IUT de Villetaneuse — Année 98 - 99

Département Informatique — Système 2ème année

TD - TP 1

Processus

Semaine du 14/9/98

Objectifs : Etudier la notion de processus (multitâche, création, etc.)

Rappels

Un *programme* est l'expression d'un algorithme à l'aide d'un langage de programmation. Une *tâche* est la conjonction d'un programme et des données auxquelles il s'applique. Un*processus* est un exemplaire d'un programe en cours d'exécution (c'est donc aussi une tâche en cours d'exécution). Il est caractérisé par la tâche et son contexte d'exécution, c.à.d. la valeur du compteur ordinal, les valeurs des registres etc.

En *monoprogrammation*, il y a un seul processus à la fois en mémoire. Lorsqu'une tâche est soumise et que le processeur est disponible, on la charge en mémoire puis on exécute le processus associé jusqu'à ce qu'il soit terminé. On passe alors à la tâche suivante.

En *multiprogrammation*, il peut y avoir plusieurs processus à la fois en mémoire. Une tâche soumise est chargée en mémoire s'il y a de la place et donne naissance à un processus. Un processus est prêt s'il n'est pas en attente d'une entrée-sortie. Les processus prêts s'exécutent à tour de rôle, on parle de commutation de processus. La multiprogrammation peut être utilisée en batch ou en temps partagé.

En *batch (multi-programmation non préhemptive)*, il n'y a commutation que si le processus actif doit effectuer une entrée-sortie.

En *temps partagé*, il y a commutation si le processus actif doit effectuer une entrée-sortie ou s'il a épuisé son quantum.

Un *quantum* est une durée élémentaire (de l'ordre de 10 à 100 ms).

L'*overhead* est le temps passé (perdu) à effectuer une commutation entre deux processus en temps partagé.

Une *unité d'échange* sert à effectuer des transferts entre la mémoire principale et un périphérique sans utiliser le processeur (DMA). Elle reçoit un ordre du processeur, effectue le transfert, puis avise le processeur de la fin de l'échange. Durant le transfert, le processeur peut faire d'autres calculs.

1. Etude du taux d'occupation

Considérons deux tâches identiques T_1 et T_2 effectuant n fois le traitement suivant :

- Lecture d'une donnée sur le disque (durée l)
- Opération de calcul sur la donnée (durée c)
- Ecriture du résultat sur le disque (durée e)

1.1) La tâche T_1 est soumise seule. Représentez sur un diagramme des temps l'exécution de la tâche T_1 en monoprogrammation. Calculez le temps total d'exécution, le taux d'occupation du processeur et de l'unité d'échange.

1.2) Les tâches T_1 et T_2 sont soumises simultanément. Représentez sur un diagramme des temps l'exécution des tâches T_1 et T_2 en monoprogrammation. Calculez le temps total d'exécution, le taux d'occupation du processeur et de l'unité d'échange. Application numérique : l = e = 7, c = 6, n = 4.

1.3) Répondez à la question précédente en supposant une multiprogrammation en mode batch.

Y a-t-il un intérêt à la multiprogrammation en mode batch si l'ordinateur ne dispose pas d'une unité d'échange?

2. Influence de l'overhead et du quantum.

On reprend le problème précédent en supposant qu'il y a un overhead de s secondes à chaque commutation.

2.1) Tous les utilisateurs commencent à travailler en même temps. Exprimez les temps de réponse r @&&et R en fonction des autres paramètres en régime permanent. Calculez r et R pour n = 10, c = 0,1 s, N = 5, C = 3 s, s = 1 ms et q = 100 ms; puis pour q = 10 ms.

2.2) Quels sont les critères de choix du quantum?

3. Éxécution simultannée de pocessus

Copiez le programme `~levy/slowprint1` dans votre répertoire de travail.

3.1) Exécutez : `slowprint1`, puis `slowprint1 a`. Essayez d'intrompre le programme avant la fin par ^C.

3.2) Exécutez : `slowprint1 a 10 ; slowpint1 b 10.`

Comparez avec `slowprint1 a 10 &; slowpint1 b 10.`

Quel est le temps écoulé dans les deux cas ? Que se passe-t-il maintenant si vous interompez par ^C ? Expliquez.

3.3) Immédiatement après avoir lancé : `slowprint1 a 10 &; slowpint1 b 10`,
tapez `ps` (liste des processus). Refaites la même chose, mais en lançant
`slowprint1 a 10 &; slowpint1 b 10 &`
Expliquez. Avec la même commande, essayez ^C. Essayez aussi `fg` (foreground).

4. Création de processus en C

Soit le programme suivant **:**

```
/* Fichier pere_fils1.c */
#include <unistd.h>
#define   N    .........

void main ()
{ int  i;
pid_t       pid, ppid, npid;
pid = (pid_t) getpid();
printf('' Debut du processus n° : %d  \n'', pid );
for ( i = 1 ; i < N ; i++ ) {
      npid = ( pid_t ) fork();
      pid  = (pid_t) getpid();
      ppid  = (pid_t) getppid();
      printf(''Le processus est %d , le père  est : %d'', pid,ppid) ;
      }
printf('' \t  \t  \t  Fin du processus n° : %d  \n'', pid );
}
```

Combien de processus sont créés pour N=2, N=3, N=4, ... ? Donner l'arborescence des processus.