Corrigé

1 Cours

Il existe plusieurs raisons, pour lesquelles une solution optimale n'est pas utilisée en pratique. Pour chacune nous donnons une liste non exhaustive d'exemples.

- Une hypothèse au moins n'est pas réaliste.

  1. L'algorithme optimal pour la gestion des defauts de page (quelle page doit-on évincer lorqu'une nouvelle page doit être mise dans un cadre) utilise comme données en entrée, la séquence des pages qui seront utilisées dans le futur. Cette connaissance étant évidemment impossible, l'algorithme optimal ne peut être implémenté.
  2. L'algorithme optimal pour l'ordonnancement de l'unité centrale est l'algorithme du "travail plus court d'abord". Il a besoin de connaître, pour chaque processus, la durée du prochain cycle UC. Cette connaissance étant évidemment impossible en général, l'algorithme optimal ne peut être utilisé.

  Remarque : Ces algorithmes peuvent être implémentés en utilisant comme entrèes, la connaissance du passé. Cela revient à faire l'hypothèse que le futur est le dual du passé.

- La solution possède au moins une propriété non souhaitée.

  1. L'algorithme optimal pour diminuer le temps de recherche des blocs sur le disque, va intuitivement privilégier les blocs *proches* de la position courante de la tête de lecture. Avec une telle solution, une requête peut ne jamais être satisfaite, cet algorithme n'est donc pas équitable, et ne peut donc pas être raisonnablement implémenté.

- Le coût de la solution est trop important.

  1. L'algorithme de prévention de l'interblocage est très peu souvent utilisé du fait de son coût CPU.

2 Sémaphores

Les réponses diffèrent légèrement en fonction de la définition des sémaphores utilisée.

Définition 1 : attente active les opérations wait (P) et signal (V) sont définies comme suit :

P(S)	V(S)
tant que $S \leq 0$ faire rien $S := S - 1$	$S := S + 1$

1. Avec les deux sémaphores S_0 et S_1 initialisés respectivement à -1 et 0, l'ordonnancement peut-être obtenu de la façon suivante :

T_{01}	T_{02}	T_0	T_1
code du processus V(S_0)	code du processus V(S_0)	P(S_0) code du processus V(S_1)	P(S_1) code du processus V(S_0)

2. Avec les deux sémaphores S_0 et S_1 initialisés respectivement à $-N_0+1$ et à $-N_1$, l'ordonnancement peut-être obtenu de la façon suivante :

T_{0i} $(1 \leq i \leq N_0)$ code du processus V(S_0)	T_{1j} $(1 \leq j \leq N_1)$ code du processus V(S_1)	T_0 P(S_0) code du processus V(S_1)	T_1 P(S_1) code du processus V(S_0)

Définition 2 : implémentation classique les opérations wait (P) et signal (V) sont définies comme suit :

P(S) $S.n := S.n - 1$ if $S.n < 0$ then *mettre processus dans* $S.f$	V(S) $S.n := S.n + 1$ if $S.n \leq 0$ then *enlever un processus de* $S.f$

1. Avec les deux sémaphores S_0 et S_1 initialisés à 0, l'ordonnancement peut-être obtenu ainsi :

T_{01} code du processus V(S_0)	T_{02} code du processus V(S_0)	T_0 P(S_0) P(S_0) code du processus V(S_1)	T_1 P(S_1) code du processus V(S_0) V(S_0)

mais également par :

T_{01}
 	
code du processus
V(S_0)
	T_{02}
	
	code du processus
	V(S_0)
		T_0
		
		P(S_0)
		while true do
		    P(S_0)
		    code du processus
		    V(S_1)
			T_1
			
			P(S_1)
			code du processus
			V(S_0)

2. Avec les deux sémaphores S_0 et S_1 initialisés à 0, l'ordonnancement peut-être obtenu ainsi :

T_{0i} $(1 \leq i \leq N_0)$
 	
code du processus
V(S_0)
	T_{1j} $(1 \leq j \leq N_1)$
	
	code du processus
	V(S_1)
		T_0
		
		$(N_0 - 1)$ fois P(S_0)
		while true do
		    P(S_0)
		    code du processus
		    V(S_1)
			T_1
			
			N_1 fois P(S_1)
			while true do
			    P(S_1)
			    code du processus
			    V(S_0)

mais également par :

T_{0i} $(1 \leq i \leq N_0)$ code du processus V(S_0)	T_{1j} $(1 \leq j \leq N_1)$ code du processus V(S_1)	T_0 N_0 fois P(S_0) code du processus V(S_1) $(N_0 - 1)$ fois V(S_0)	T_1 $(N_1 + 1)$ fois P(S_1) code du processus V(S_0) N_1 fois V(S_1)

3 Interblocages

1. L'objectif de l'algorithme (dit du banquier) de prévention de l'interblocage est de maintenir le système dans un état sain. Pour cela les ressources ne sont allouées à un processus, que si l'état résultant de cette allocation est un état sain.

2. l'objectif de la variante, utilisée pour la détection de l'interblocage, est de savoir si le système est dans un état bloquant ou bloqué. La correction consiste à forcer la remise du système dans un état sain ou malsain.

