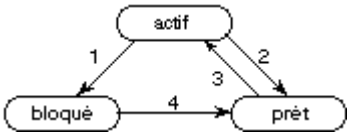
I) PROCESSUS

Exécution de processus en multiprogrammation

On considère un système monoprocesseur dans lequel les processus partagent un disque comme seule ressource (autre que le processeur). Cette ressource n'est accessible qu'en accès exclusif et non requérable, c'est-à-dire qu'une commande disque lancée pour le compte d'un processus se termine normalement avant de pouvoir en lancer une autre. Un processus peut être en exécution, en attente d'entrée-sortie ou en attente du processeur.

A- Expliquer le schéma suivant représentant les états possibles d'un processus et les transitions entre ces états. Expliquer pourquoi certaines transitions ne sont pas possibles.



B- En fait l'état bloqué se divise en deux états : attente de la ressource disque et attente de la fin d'exécution de l'opération. Les demandes d'entrées-sorties sont gérées à l'ancienneté, et l'allocation du processeur est faite selon la priorité affectée au processus, et représentée par une valeur entière. Le processus prioritaire est celui qui a la plus grande valeur et si deux processus ont même priorité, c'est le plus ancien dans la file d'attente des processus prêts.

Nous considérons les 4 processus dont le comportement est le suivant (la priorité au démarrage est indiquée entre parenthèses) :

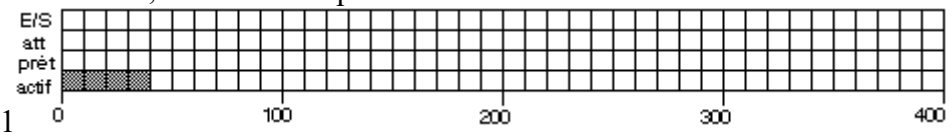
P1 (100) Calcul pendant 40 ms
Lecture disque pendant 50 ms
Calcul pendant 30 ms
Lecture disque pendant 40 ms
Calcul pendant 20 ms

P2 (99) Calcul pendant 30 ms
Lecture disque pendant 80 ms
Calcul pendant 80 ms
Lecture disque pendant 20 ms
Calcul pendant 10 ms

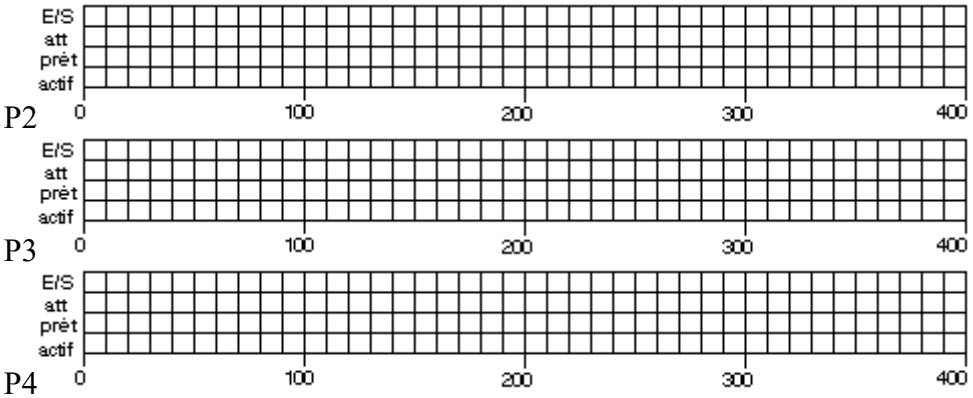
P3 (98) Calcul pendant 40 ms
Lecture disque pendant 40 ms
Calcul pendant 10 ms

P4 (97) Calcul pendant 80 ms

B.1- Les 4 processus sont lancés en même temps et gardent leur priorité initiale pendant toute leur exécution. Établir le chronogramme des 4 processus sur le diagramme suivant. Vous noircirez les cases correspondant à l'état du processus, comme cela a été fait pour le début du processus P1, à titre d'exemple.



