UNIVERSITÉ DE ROUEN	ANNÉE 2007-2008
FACULTÉ DES SCIENCES	SYSTÈME D'EXPLOITATION
LICENCE INFORMATIQUE 3ème ANNÉE	CONTRÔLE CONTINU
26 NOVEMBRE 2007	2H00

Aucun document n'est autorisé.

Exercice 1 Soit $T_1, T_2, T_3, T_4, T_5, T_6$ six processus dont la durée relative (t_i)et le temps d'arrivée (τ_i) sont donnés par le tableau suivant :

	τ_i	t_i
T_1	6	0
T_2	12	0
T_3	4	1
T_4	2	2
T_5	6	3
T_6	12	3

Vous donnerez un diagramme de Gantt pour chacun des algorithmes suivants. Vous donnerez également le temps de réponse de chacun des processus pour chacune des politiques d'ordonnancement

1. FCFS
2. SJF
3. tourniquet avec un quantum $q = 2$
4. tourniquet avec un quantum $q = 2$ et priorité dynamique $P_0(T_j) = j$ et $P_{i+1}(T_j) = P_i(T_j) + TTU/2$ où TTU est le temps total utilisé par le processus.

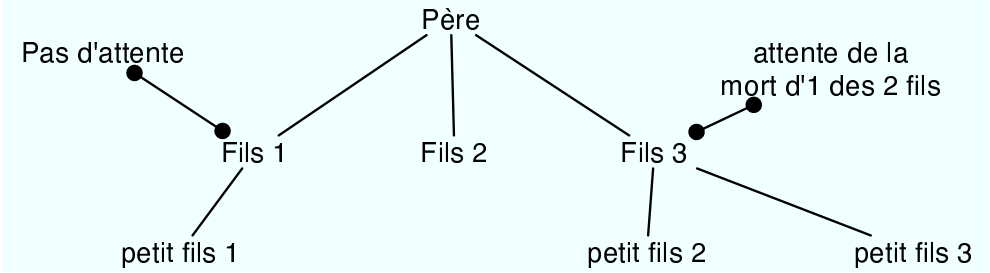
Exercice 2

1) Pour chacune des transitions suivantes entre les états des processus, indiquez si la transition est possible. Si c'est le cas, donnez un exemple.

- En exécution - prêt
- En exécution - bloqué
- Bloqué - en exécution
- En exécution - terminé

Exercice 3

1) Écrire un programme permettant d'obtenir l'arbre des processus suivants :



Exercice 4 On suppose que des processus souhaitent accéder à une variable partagée x à l'aide d'une instruction du type

$x \longleftarrow$ Expression(x).

1) Afin que plusieurs processus n'aient pas accès à x en même temps, indiquer comment protéger l'accès à la variable x à l'aide de sémaphores.

On suppose dèsormais qu'il y a, en plus des processus "écrivains" de la question précédente, des processus "lecteurs" qui effectuent des instructions du type

traiterExpression(x).

La politique d'accès à la variable partagée sera alors la suivante :

- Plusieurs lecteurs pourront avoir accès à la variable x en même temps ;
- Un lecteur n'a pas accès à la variable si un écrivain y a déjà accès ;
- Les écrivains ne peuvent avoir accès à la variable si un autre processus (lecteur ou écrivain) y a déjà accès.

On propose l'algorithme 1 pour résoudre notre problème.

Algorithme 1 Les processus écrivains et lecteurs

Requis : Un sémaphore S_L initilisé à 0 (compte les lecteurs)
Requis : Un sémaphore S_E initilisé à 0 (compte les écrivains)

Processus Lecteur	**Processus** Écrivain
V(S_L)	Z(S_L)
Z(S_E)	Z(S_E)
traiterExpression(x)	V(S_E)
P(S_L)	$x \longleftarrow$ Expression(x)
fin Processus	P(S_E)
	fin Processus

2) Montrez, à l'aide d'un graphe d'état que la solution proposée est incorecte.

3) Proposez une solution répondant au problème posé.

Nous supposons maintenant que les processus accèdent à un tableau T en mémoire partagée. On ne différenciera plus les processus lecteurs des processus écrivains mais, par contre, on souhaite protéger l'accès au tableau composante par composante. Ainsi, si un processus accède à la composante i du tableau alors un autre processus ne pourra pas accéder à la composante i, par contre il pourra accéder aux autres composantes non utilisées.

Comme précédemment, la synchronisation se fera à l'aide de sémaphores. Nous gèrerons ici un tableau de sémaphores S indicé de 1 à N.

Pour gérer l'accès aux composantes on se propose d'écrire une fonction `indice prend_semahore (indice j)` et une procédure `rend_semaphore (indice j)`.

La fonction `indice prend_semaphore(indice j)` retourne l'indice du sémaphore à utiliser par le processus pour protéger la composante j de tableau T.

La procédure `rend_semaphore(indice j)` indique que le processus n'a plus besoin d'avoir accès à la composante j du tableau.

4) Donner le code de `prend_semaphore` et de `rend_semaphore` lorsqu'on peut disposer d'autant de sémaphores qu'il y a de composantes dans le tableau T.

5) Donner le code de `prend_semaphore` et de `rend_semaphore` lorsqu'il y a M composantes dans le tableau T, avec $M > N$.

Vous pourrez considérer que la fonction `indice prend_semaphore(indice j)` renvoie 0 s'il n'y a plus de sémaphores de libres pour protéger l'accès à la composante du tableau.