École Polytechnique de Montréal

Département de Génie Informatique et de Génie Logiciel

Cours INF2610

Examen final

Hiver 2010

• Date : **19 avril 2010** de **9h30 à 12h00**	• Pondération : 4**0 %**
• Professeur : **Boucheneb Hanifa**	• Nbre. de questions : **3**
• Documentation permise : Polycopié du cours	• Total : **20 points**
• Calculatrices programmables et cellulaires non permis	

Question 1 (8 pts) : Généralités

Répondez aux questions suivantes (les réponses doivent être justifiées, **concises et claires)** :

a) [1 pt] Donnez l'arborescence des processus créés par le bout de code suivant (supposez que l'appel à « fork » ne retourne pas d'erreur) :

```
for (i=1; i<=4; i++)
{        pid = fork();
         if (pid != 0) printf("%d\n", pid);
}
```

b) [1 pt] Considérez l'exemple Serveur Web (usage des threads) vu en classe. Est-il plus approprié d'utiliser des threads noyau dans ce cas ?

c) [1 pt] Deux threads appartenant à un même processus peuvent-ils se synchroniser au moyen d'un sémaphore du noyau, s'ils sont implémentés au niveau utilisateur ? Supposez que ce sémaphore est utilisé uniquement par ces deux threads.

d) [1 pt] Soient n tâches qui partagent m ressources de type R. Le nombre maximal de ressources de type R que peut détenir une tâche est 2. Le temps d'utilisation des ressources est supposé borné. Quelle est la valeur minimale de m qui garantit l'absence d'interblocage ?

e) [3 pts] Montrez comment les sémaphores compteurs (qui peuvent détenir une valeur arbitraire) peuvent être implémentés en utilisant uniquement des sémaphores binaires et des instructions ordinaires (ne comportant pas d'autres appels système). Utilisez les types SemaphoreC et SemaphoreB pour distinguer les sémaphores compteurs des sémaphores binaires. Les files d'attente des sémaphores binaires et des sémaphores compteurs sont gérées selon la même discipline qui est FIFO (First In First Out).

On veut maintenant offrir la possibilité de gérer les files d'attente des sémaphores compteurs selon une discipline LIFO (Last In First Out) ou

FIFO, indiquée par l'utilisateur lors de la déclaration. Expliquez clairement les modifications à faire pour permettre ce choix de discipline.

f) [1 pt] Un système temps réel doit gérer une communication vocale qui consomme 1 ms de temps CPU toutes les 5 ms, ainsi qu'un flux vidéo à 25 trames par seconde. Le traitement de chaque trame nécessite 20 ms de temps CPU. Est-il possible d'ordonnancer ce système ?

Question 2 (7 pts) : Synchronisation

La capacité maximale d'un lave-auto est de 1 camion ou 3 voitures (c.-à-d. on peut avoir jusqu'à 3 voitures au lavage au même temps, par contre on ne peut pas avoir plus d'un camion à la fois). On vous demande de gérer l'accès au lave-auto de sorte que la capacité maximale du lave-auto soit respectée. Le lave-auto a une seule entrée et une seule sortie. Supposez que les camions et les voitures sont des threads qui demandent périodiquement l'accès au lave-auto.

a) [2,5 pts] Synchronisez l'accès au lave-auto en utilisant les sémaphores. Votre solution ne doit privilégier ni les camions, ni les voitures.
b) [2,5 pts] Synchronisez l'accès au lave-auto en utilisant les moniteurs et éventuellement les variables de condition. Votre solution ne doit privilégier ni les camions, ni les voitures.
c) [2 pts] Donnez un scénario d'exécution pour a) et pour b) dans le cas où à l'entrée du lave-auto, on a, dans l'ordre, 1 voiture, 1 camion et 1 voiture. Supposez qu'aucune autre voiture ni camion n'arrive dans le lave-auto et que les demandes d'accès sont traitées selon l'ordre d'entrée. Le scénario doit indiquer l'ordre d'exécution des instructions et l'évolution des valeurs des variables.