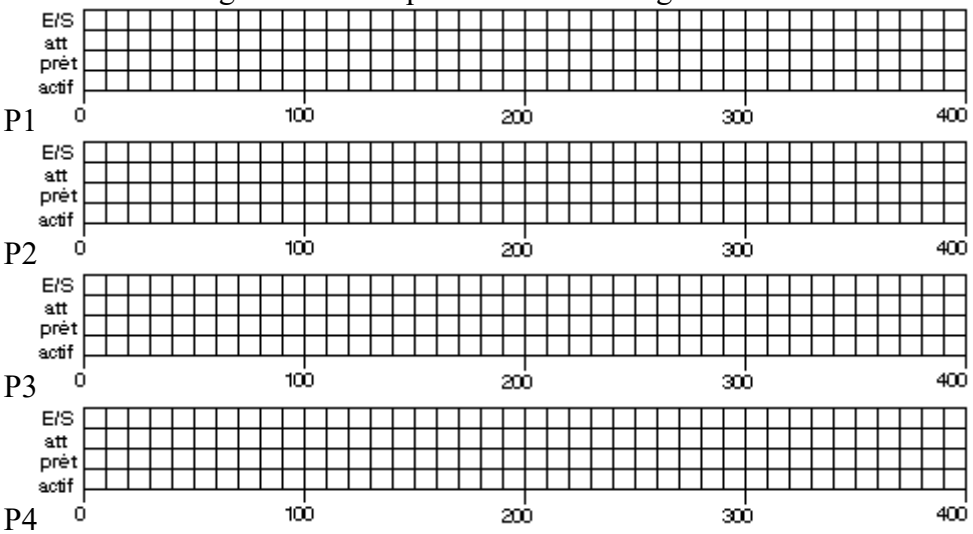
P1



B.2- Les 4 processus sont lancés en même temps, mais leur priorité est variable. Chaque fois qu'un processus quelconque quitte l'état bloqué, on recalcule la priorité de chaque processus selon la formule suivante :

Priorité Nouvelle = Priorité Initiale - (Temps processeur utilisé) / 10

Établir le chronogramme des 4 processus sur le diagramme suivant.



C- Donner le temps total de l'exécution de ces 4 processus dans les trois cas suivants :

C1 L'activation des 4 processus est demandée à l'instant initial et ils s'exécutent en monoprogrammation dans l'ordre P1, P2, P3 puis P4,

C2 Gestion de B.1,

C3 Gestion de B.2.

D- Comparer les temps de réponse moyens dans les trois cas C1, C2, C3

E- Donner le taux d'utilisation du processeur dans les 3 cas C1, C2 et C3

Exercice 2

Donner et comparer les assignations produites par les algorithmes FIFO, PCTE, tourniquet avec un quantum de 1, PCTER dans l'exemple suivant :

Ordre d'arrivée des tâches :

	T1	T2	T3	T4	T5	T6	T7
durée	7	4	6	1	2	4	1
date d'arrivée	0	0	1	1	1	2	2

Exercice 3
Sur un ordinateur, l'ordonnanceur gère l'ordonnancement des processus par un tourniquet avec un quantum de 100 ms, sachant que le temps nécessaire à une commutation de processus est de 10 ms, calculer le temps d'exécution moyen pour :

	T1	T2	T3	T4	T5	T6	T7
durée	700	400	600	100	200	400	100
date d'arrivée	0	0	100	100	150	200	200

Si l'on définit le rendement du processeur comme le rapport *temps pendant lequel l'UC exécute les processus/temps total de traitement,* calculer le rendement en ce cas.

Exercice 4
Un SE utilise 3 niveaux de priorité (numérotés par ordre croissant). Le processus se voit affecter un niveau fixe. Une file de processus est attachée à chaque niveau. Chaque file est gérée par un tourniquet avec un quantum de 0,5. Un tourniquet de niveau n n'est activé que si toutes les files de niveau supérieur est vide.
Que peut-il se passer ?
Donner l'assignation pour :

	T1	T2	T3	T4	T5	T6	T7
durée	7	4	6	1	2	4	1
date d'arrivée	0	0	1	1	1	2	2
priorité	2	3	1	2	3	1	2

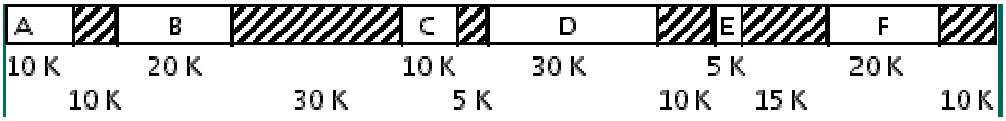
Maintenant on suppose que la priorité n'est pas fixe. Toutes les 2 unités de temps, tout processus n'ayant pas disposé de l'UC monte d'un niveau alors que ceux en ayant disposé 2 fois en descendent. Donner la nouvelle assignation.