3. L'état du système est bloqué si le système est inactif. Cela signifie qu'aucun des processus ne peut être actif, (ie il ne possède aucune ressource, ou bien sa requête courante ne peut être satisfaite). Par conséquent, un état du système n'est pas bloqué s'il existe un processus actif. Avec les notations (m processus, n types de ressources, Disponibles : vecteur des ressources, Allouées, Requêtes : matrices des ressources par processus), la variante suivante permet donc de détecter si le système est dans un état bloqué.

```
bloque := vrai
pour i de 1 a m faire
   si ((0,0,...,0) < Requetes[i] < Disponibles) // la requete peut etre satisfaite
      ou
      ((0,0,...,0)=Requetes[i] et Allouees[i]>(0,0,...,0)) // Pi utilise des ressources
   alors
      bloque := faux
```

4. Les schémas de programme des trois algorithmes sont ceux-ci :

Algorithme du banquier	**Variante du banquier**	**Nouvel algorithme**
Trouver un ordonnancement (sans retour arriere) $i_1, i_2, \ldots i_m$ pour chaque i condition sur n valeurs	Trouver un ordonnancement (sans retour arriere) $i_1, i_2, \ldots i_m$ pour chaque i condition sur n valeurs	Trouver un processus i parmi $i_1, i_2, \ldots i_m$ condition sur n valeurs

Les compléxités sont donc :

Algorithme du banquier	**Variante du banquier**	**Nouvel algorithme**
$o(m^2n)$	$o(m^2n)$	$o(mn)$

5. Afin de comprendre l'influence des diverses actions, nous ajoutons sur le graphe d'allocation des ressources les changements d'états qu'elles peuvent occasionner (il faut noter que ces actions peuvent aussi ne pas occasionner de changements d'états).

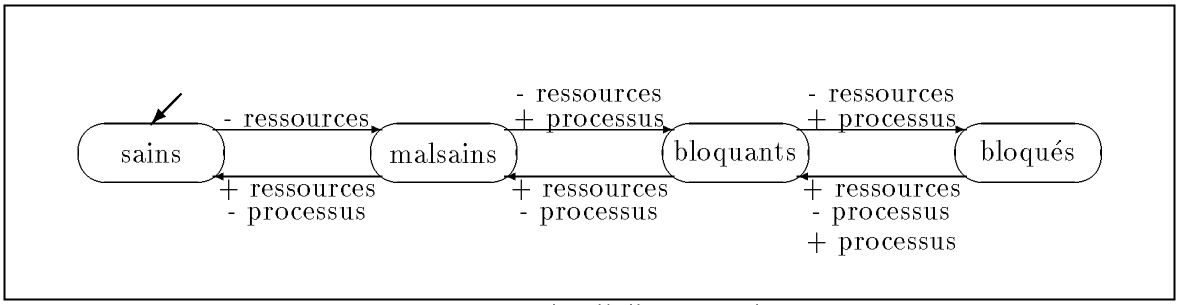

Figure 4: Nouveau graphe d'allocation des ressources.

Il faut remarquer qu'ajouter des processus ne transforme pas un état sain en état malsain. Ces processus peuvent en effet être mis en fin de l'ordonnancement existant pour donner un nouvel ordonnancement.

La seule contrainte est que leur requête maximale soit inférieure aux ressources globales, ce qui semble raisonnable.

Le tableau suivant représente les conséquences sur la validité des algorithmes. "Valide" signifie que le dernier résultat calculé par l'algorithme reste vrai après la modification du système. "Non Valide" signifie qu'il faut exécuter l'algorithme après la modification du système (les deux variantes sont des algorithmes appelés périodiquement). "Critique" signifie que l'algorithme peut-être désormais inutilisable.

	Banquier	Variante du cours	Nouvelle variante
Ajouter des ressources	Valide	Non Valide	Non Valide
Enlever des ressources	Critique	Valide	Valide
Ajouter des processus	Valide	Valide	Non Valide
Enlever des processus	Valide	Non Valide	Non Valide

6. La question 4 montre que le nouvel algorithme est de loin le plus efficace. Malheureusement il est inexploitable, d'une part il ne résiste pas à l'ajoût et à la suppression de processus, qui sont des modifications fréquentes sur un système réel; d'autre part la probabilité qu'un système soit bloqué est extremement faible : Il ne faut pas confondre un systeme bloqué (tous les processus inactifs) et un interblocage (des processus sont bloqués).

   Le choix entre les deux autres options est plus délicat, et dépend d'autre critères. On peut schématiquement faire le tableau suivant :

	Prévention	Détection
Les propriétés		
Fréquence des interblocages	nulle	dépend du système
Coût surcharge CPU	élevé	paramétrable
Coût de remise en état du système après un interblocage		dépend du système
Risque de famine	faible	élévé

Avec le tableau précédent, il est possible de construire le tableau suivant, qui indique le choix à effectuer, en fonction d'objectifs assignés et/ou d'hypothèses connues sur le système.

	Prévention	Détection
Les choix en fonction des hypothèses et/ou objectifs		
Si la fréquence des interblocages est :	élevée	faible
Si le risque de l'interblocage est :	interdit	accepté
Si le coût surcharge CPU est :	accepté	pénalisant
Si le coût CPU après un interblocage est :		accepté

Par exemple, le risque d'interblocage est interdit dans un système embarqué ou l'intervention humaine est impossible.

Le problème de la famine existe avec les deux solutions, mais est sans doute moins fréquent avec la prévention.