Attention : Vous pouvez donner vos réponses sous forme de pseudo-code. Par contre, la solution doit être claire. Vous devez expliquer le rôle de chacune des variables utilisées et, s'il y a lieu, vos suppositions. Vous ne devez en aucun mélanger les mécanismes de synchronisation.

Question 3 (5 pts) : Ordonnancement de processus

a) [3 pts] On considère un système monoprocesseur et les 5 processus P1, P2, P3, P4 et P5 décrits dans le Tableau 1 :

Processus	Dates d'arrivée	Priorités	Temps d'exécution
P1	0	3	5 (2) 3
P2	1	5	6
P3	2	2	2
P4	4	1	4
P5	5	4	3

Tableau 1

On suppose que :

- le temps de commutation est égal à 0,
- La priorité 1 est la priorité la plus basse.

Donnez, pour chacun des algorithmes suivants, le diagramme de Gantt montrant l'ordonnancement des processus ainsi que les temps de séjour moyens :

1) Ordonnancement circulaire avec un quantum égal à 3 (sans tenir compte des priorités);
2) Ordonnancement par priorités pures (pas de quantum, préemptif);

b) [2 pts] On veut appliquer l'ordonnancement « le plus court d'abord » aux processus interactifs. Cet algorithme se base sur l'estimation des temps d'exécution des rafales des processus avec α = 1/2. Supposez que 2 processus A et B sont à l'état prêt. Le processus A a déjà exécuté 2 rafales qui ont respectivement duré 20 et 35 ms. Le processus B a exécuté 3 rafales de durées respectives 40, 15 et 25 ms. Le temps estimé pour la première rafale est fixé à 30 ms pour tous les processus. Lequel des deux processus A et B sera élu par l'ordonnanceur ?

Le corrigé final INF2610

Question 1

a) Rép:

P0 crée 4 fils P01, P02, P03 et P04

P01 crée 3 fils P011, P012 et P013

P02 crée 2 fils P021 et P022

P03 crée 1 fils P031

P011 crée 2 fils P0111 et P0112

P012 crée 1 fils P0121

P021 crée 1 fils P0211

P0111 crée 1 fils P01111

b) Rép : La lecture du disque étant bloquante, il est plus approprié d'utiliser des threads noyau. Ainsi, le blocage d'un thread Worker ne bloquera pas les autres threads Worker et le Dispatcher. Par contre, s'ils sont implémentés au niveau utilisateur, le blocage d'un thread Worker va bloquer tous les threads Worker et le Dispather.

c) Rép : Si un thread bloque en attente d'un sémaphore noyau, tout le processus est bloqué pour toujours.

d) Rèp : La valeur minimale de m est n+1. La pire configuration est celle où les tâches détiennent chacune 1 ressource et en demande une autre. Avec une ressource de plus, on pourrait satisfaire les demandes d'une tâche qui s'exécutera puis libérera les 2 ressources qu'elle détient. Ce qui permettra aux 2 autres de s'exécuter et ainsi de suite... Pour m=n+1, on pourra toujours satisfaire les besoins en ressources de ces tâches.

e) Rép : Il suffit d'associer à chaque sémaphore compteur deux sémaphores binaires, *M*, pour assurer l'exclusion mutuelle, et *B*, pour

bloquer/débloquer un processus. Il faut également associer un compteur qui indique la valeur courante du sémaphore et un enfin le nombre de processus bloqués, nbwait, en attente du sémaphore:
SemaphoreC S = 10; → SemaphoreB M=1, B=0; int val=10, nbwait=0;
P(S) → P(M); if (val>0) {val--; V(M); } else { nbwait++; V(M); P(B); }
V(S) → P(M); if (nbwait>0) {nbwait--;V(B); V(M); } else { val++; V(M); }