II) GESTION DE LA MEMOIRE

Exercice 1

Allocation par partitions variables

Soit un système qui utilise l'allocation par partitions variables. On parle aussi d'allocation par zones quand la taille des zones allouées est variable, ou d'allocations par blocs quand la taille des blocs alloués est fixe.
L'allocation se fait selon le choix first fit. C'est-à-dire que la première zone rencontrée dont la taille est supérieure ou égale à la taille du processus à charger est celle qui est allouée au processus.
On considère à l'instant t l'état suivant de la mémoire centrale :



Représenter l'évolution de la mémoire centrale en fonction de l'arrivée des évènements suivants :

1 - Arrivée du programme G (20 K)
2 - Départ du programme B
3 - Arrivée du programme H (15 K)
4 - Départ du programme E
5 - Arrivée du programme I (40 K)

Exercice 2

Pagination

Soit la liste des pages virtuelles référencées aux instants t = 1, 2, ..., 11 :
3 5 6 8 3 9 6 12 3 6 10
La mémoire centrale est composée de 4 cases initialement vides.
Représentez l'évolution de la mémoire centrale au fur et à mesure des accès pour chacune des deux politiques de remplacement de pages FIFO et LRU. Notez les défauts de pages éventuels.

Un programme a besoin de 5 pages virtuelles numérotées 0 à 4. Au cours de son déroulement, il utilise les pages dans l'ordre suivant : **012301401234**
Question :

1. S'il reste 3 pages libre en mémoire, indiquer la séquence des défauts de page, sachant que l'algorithme de remplacement est FIFO,
2. Même question avec 4 pages disponibles en mémoire. Observation ?

Soit la table de pages suivante :

0	4
1	6
2	8
3	9
4	12
5	1

Sachant que les pages virtuelles et physiques font 1K octets, quelle est l'adresse mémoire correspondant à chacune des adresses virtuelles suivantes codées en hexadécimal :
142A et 0AF1

Exercice 3

Temps d'accès effectif

Sur un système qui a recours à la mémoire paginée à la demande, il faut 200 ns pour satisfaire une requête mémoire si la page reste en mémoire. Si tel n'est pas le cas, la requête prend 7 ms si un cache libre est disponible ou si la page à extraire n'a pas été modifiée. Il faut par contre 15 ms si la page à extraire a été modifiée.
Quel est le temps d'accès effectif si le taux de défaut de page est de 5 % et que, 60 % du temps, la page à remplacer a été modifiée ?

Exercice 4

L'algorithme LRU est théoriquement excellent à la fois pour le cache du système de fichiers et pour le remplacement de page en mémoire virtuelle. Cependant, de nombreux systèmes d'exploitation l'utilisent uniquement pour le système de fichiers et préfèrent des algorithmes de « compromis », donnant des taux de fautes de pages plus élevés que LRU, pour gérer les pages de mémoire virtuelle. Pouvez-vous expliquer les raisons de ce choix, et pourquoi LRU est utilisable pour le cache du système de fichiers ?

Exercice 5

Soit un système disposant de 16 Mo de mémoire physique, utilisant une taille de page de 1 Mo. Quatre processus, numérotés de 1 à 4, tournent sous ce système, dans cet ordre. Chacun de ces processus peut demander l'accès et retenir jusqu'à 8 pages, numérotées de 1 à 8. Pendant l'exécution, chacun d'eux demande un accès à des pages mémoires, suivant le tableau ci-après :

	Numéro de page															
Processus 1	1	2	3	4	5	6	5	5	1	2	3	4	1	2	3	4
Processus 2	2	3	4	2	3	4	2	3	2	3	2	3	2	3	2	3
Processus 3	1	2	3	4	1	2	3	4	1	2	2	2	1	6	6	1
Processus 4	5	6	5	6	1	1	6	6	1	8	8	8	5	1	5	1

Supposons que les processus soient ordonnés par l'algorithme du tourniquet avec un quantum de temps tel qu'il y ait 4 accès mémoire par quantum. Est-il possible d'exécuter les processus sans qu'aucun d'eux soit suspendu ?

