Institut Galilée
Année 2010-2011

Systèmes d'exploitation, Réseaux
Licence 2

Contrôle Systèmes d'exploitation, Réseaux

Mercredi 16 Mars 2011
9h - 12h
Aucun document n'est autorisé

Exercice 1 : Création de processus (1,5)

1. Lequel des programmes A, B, C suivants crée le plus de processus sur un système de type Unix (expliquer votre réponse en indiquant le comportement de chaque programme lors de son exécution) ? Lequel pourrait causer un blocage du système ?

```
Programme A: int main(int argc, char* argv[]) { while(fork() !=0 ) { } return 0;}
Programme B: int main(int argc, char* argv[]) { while(fork() ==0 ) { } return 0;}
Programme C: int main(int argc, char* argv[]) { while(1) {fork();} return 0; }
```

Exercice 2 : Ordonnancement de processus (7,5 = 1 + 1,5 + 3 + 2)

On considère les sept exécutions de processus suivants :

Processus	Date d'arrivée	Durée
1	0	7
2	0	4
3	1	6
4	1	1
5	1	2
6	2	4
7	2	1

1. Rappeler en quelques lignes ce qu'est la commutation de contexte.
2. Rappeler en quelques lignes ce qui caractérise les ordonnancements à court terme, à moyen terme, à long terme.
3. Donner les diagrammes de Gantt et les temps de traitement moyen obtenus à l'aide des algorithmes d'ordonnancement FIFO (Premier arrivé - Premier servi) et Tourniquet (avec un quantum de 1) en supposant un temps de commutation de contexte négligeable.
4. Avec un ordonnancement par tourniquet, donner le temps d'attente entre 2 passages en exécution d'un processus en fonction du temps de commutation et du nombre de processus prêts. Expliquer votre résultat.

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:

```
#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)
```