Final SR02
P2017
Durée de l'examen : 2h

H. Lakhlef, L. Zaouche, A. Sahli, D. Nace, A. Lounis

- Répondez sur les feuilles de l'examen. Seules les réponses sur les feuilles de l'examen sont considérées
- Les documents de cours et de TD (en version papier) sont autorisés
- L'examen est constitué de quatre parties indépendantes
- Indiquez votre nom et prénom dans chaque partie

Partie I

Question 1 (1 pt). Pourquoi les caches sont-ils utiles ? Quels problèmes provoquent-ils ?

……..……
……..
……..……
……...
……...
……..……
……..
……..
…….

Question 2 (1 pt). Sur un système de 1Mo de mémoire et qui utilise le système de zones siamoises.

Elaborer un diagramme d'allocations de la mémoire après chacun des événements suivants :

1. Arrivée du processus A, 100 Ko.
2. Arrivée du processus B, 240 Ko.
3. Arrivée du processus C, 64 Ko.
4. Arrivée du processus D, 256 Ko.
5. Sortie du processus B.
6. Sortie du processus A.
7. Arrivée du processus E, 75 Ko.
8. Sortie du processus C.
9. Sortie du processus E.
10. Sortie du processus D.

……..……
……..
…….
…….
……..……
…….
…….
…….
……..……
…….
……
……

Question 3 (1 pt). Considérons un espace d'adressage logique de 32 pages avec 1024 mots par page, avec une mémoire physique de 16 cases.

a) Combien de bits sont requis dans l'adresse logique ?

……
……
……
……

b) Combien de bits sont requis dans l'adresse physique ?

……
……
……
……

Question 4 (1 pt). Que se passe-t-il quand le degré de la multiprogrammation est trop élevé ? Quelle solution vous proposez pour limiter ce problème ?

……
……
……
……

Question 5 (1 pt). Un ordinateur possède une mémoire de 4 pages. Pour chacune des pages, le gestionnaire de mémoire tient à jour les indicateurs suivants : date de chargement, date de dernière référence, rb (bit indiquant si la page a été référencée), mb (bit indiquant si la page a été modifiée). A un instant donné, la situation est la suivante :

Page	Date Chargement	Date Dern. Refer.	rb	mb
0	45	100	1	1
1	22	130	1	0
2	60	155	0	1
3	56	188	1	1

- Indiquer la page qu'il faudra remplacer prochainement dans le contexte de chacune des stratégies suivantes : 1) FIFO, 2) LRU, 3) NRU.

……

……

……

……

……

……

Partie II

Nous considérons une matrice de dimension M×M, dont les lignes seront traitées par N threads (avec N<M). Chaque ligne de la matrice n'est traitée qu'une seule fois. Le programme principal appelle la fonction *create_threads* qui crée N threads exécutant la fonction *traiter_lignes*. Un thread qui exécute *traiter_lignes* reçoit comme argument son numéro d'ordre i (i = 1 .. N). Il récupère alors la prochaine ligne de la matrice qui n'a pas encore été traitée et la traite. Dès qu'il a fini le traitement, le thread récupère une nouvelle ligne, s'il en reste encore. Le traitement fait par un thread se résume à multiplier la ligne récupérée par son numéro d'ordre (numéro du thread). À la fin, le thread renvoi le nombre de lignes qu'il a traités. Le programme principal attend la fin de tous les threads en appelant la fonction *attendre_fin_traitement* avant d'afficher le contenu final de la matrice *mat*. Cette fonction attend la fin de tous les threads en affichant le nombre de lignes traitées par chaque thread. Le squelette du programme est donné ci-dessous. La fonction *initialiser_mat* initialise tous les éléments de la matrice mat à 1. La fonction *afficher_mat* affiche le contenu de la matrice.

```
#define N 3 /* nombre de threads */
#define M 10 /* dimensions de la matrice */
int mat[M][M];
pthread_t tid [N]; /* sauvegarder les identifiants des threads */
void init_mat (void) {
     int i,j ;
     for (i=0; i< M; i++)
          for (j=0; j< M; j++)
               mat[i][j] =1;
}
void afficher_mat (void) {
     int i,j ;
     for (i=0; i< M; i++)  {
          for (j=0; j< M; j++)
               printf ("%d ",mat[i][j]);
          printf ("\n");
     }
}
void *traiter_lignes (void* arg) {
 /* à compléter */
}
void create_threads (int nb_threads, pthread_t * tid) {
 /* à compléter */
}
void attendre_fin_traitement (int nb_threads, pthread_t * tid) {
 /* à compléter */
}
void main (int argc, char ** argv) {
   init_mat (); /* initialiser matrice */
   create_threads (N, tid);
   attendre_fin_traitement (N, tid);
   afficher_mat();/* afficher le contenu final de la matrice*/
}
```

Question 1 (1 pt). Complétez la fonction *create_threads*. Elle doit créer les N threads qui exécuteront *traiter_lignes* en passant comme argument à chaque thread son numéro d'ordre de création i (i = 1 .. N). La fonction reçoit comme paramètres le nombre de threads *nb_threads* à créer et l'adresse de la variable *tid* où les identifiants des threads doivent être sauvegardés.

