req[i,j]:

  i) vérifier si la requête ne dépasse pas le nbre max déclaré (requête valide): Req[i,j]≤Bes[i,j], sinon requête refusée
  ii) Vérifier si le nbre de ressources demandés est disponible (requête satisfesable): Req[i,j]≤disp[j], sinon requête retardée.
  iii) Créer un état virtuel tel que la requête est satisfaite:
  Alloc[i,j]=Alloc(i,j]+Req[i,j]
  disp[j]=disp[j]-Req[i,j]
  Bes[i,j]=bes[i,j]-Req[i,j]
  iv) Vérifier si l'état virtuel est sain
  v) Si l'état virtuel est sain alors la requête est satisfaite (allocation réellement effectuée)
  vi) Sinon la requête est refusée ou retardée.

Inconvénient: Il est difficile d'estimer les besoins max de chaque processus. Exemple: dans les bases de données, c'est la consultation des premières tables (ressources) qui déterminera les prochaines tables à consulter. En plus, si tous les processus annoncent leurs besoins max en ressources au depart alors le parallélisme disparaît!

Solution de l'inconvénient: Combiner entre les méthodes statiques et dynamiques:

1. ranger les ressources par catégorie
2. utiliser la prévention statique entre catégorie
3. utiliser la prévention dynamique entre ressources d'une même catégorie.

Détection de l'interblocage

On utilisera la méthode de réduction de graphe. C'est un graphe d'allocations /requêtes bipartite orienté pour représenter l'état instantané d'un système à ressources critiques. Le graphe G={P+R, Demande+Possède} ou:

- P={ensemble des sommets processus}
- R={ensemble des sommets ressources}
- Demande={ensemble des arcs de requêtes $P \rightarrow R$ }: il décrit les processus bloqués en attente de la ressource demandé.
- Possède={ensemble des arcs d'allocation $R \rightarrow P$}: il décrit l'état d'utilisation des ressources en cours.

La présence d'un circuit indiquera un interblocage.

Algorithme de réduction de graphe

i) si un sommet processus n'as pas d'arcs de requêtes y émanant alors effacer ses arcs d'allocations.

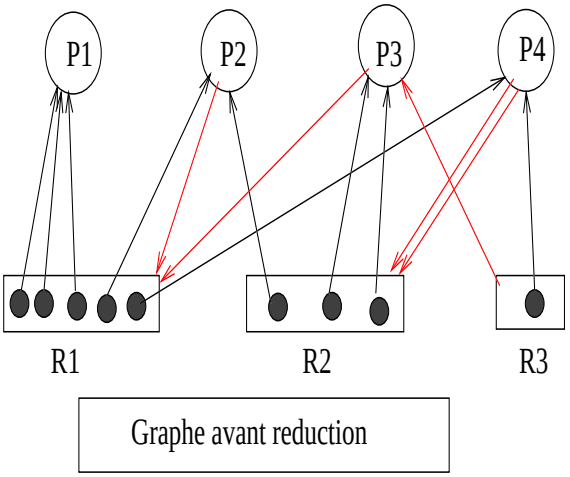
ii) si un sommet ressource n'a pas d'arcs de requêtes y sortant alors effacer ses arcs d'allocations

iii) si un sommet processus a des arcs de requêtes vers une ressource disponible alors ses arcs de requêtes sont transformés en arcs d'allocations

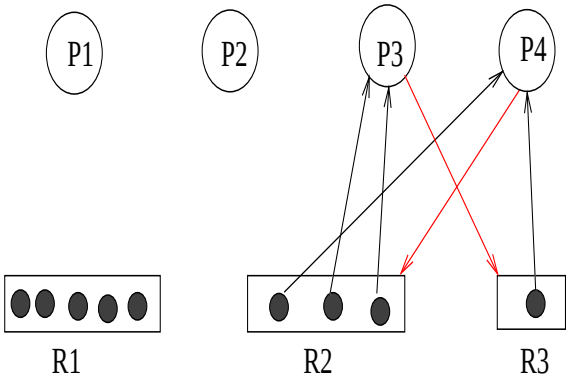
iv) la procédure est répété jusqu'à impossibilité de réduction. le système n'est pas en interblocage s'il ne reste plus aucun arc dans le graphe.

Exemple: Soit la table d'utilisation suivante d'un systeme d'allocation de trois ressources.

Processus	Allocations	Requêtes	disponibles
P1	3 0 0	0 0 0	
P2	1 1 0	1 0 0	0 0 0
P3	0 2 0	1 0 1	
P4	1 0 1	0 2 0	



Graphe avant reduction



Graphe apres reduction

Fréquence d'appel l'algorithme de détection est appelé soit:

- a chaque requête d'un processus (lors d'un ajout d'un arc de requête)

- a chaque allocation d'une ressource (lors d'un ajout d'un arc d'allocation)
- à des périodes de temps fixées au préalable.
- quand le taux d'occupation du processeur passe en dessous d'un seuil fixé (moins de 40%)

Inconvnient:

- Délai trop court ⇒ risque de surcharge de l'unité centrale
- Délai trop long ⇒ si un interblocage survient au début du délai, il risque de gagner tout le système.

Correction d'un interblocage

1. Préemption: le système sélectionne un ensemble de ressources allouées à préempter à un ensemble de processus suivant:
   - priorité du processus
   - temps d'exécution écoulé
   - temps d'exécution total
   - nbre de ressources requises
   - nbre de ressources à préempter pour corriger l'interblocage
   - nbre de processus affectés par cette préemption

   Inconvénient: il y'a un risque de famine si c'est toujours le même processus qui est affecté

   Solution: Un compteur de préemption est associé à chaque processus, et aucun ressource n'est préempté du processus ayant la valeur du compteur la plus élevé
2. Rollback: les processus passe à un état précédant et donc libère des ressources acquises durant le délai de retour. le système pose régulièrement des points de reprise (check point) sur l'exécution du processus. Le check point est un fichier qui contient une image mémoire de l'état du fichier (son contexte)⇒ n check points veut dire n fichiers. Le processus à qui on a décidé de retirer des ressources passe à un état précédant égal au check point nécessaire (on effectue un rollback: un rembobinage)

   Inconvénient: c'est coûteux et souvent impossible à réaliser pour tous les processus plus un risque de perte d'informations.
3. Suppression de processus:

   i) tuer un processus (ou plus) de la chaîne circulaire jusqu'à la briser

   ii) tuer un processus en dehors de la chaîne circulaire détenteur de ressources requises par des processus de la chaîne

   iii) tuer un processus qui peut se ré-exécuter sans perte d'information (exemple: la compilation)
4. Intervention manuelle: si un opérateur existe (administrateur) alors c'est à lui de décider qui tuer ou quoi préempter et combien!

Mme YAICI