EX02 – Processus / Ordonanceur

Systèmes d'exploitation / Classes T-2adfg

– Solutions –

1. Pour quelle raison un «thread» céderait-il volontairement le processeur en appelant «thread_yield» ? Après tout, étant donné qu'il n'y a pas d'interruption d'horloge périodique, il risque de ne plus jamais récupérer de temps processeur.

   Solution : Les threads d'un processus coopèrent. Ils ne sont pas hostiles les uns envers les autres. Si pour le bien de l'application, une conciliation est nécessaire, le thread cède. Après tout, c'est souvent le même programmeur qui écrit le code de tous les threads.

2. Est-il possible d'évaluer si un processus est dépendant du CPU[1] ou des I/O[2] en analysant son code source ? Comment peut-on procéder à une telle évaluation en cours d'exécution ?

   Solution : Dans les cas les plus simples, il est possible de déterminer si les E/S seront limitatives en examinent le code source. Par exemple, un programme qui lit tous ses fichiers d'entrée dans les tampons au départ, ne sera probablement pas dépendant des E/S, mais un programme qui lit et écrit de manière incrémentielle dans un certain nombre de fichiers (comme un compilateur) risque d'être dépendant des E/S.

   Si le système d'exploitation propose un utilitaire comme la commande UNIX «`ps`» qui indique la quantité de temps CPU utilisée par un programme, on peut comparer ce chiffre au temps total nécessaire à l'exécution complète du programme. Ceci est, bien entendu, plus significatif sur un système où vous êtes le seul utilisateur.

3. L'ordonnancement peut être parfois amélioré si un processus important peut jouer un rôle dans la sélection du prochain processus à exécuter en cas de blocage. Présentez une situation dans laquelle exploiter un tel comportement, et expliquez comment faire.

   Solution : Pour plusieurs processus présents dans un pipeline, le parent commun pourrait passer au système d'exploitation des informations concernant le flot des données. Avec ces informations, le système d'exploitation peut, par exemple, déterminer le processus capable de fournir la sortie d'un processus bloquant sur un appel en entrée.

4. Cinq jobs sont en attente d'exécution. Leurs délais d'exécution sont respectivement de 9, 6, 3, 5 et X minutes. Dans quel ordre faut-il les exécuter pour réduire le temps de réponse moyen ? Votre réponse dépendra de X.

   Solution : En commençant toujours par le job le plus court pour minimiser le temps de réponse moyen :

[1] «Central Processing Unit» ou «Unité Centrale» en français
[2] «Input/Output» ou «Entrées/Sorties» en français

$0 < X \leq 3 \quad \rightarrow X, 3, 5, 6, 9$
$3 < X \leq 5 \quad \rightarrow 3, X, 5, 6, 9$
$5 < X \leq 6 \quad \rightarrow 3, 5, X, 6, 9$
$6 < X \leq 9 \quad \rightarrow 3, 5, 6, X, 9$
$X > 9 \quad \rightarrow 3, 5, 6, 9, X$

5. Cinq jobs de traitement par lots, de A à E, arrivent dans un centre informatique pratiquement au même instant. Leurs délais d'exécution respectifs sont de 10, 6, 2, 4 et 8 minutes. Leurs priorités respectives (déterminées extérieurement) sont de 3, 5, 2, 1 et 4 (5 étant la priorité la plus élevée). Pour chacun des algorithmes d'ordonnancement suivants, déterminez le délai de rotation moyen des processus. Ignorez la surcharge engendrée par les changements de processus.

   (a) Round-robin.

   Solution : Durant les 10 premières minutes, chaque job obtient $1/5$ du processeur. À la fin des 10 minutes, *C* se termine. Pendant les 8 minutes suivantes, chaque job obtient $1/4$ du processeur ; après ce délai *D* se termine. Ensuite, chacun des trois jobs restants obtient $1/3$ du processeur pendant 6 minutes, jusqu'à ce que *B* se termine, etc. Les temps d'exécution des cinq jobs sont 10, 18, 24, 28 et 30, pour une moyenne de **22 minutes**.

   (b) Ordonnancement par priorités.

   Solution : *B* s'exécute en premier et il se termine 6 minutes plus tard. Les autres jobs se terminent à 14, 24, 26 et 30, pour une moyenne de **20 minutes**.

   (c) Premier arrivé, premier servi (exécutez dans l'ordre 10, 6, 2, 4, 8).

   Solution : Les jobs se terminent à 10, 16, 18, 22 et 30 pour une moyenne de **19.2 minutes**.

   (d) Job le plus court en premier.

   Solution : Les temps produits sont 2, 6, 12, 20 et 30, pour une moyenne de **14 minutes**.

   Pour (a), partez du principe que le système est multiprogrammé et que chaque job récupère une partie équitable de temps processeur. Pour (b) à (d), supposez qu'un seul job est exécuté à la fois, jusqu'à ce qu'il se termine. Tous les jobs sont entièrement dépendants du CPU.

6. L'algorithme de vieillissement (avec $a = 1/2$) est utilisé pour estimer des délais d'exécution. Les quatre premières exécutions, de la plus ancienne a la plus récente, ont été de 40, 20, 40 et 15 ms. Quelle va être l'estimation du prochain délai d'exécution ?

   Solution : La séquence des prédictions est 40, 30, 35 et **25** ms.