Exercice 6 : Fragmentation

a) Quel est le problème principal de la fragmentation ?
b) Que risque-t-il de se passer pour la mémoire libre, vis-à-vis de la fragmentation, quand on alloue des blocs de mémoire ? Quel problème cela pose-t-il ?
c) Existe-il un problème de fragmentation avec les partitions de tailles fixes ?
d) Peut-on autoriser une zone de données à être séparée en mémoire ? Quel effet cela a-t-il sur l'efficacité du système ?

III) SYNCHRONISATION ET COMMUNICATION

1) Synchronisation : le coiffeur

Une illustration classique du problème de la synchronisation est celui du salon de coiffure. Dans le salon de coiffure, il y a un coiffeur C, un fauteuil F dans lequel se met le client pour être coiffé et N sièges pour attendre.

- S'il n'a pas de clients, le coiffeur C somnole dans le fauteuil F.
- Quand un client arrive et que le coiffeur C dort, il le réveille, C se lève. Le client s'assied dans F et se fait coiffer.
- Si un client arrive pendant que le coiffeur travaille :
  - si un des N sièges est libre, il s'assied et attend,
  - sinon il ressort.

En considérant que vous êtes devant un système d'exploitation, décrivez les différents éléments ci-dessus présentés.

2) La SCNF a reçu un grand nombre de réclamations pour un problème lié aux réservations et qui se produisait par grande affluence. Chaque train a un nombre limité de places. Supposons qu'un voyageur décide de changer de train. S'il commence par annuler la réservation pour le premier train et que le second soit plein, il risque de ne plus retrouver de place dans le train qu'il vient d'annuler. Cela peut arriver si un autre voyageur réserve pour le premier train dans l'intervalle. La SNCF souhaite donc mettre en place un système de

permutation de réservation, permettant à un voyageur de changer de billet sans perdre son trajet initial.

a) Décrire par vos propres expressions la solution à implémenter.

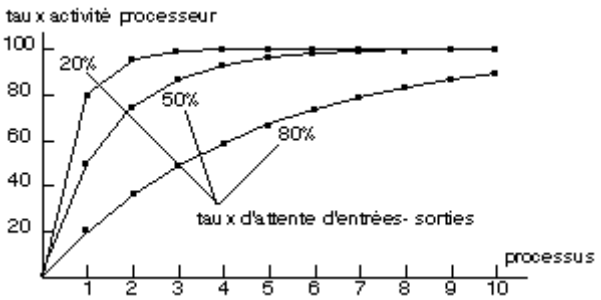
IV) Ordonnancement

1) Soit **TS** le temps de service d'un travail, c'est à dire le temps écoulé entre la soumission du travail et sa fin. On considère un système de traitement séquentiel (*batch*) dans lequel quatre travaux arrivent dans l'ordre suivant :

NO du travail	Instant d'arrivée	Durée
1	0	8
2	1	4
3	2	9
4	3	5

a) Donner le **TS** moyen dans le cas où l'on adopte la politique PAPS (Premier Arrivé, Premier Servi, ou encore FCFS, *Fist Come Fisrt Served*)
b) Donner le **TS** moyen dans le cas où l'on adopte la politique préemptive : PCA (le plus court d'abord, ou encore SJN, *Shortest Job Next*)

2) Voici ci-dessous une représentation. Expliquez ce dessin (1/2 page)



3) On dispose d'une machine équipée d'un système de page à la demande. L'espace mémoire centrale disponible pour les processus est de 800Ko. Un défaut de page entraîne le blocage du processus en attente de page pendant en moyenne 50 millisecondes.

Les caractéristiques moyennes des programmes de l'installation sont les suivantes :

- Lorsqu'un programme est exécuté seul, dans 800 Ko de Mémoire, il n'y a pas de défaut de page, il utilise 40 secondes d'Unité Centrale, et attend ses entrées-sorties pendant 60 secondes.
- Lorsqu'il est exécuté dans un espace mémoire réduit, les nombres de défaut de page sont les suivants :

Espace mémoire	100 Ko	200 Ko	400 Ko
Défauts de page	10000	2500	1000

On répartit équitablement l'espace mémoire centrale disponible entre 2, puis 4 et ensuite 8 processus. Donner dans chaque cas le taux d'attente pour entrées-sorties, et en déduire le taux d'activité du processeur.
Quels commentaires cela vous suggère-t-il, selon que vous preniez le point de vue de l'utilisateur, ou le point de vue du chef d'exploitation ?