……
……
……
……
……
……
……
……
……
……..

Question 2 (2 pt). Programmez le corps de la fonction *traiter_lignes*. Spécifiez également la déclaration et l'initialisation des variables globales dont vous avez besoin dans votre solution.

……
……
……
……
……
……
……
……
……
……
……
……
……..

Question 3 (2 pt). Complétez la fonction *attendre_fin_traitement* qui attend la terminaison de tous les threads dans l'ordre de leur création en affichant le nombre de lignes traitées par chacun des threads ainsi que son numéro d'ordre. Le nombre de threads à attendre et leur identifiant respectif sont passés en argument.

……
……
……
……
……
……
……
……
……
……
……
……
……
……
……

Partie III

Question 1 (1 pt). Démontrez formellement que dans le cas où tous les processus sont présents dans la file d'attente des processus prêts au moment où commence l'allocation du CPU, l'algorithme SJF minimise le temps moyen de traitement, pour l'ensemble des algorithmes sans réquisition.

……
……
……
……
……
……
……
……
……
……
……
……

Question 2 (1 pt).

a. Donnez l'ordre d'exécution des processus pour la politique d'ordonnancement SRT avec quantum 1. Calculez le temps de traitement moyen et le temps de traitement maximal et minimal. En cas d'égalité, utilisez la règle FCFS.

	Date d'arrivée	durée
P1	0	5
P2	1	3
P3	2	1
P4	2	2
P5	3	3

……
……
……
……
……
……
……
……

……

……

……

Question 3 (1 pt). Supposons que l'algorithme de Shoshani et Coffman se termine avec plusieurs processus Fini[i]=faux. Démontrez que le système est nécessairement en situation d'interblocage.

……

……

……

……

……

……

……

Question 4 (1 pt). On suppose que 3 processus se partagent 10 exemplaires de ressources d'une classe A, et que la demande maximale Ta pour chaque processus est 4.

a. Est-ce que le système peut se bloquer ? Expliquer.

……

……

……

……

……

……

……

b. En cas d'exécution séquentielle quel serait le nombre d'exemplaires minimal pour garantir l'exécution des processus ?

……

……

……

……

……

……

Question 5 (1 pt). Considérons la situation suivante : Y-a t il un interblocage ? Justifiez.

Allocation

processus	R1	R2	R3
a	1	2	1
b	0	1	1
c	1	4	0
d	0	0	1

Demande

processus	R1	R2	R3
a	2	3	1
b	1	1	3
c	1	2	1
d	1	0	1

Disponible = [1,1,2]

……

……

……...

……

……

……

Partie IV

Soit un téléphérique qui desserve un sommet difficile d'accès. Ce téléphérique dispose d'une cabine qui peut contenir maximum quatre personnes pour chaque trajet effectué. Ce téléphérique peut faire le trajet uniquement si la cabine est pleine. Nous supposons que ce véhicule ne prend des voyageurs que pour l'aller. Une personne à la fois peut rentrer dans la cabine. Après chaque trajet le téléphérique doit pouvoir prendre des nouveaux voyageurs qui attendent dans l'arrêt.

Question 1 (2 pt). Compléter le code du téléphérique et du voyageur en utilisant les sémaphores pour synchroniser leurs activités.

Initialisation :

……
……
……
……
……
……..

Téléphérique	Voyageur
Répéter FaireLeTrajet() jusqu'à faux	 Voyager()

Question 2 (3 pt). Modifier la solution de la question 1 afin que la condition suivante soit satisfaite :

- Si un voyageur arrive dans l'arrêt pendant que la cabine est pleine ou le téléphérique est en trajet :
  - – si un des N sièges du banc d'arrêt est libre, il s'assied et attend,
  - – sinon il repart.

Initialisation :

……

……

……

……

……

……..

Téléphérique	Voyageur
Repeter FaireLeTrajet() jusqu'à faux	 Voyager()