Institut Galilée
Année 2011-2012

Systèmes d'exploitation, Réseaux
Licence 2

Contrôle Systèmes d'exploitation, Réseaux

Mercredi 20 Juin 2012
1h30
Aucun document n'est autorisé

Exercice 1 : Ordonnancement (2)

On supposera que le temps de commutation de contexte est négligeable. Les processus suivants doivent être exécutés sur un ordinateur ayant un seul CPU :

Processus	Date de début d'exécution	Durée supposée d'exécution
P_1	0	9
P_2	4	8

Donner le diagramme de Gantt et le temps moyen de traitement lorsque l'algorithme d'ordonnancement de processus utilisé par le système d'exploitation est la méthode par quantum (on prendra 4 comme durée d'un quantum).

Exercice 2 : Parallélisation et threads (8 = 1 + 1 + 1 + 1 + 2 + 2)

1. Donner un exemple simple de système de tâche ne pouvant pas être décrit avec les primitives parbegin/parend.

   On considère pour la suite de cet exercice le programme utilisant les primitives parbegin/parend donné ci-dessous:

```
begin
  parbegin
    lire(a)
    lire(b)
  parend ;
  parbegin
    c := a*a
    begin
      d := a*b ;
      e := d*a ;
    end
  parend ;
  e := c + d + e
end
```

2. Donner le graphe de flot en précisant les tâches que vous considérez.
3. Indiquer les domaines de lecture et écriture des différentes tâches.
4. Le système est-il de parallélisme maximal ?
5. Donner le programme correspondant de parallélisme maximal en utilisant les primitives de Conway fork/join/quit.
6. Ecrivez le programme précédent à l'aide de threads.

Fonctions et en-tête utiles:

```c
#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>

int pthread_create(pthread_t *thread, const pthread_attr_t *attr,
                   void *(*start_routine)(void*), void *arg)
int pthread_join(pthread_t thread, void **value_ptr)
void pthread_exit(void *value_ptr)
int pthread_mutex_lock(pthread_mutex_t *mutex)
int pthread_mutex_unlock(pthread_mutex_t *mutex)
int pthread_mutex_init(pthread_mutex_t *mutex, const pthread_mutexattr_t *attr)
int pthread_mutex_destroy(pthread_mutex_t *mutex)
```