Il suffit de gérer une liste de processus bloqués en attente du sémaphore compteur S et d'associer à chaque processus bloqué un sémaphore binaire. Si un processus appelle P(S) alors que val=0, on crée un sémaphore binaire puis on l'insère en tête ou en queue de liste selon la discipline (FIFO ou LIFO) puis on le bloque sur son sémaphore binaire.
Si un processus appelle V(S) et la liste n'est pas vide, on débloque le processus en tête de liste en appelant V pour son sémaphore.

f) Rép :
Oui, car 1/5 + (25*20)/1000 = 2/10 + 5/10 = 0,7 <1

Question 2 :

a) Semaphore tour =1, acces =3;

```
// tour pour traiter les voitures et les camions selon leur ordre d'arrivée

// acces pour gérer les accès au lave-auto

void acces_voiture ( )

{       P(tour);

        P(acces);

       V(tour);
```

```
}

void sortie_voiture ( )

{     V(acces);

}

void acces_camion ( )

{       P(tour);

        P(acces); P(acces); P(acces); // on a besoin de 3 emplacements

       V(tour);

}

void sortie_camion ( )

{     V(acces); V(acces);  V(acces);

}
```

b) Moniteur Lave-auto

```
{    Boolc acces;  // pour attendre l'acces au lave-auto

  int libre =3;  // nombre d'appareils libres dans le lave-auto

  file q; // file d'attente devant le lave-auto.

// Un élément de q est soit égal à 1 soit égal à 3.

  void acces_voiture ( )

   {      if(libre==0) { enfiler(q,1); wait(acces);}

          libre = libre - 1;
```

```
        }

        void sortie_voiture ( )

        {       libre ++;

                if (teteFile(q) ==1 || (teteFile(q)==3 && libre==3))

               {   defiler(q); signal(acces);  }

        // defiler élimine l'élément en tête de file si la file est non vide….

        }

           void acces_camion ( )

          {     if (libre!=3)

                {   enfiler(q,3);  wait(acces); }

                libre = libre - 3;

        }

        void sortie_camion ( )

        {       libre = 3;

                defiler(q);

               signal(acces);

        }
}
```

c) Scénario 1 :

Voiture 1 P(tour)→ tour=0; P(acces) → acces=2; V(tour)→tour=1;
rentre dans le lave-auto

Camion P(tour)→ tour=0; P(acces)→acces=1; P(acces)→acces=0;
P(acces) va le bloquer

Voiture 2 P(tour) va la bloquer

Voiture 1 sortie du lave-auto; V(acces) va débloquer le camion

Camion V(tour); va débloquer la voiture 2

Voiture 2 P(acces); va bloquer voiture 2;

Camion rentre dans le lave-auto; sort du lave-auto;
V(acces) va débloquer Voiture 2; V(acces)→ acces=1;
V(acces)-.> acces=2;

Voiture 2 V(tour) → tour=1;

Scénario 2 :

Voiture 1 : libre=2; entre au lave-auto

Camion : enfiler(q,3); wait(acces);

Voiture 2 : enfiler(q,1); wait(acces);

Voiture 1 : sort du lave-auto; libre=3; defiler(q); signal(acces);

Camion : libre = 0; entre au lave-auto; sort du lave-auto;
libre=3; defiler(q); signal(acces);

Voiture 2 : libre=2; entre au lave-auto; sort du lave-auto;
libre=3; defiler(q); signal(acces);

Question 3 :

a) P1P1P1P2P2P2P3P3P1P1P4P4P4P5P5P5P2P2P2P1P1P1P4

P1P2P2P2P2P2P2P5P5P5P1P1P1P1P3P3P1P1P1P4P4P4P4

b) Les temps estimés des 3 premières rafales de A sont :
Première : 30; Seconde : (30+20)/2 = 25; Troisième : (25 + 35)/2 =30
Les temps estimés des 4 premières rafales de B sont :
Première : 30; Seconde : (30+40)/2 = 35; Troisième :((35 + 15)/2 =25;
Quatrième : (25 + 25) /2 = 25
L'ordonnanceur va élire B car 25 < 30.