4) Les compatibles PC sont équipés de microprocesseurs de la famille 8088/286. Lors de la mise sous tension, ces microprocesseurs forcent le compteur ordinal à une valeur prédéfinie (0FFFF0 en hexadécimal), et exécutent donc l'instruction située à cet endroit en mémoire.
a) Expliquer comment, à votre avis, les constructeurs réussissent à obtenir que, lors de la mise en route par l'utilisateur, le compatible PC charge automatiquement le système MS-DOS depuis une disquette ou depuis le disque dur.
b) La solution adoptée pour résoudre a), ne permettrait-elle pas d'éviter de charger MS-DOS ? Quel est l'intérêt alors de ce chargement ?

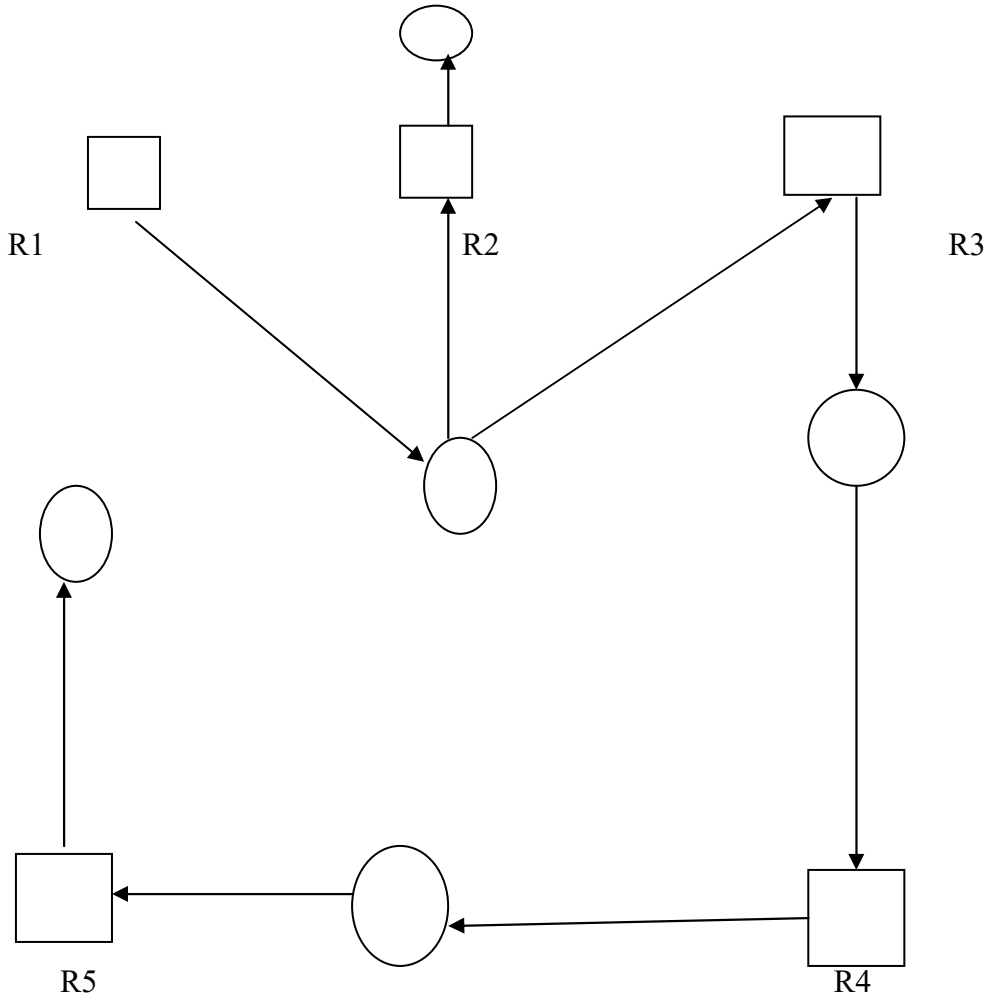
5) Linux a un noyau monolithique. Vrai ou Faux ?

6) Quelle est la réponse juste parmi les réponses ci-dessous ?

