

Contrôle Systèmes d'exploitation, Réseaux

Mercredi 9 Mars 2012
9h - 12h
Aucun document n'est autorisé

Exercice 1 : Ordonnancement de processus (6 = 3 + 3)

On considère les cinq exécutions de processus suivants (la durée est exprimée en seconde) :

Processus	Date d'arrivée	Durée
P1	0	7
P2	1	4
P3	1	2
P4	2	2
P5	3	1

1. Donner les diagrammes de Gantt et les temps de traitement moyen obtenus à l'aide des algorithmes d'ordonnancement FIFO (Premier arrivé - Premier servi), PCTER (plus court temps d'exécution restant) et Tourniquet (avec un quantum de 1) en supposant un temps de commutation de contexte négligeable.
2. Si le temps de commutation est de 0,5 seconde, quel est alors le temps moyen de traitement dans le cas d'un ordonnancement PCTER et d'un ordonnancement tourniquet. Qu'en déduisez-vous ?

Exercice 3 : Parrallélisation (3)

On considère l'ensemble de tâches suivant (qui calcule $(x + y + z)/u * v * (x + y)$) :

t1: $a = x + y$;
t2: $b = u * v$;
t3: $c = a + z$;
t4: $d = a * b$;
t5: $R = c / d$;

Le système de tâches avec l'ordre $t1 < t2 < t3 < t4 < t5$ est-il déterminé ? Si oui, donner le système de parallélisme maximal équivalent et en déduire un programme utilisant les primitives de Conway fork/join/quit.

Exercice 4 : Threads et exclusion mutuelle (8: 1 + 3 + 2 + 2)

On cherche à programmer un logiciel capable de compter le nombre de 'a' dans une chaîne de caractères. Concrètement il s'agit d'écrire un programme ayant les caractéristiques suivantes:

- Une fonction `int compter(char *s)` qui retourne le nombre de 'a' de la chaîne de caractères s.
- Le programme a une donnée globale qui est un tableau de chaîne de caractères nommé `char *requetes[]` de taille 5.
- Le programme crée un thread qui doit exécuter la fonction `void *accueil(void*s)` et 10 threads qui exécutent la fonction `void *executer(void *s)`, puis attend que tous les threads soient terminés.
- La fonction `void *accueil(void*s)` est une boucle infinie qui lit du clavier une chaîne de caractères et la met à la première case libre du tableau `requetes` si le tableau n'est pas plein.
- La fonction `void *executer(void *s)` récupère du tableau `requetes` une chaîne s'il y en a une, appelle la fonction `compter` et affiche le résultat.

1. Ecrire la fonction `compter`.
2. Sans considérer les problèmes d'exclusion mutuelle, écrire les fonctions `accueil` et `executer`.

3. Ecrire le reste du programme: fonction main et variables globales nécessaires.

4. Ajouter les mutex nécessaires au bon fonctionnement.

Fonctions et en-tête utiles:

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<pthread.h>
4
5 int pthread_create(pthread_t *thread,
6                   const pthread_attr_t *attr,
7                   void *(*start_routine)(void*), void *arg)
8 int pthread_join(pthread_t thread, void **value_ptr)
9 void pthread_exit(void *value_ptr)
10
11 int pthread_mutex_lock(pthread_mutex_t *mutex)
12 int pthread_mutex_unlock(pthread_mutex_t *mutex)
13 int pthread_mutex_init(pthread_mutex_t *mutex, const pthread_mutexattr_t *attr)
14 int pthread_mutex_destroy(pthread_mutex_t *mutex)
```