Série d'exercices #6

IFT-2245

12 février 2019

6.1 Préemption

Expliquer la différence entre ordonnancement préemptif et non-préemptif

6.2 Simulation

Soit les tâches suivantes :

Tâche	Burst time	Priorité
P1	2	2
P2	1	1
P3	8	4
P4	4	2
P5	5	3

Présumer qu'elles sont toutes arrivées au temps 0, dans l'ordre indiqué.

— Illustrer dans des tableau de Gantt l'exécution qui en découle dans le cas d'un ordonnancement de type RR, SJF, FCFS, et priorité non-préemptif.

— Pour chacun des cas, calculer le *waiting time* moyen et le *turnaround time* moyen.

6.3 Durée du quantum

Donner les avantages et inconvénients d'avoir un quantum plus court ou plus long. Expliquer quel est l'avantage d'avoir des quantum différent pour les différentes queues d'un ordonnanceur à plusieurs niveaux.

6.4 RR mutiple

Soit une variante de RR dans laquelle les entrées dans la queue des processus prêts à l'exécution sont des pointeurs vers des PCBs :

1. Quel serait l'effet de placer deux pointeurs vers le même PCB dans la queue ?
2. Quels seraient les avantages et inconvénients de cet usage ?
3. Comment modifier l'algorithme RR de base de manière à obtenir le même résultat mais sans avoir besoin de placer plusieurs copies du même pointeur.

6.5 Quantums

Soit un système avec 10 tâches *I/O-bound* et une tâche qui n'utilise pas les périphériques (et donc *CPU-bound*). Chaque tâche *I/O-bound* lance une opération d'entrée/sortie après chaque milliseconde de calcul qui prend 10ms pour se terminer. Soit un coût de 0.1ms pour chaque changement de contexte. Si on présume que toutes les tâches s'exécutent indéfiniment, décrire l'utilisation du CPU dans le cas d'un ordonnancement de type *round-robin* :

1. Avec un quantum de temps de 1ms.
2. Avec un quantum de temps de 10ms.

6.6 Multiple niveaux

Soit un algorithme de queues à niveaux multiples, où un processus passe à la queue de priorité supérieure s'il relâche le CPU avant la fin de son *quantum* et passe à la queue de priorité inférieure dans le cas inverse. Quels genre de comportement peut-on attendre d'un tel algorithme, en présence de processus variés, certains utilisant beaucoup le CPU d'autres seulement les périphériques, d'autres un mélange des deux ?

6.7 Priorités dynamiques

Soit un algorithme d'ordonnancement préemptif basé sur des priorités dynamiques. Quand un processus est en attente du CPU, sa priorité augment à un rythme α ; quand il est en cours d'exécution, sa priorité augmente à un rythme β. Tous les processus commencent avec une priorité de 0. Les paramètres α et β peuvent être choisis pour obtenir différents résultats :

1. Quel algorithme obtient-on si $\beta > \alpha > 0$?
2. Quel algorithme obtient-on si $\alpha < \beta < 0$?