IUT de Villetaneuse — Année 99-00
Département Informatique — Système 2ème année

TD 1
Processus, ordonnancement
Semaine du 14/9/98

Objectifs : Etudier le partage du processeur entre différents processus.

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 programme 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 mode non préemptif ou préemptif (temps partagé).

En *multi-programmation non préemptive*, 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 donc le temps de calcul maximal alloué avant préemption (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. C'est le temps nécessaire au commutateur pour sauvegarder un contexte et en recharger un autre.

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 non préemptif.

Y a-t-il un intérêt à la multiprogrammation en mode non préemptif 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 mode préemptif. Le quantum est **q** et il y a un overhead de **s** secondes à chaque commutation. On a maintenant **N** utilisateurs travaillant simultanément (donc pour simplifier N tâches). On considère des tâches sans entrée-sortie (l=e=0).

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

2.2) Quels sont les critères de choix du quantum?

3. Influence de l'ordonnancement.

On se propose d'étudier analytiquement l'influence des différentes politiques de gestion d'un processeur. On dispose d'un ordinateur ayant une unité d'échange travaillant en parallèle avec la CPU. On suppose, *dans un premier temps* , qu'il y a suffisamment de mémoire pour contenir tous les processus.

L'allocation du processeur, consiste à choisir pour (au plus) un quantum de temps, parmi un ensemble de tâches prêtes, celle qui va occuper le processeur. En anglais on parle de "scheduling".

3.1) **L'algorithme d'ordonnancement circulaire** (round robin) consiste à ranger les tâches dans une file unique. Le processeur est donné à la **première** tâche **prête** de la file. La tâche perd le processeur en cas d'entrée/sortie ou quand elle a épuisé son quantum de temps (q). Elle est alors mise en fin de la file d'attente des tâches. Si une tâche arrive au début de la file dans l'état "attente" (attente d'une E/S), elle reste en début de file. Toute nouvelle tâche est mise à la fin de la file. On considère les tâches suivantes :

	Temps CPU	**E/S**	**durée E/S**
T1	400 ms	aucune	
T2	40 ms	ttes les 10 ms	250 ms
T3	300 ms	aucune	
T4	60 ms	ttes les 20 ms	180 ms

Décrire précisément l'évolution du système : instant, nature de l'événement (commutation, demande d'E/S, fin d'E/S), tâche élue, état de la file. Donner pour chacune des tâches l'instant où elle termine son exécution. On prendra un quantum de 100 ms, et on supposera qu'elles sont initialement prêtes et rangées dans l'ordre de leur numéro. Quels sont les avantages et les inconvénients de cette technique?

3.2) **Priorités statiques :** Chaque tâche à sa création dispose d'une priorité, qu'elle conserve tout au long de son exécution. En supposant qu'il existe 4 niveaux de priorité numérotés de 0 à 3 (0 étant la plus faible priorité), reprendre la question précédente pour cet algorithme en considérant les tâches suivantes :

	Temps CPU	E/S	durée E/S	Priorité
T1	400 ms	aucune		0
T2	40 ms	ttes les 10 ms	250 ms	3
T3	300 ms	aucune		1
T4	60 ms	ttes les 20 ms	180 ms	3
T5	400 ms	aucune		1

3.3) **Priorités dynamiques:** Les priorités affectées aux tâches changent en cours d'exécution. En gros, quand un processus a attendu trop longtemps (suivant un critère à choisir), sa priorité augmente ; quand il a reçu suffisament de temps de calcul (suivant un critère à choisir), sa priorité diminue (suivant une règle à choisir). Plusieurs algorithmes sont possibles suivant le critère de changement de priorité, la règle de diminution et les variations de quantum selon la priorité .

Donnez 3 exemples de choix des paramètres (critère de changement de priorité, règle de diminution, variations de quantum) pour un système à 5 files d'attente.

3.4) Trouver un algorithme où les tâches n'utilisant pas tout leur quantum de temps sont favorisées. Trouver un exemple qui illustre ce phénomène.