Systèmes d'exploitation, Réseaux

Vendredi 1 Juillet 2011
1h30
Aucun document n'est autorisé

On s'efforcera de donner des explications suffisantes pour chacune des réponses.

Exercice 1 (2 = 1+1)

On considère 5 programmes P_1, P_2, P_3, P_4, P_5 de début d'exécution respectif aux temps $t_0, t_0+1, t_0+3, t_0+5, t_0+7$ et de durée d'exécution respectif $4, 4, 2, 4, 2$. Donner les diagrammes de Gantt pour des ordonnancements de type FIFO (premier arrivé premier servi) et à tourniquet (pour un quantum de 2).

Exercice 2 (4 = 2 + 2)

Fonctions pouvant être utiles :

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <signal.h>
#include <sys/wait.h>
#include <pthread.h>

pid_t getpid(void); // PID du processus
pid_t getppid(void); // PID du parent du processus
pid_t fork(void); // création de processus

int pthread_create(pthread_t *thread, pthread_attr_t *attr, void *(*start_routine)(void *), void *arg);
     // création d'un thread
void pthread_exit(void *value_ptr); // fin d'un thread
int pthread_join(pthread_t thread, void **value_ptr); // suspension pour synchronisation

void (*signal(numsig, void (*fonction) (int))) (int); // association d'une fonction à un num. de signal
int raise(int numsig); // envoi d'un signal au processus courant

int open(char *chemin, int mode); // ouverture d'un fichier
int close (int fd); // fermeture
int read(int fd, char *tab, int nbcar); // lecture de données
int write(int fd, char *tab, int nbcar); // écriture de données
int pipe(int p[2]); // création d'un tube

pid_t wait(int *pointer_status); // attente du changement d'état d'un processus fils
int kill(int pid, int numsig); // envoi d'un signal à un processus
int sleep(int temps); // attente sur le processus courant
```

1. Ecrivez un programme dans lequel le processus père crée 2 fils. Chaque fils attend 5 secondes, puis envoie au père le signal `SIGUSR1`, et les processus fils terminent. Le père termine lorsqu'il a reçu un des deux signaux.
2. En utilisant le mécanisme des tubes, modifiez le programme de telle sorte que chaque fils, après 5 secondes, envoie son pid par un tube au père, et que celui-ci affiche alors le pid reçu.

Exercice 3 (4 = 1 + 1 + 0.5 + 0.5 + 1)

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 dans l'annexe A.

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.

Annexe A

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