Le **pseudo-parallélisme** c'est lorsque le processeur exécute un programme pendant un instant puis un autre pendant une durée d'exécution assez petite.
Le **pseudo-parallélisme** c'est l'exécution simultané deux programme sur le même processeur.
Le **pseudo-parallélisme** c'est l'exécution des processus d'un programme sur plusieurs processeurs

7) Quand on parle d'efficacité, pour un ordonnanceur, à quoi se réfère-t-on ?
8) Donnez un exemple où un processus a besoin d'avoir accès aux fonctionnalités du système d'exploitation. Pourquoi passe-t-on par le système d'exploitation pour faire de telles actions, et pourquoi le processus ne les fait-il pas directement ? Comment invoque-t-on une fonctionnalité du système d'exploitation ?

9) Le graphe d'allocation de ressources pour un système à un moment donné est le suivant :



Y a-t-il risque d'interblocage en ce moment ? Si oui, justifier. Si non, modifier ce graphe en ajoutant une flèche pour que le risque d'interblocage existe.

10) On considère un système composé de quatre ressources identiques qui sont partagées par trois processus. Chacun des processus utilise, au plus deux ressources. Montrez qu'un deadlock est impossible dans un tel système.

11) Six étudiants viennent voir l'enseignant Nicolas. Nicolas dispose d'une heure pour discuter avec les étudiants et passer d'un étudiant à un autre lui prend une minute. La table suivante donne l'ordre d'arrivée des étudiants et le temps dont ils ont besoin. Les étudiants arrivent tous au même moment.

Etudiant	Ordre d'arrivée	Temps requis
E1	1	45 min
E2	2	15 min
E3	3	10 min
E4	4	5 min
E5	5	6 min
E6	6	20 min

a) Sachant que Nicolas ne peut consacrer en tout qu'une heure à l'ensemble des étudiants, quels est l'algorithme d'ordonnancement qui lui permettra de traiter le plus d'étudiants complètement. Justifier votre réponse par un diagramme de Gant.

- premier arrivé, premier servi
- le plus court d'abord
- tourniquet (quantum=5mn, l'intervalle de 1 min reste valable même si Nicolas reste avec le même étudiant pendant 2 quantums de suite).

b) Un ordonnancement de type tourniquet peut utiliser différents quantums. Donner une raison d'utiliser un petit quantum ainsi qu'une raison d'utiliser un grand quantum.

c) On choisit un quantum de telle manière qu'il soit plus long que n'importe quel processus. Quel sera l'effet résultant de ce choix ?

Considérons maintenant que Nicolas a tout son temps. Donner le diagramme de Gantt et le temps moyen d'exécution pour un ordonnancement de type tourniquet avec priorités. Chaque fois qu'un étudiant quitte le professeur à la fin de son quantum de temps, on recalcule sa priorité. Priorité Nouvelle = Priorité Initiale - (Temps processeur utilisé) / 10

Voici le nouveau table au d'arrivée et de priorité des étudiants.

Etudiant	Date 'arrivée	Temps requis	Priorité
E1	0	45 min	1
E2	5	15 min	2
E3	5	10 min	3
E6	5	20 min	1
E4	10	5 min	2
E5	10	6 min	2

V) Interruption

1) Quel est l'intérêt des interruptions pour le système d'exploitation ? Pouvez-vous donner un exemple d'interruption logiciel ?
2) Le simulateur Nachos, bien que très réaliste sur de nombreux aspects, introduit tout de même plusieurs simplifications par rapport aux systèmes d'exploitations réels. En particulier, la protection des régions de code nécessitant un accès en exclusion mutuelle s'effectue dans Nachos en masquant les interruptions pendant toute la durée du traitement (au moyen de l'instruction interrupt->SetLevel(IntOff)). En voici une illustration avec le code de la primitive Semaphore::V() :

```
void Semaphore::V()
{
    Thread *thread;
    IntStatus oldLevel = interrupt->SetLevel(IntOff);

    thread = (Thread *)queue->Remove();
    if (thread != NULL)    // make thread ready, consuming the V immediately
        scheduler->ReadyToRun(thread);
    value++;

    (void) interrupt->SetLevel(oldLevel);
}
```

a) Expliquez pourquoi, sur une machine multiprocesseur, une telle stratégie ne fonctionnerait pas.
b) Quel mécanisme de bas niveau peut-on utiliser dans les noyaux pour protéger l'accès à des portions de code très courtes ? Décrivez précisément une situation o`u cette utilisation peut conduire à de très mauvaise performances (pensez à des processus s'exécutant sur le même processeur).
c) Expliquez comment, en utilisant conjointement le mécanisme précédent et le masquage des interruptions, on pourrait résoudre ce problème (on ne demande pas d'écrire du code précis).