1 - MULTI-PROGRAMMATION

1. INTRODUCTION

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 l'exécution d'une tâche. Il est caractérisé par la tâche et son contexte d'exécution, c'est-à-dire 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'un événement extérieur (fin d'entrée-sortie, libération de ressource, etc.). 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*, 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 est en attente d'un événement 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.

On se propose d'étudier analytiquement l'influence des différentes politiques de gestion d'un processeur. On considère un ordinateur doté d'un processeur principal et d'une unité d'échange effectuant les entrées-sorties. On suppose qu'il y a suffisamment de mémoire pour contenir tous les processus.

1.1.

Représentez sous forme de diagramme d'états (noeuds = états d'une tâche, arcs = actions ou événements) l'exécution d'une tâche en mode batch puis en temps partagé.

2. ETUDE DU TAUX D'OCCUPATION

On s'intéresse à l'exécution de trois tâches T_1, T_2 et T3 :

T1 dure 200 ms (hors entrée / sortie) et réalise une unique E / S au bout de 110 ms ;

T2 dure 50 ms et réalise une unique entrée / sortie au bout de 5 ms ;

T3 dure 120 ms sans faire d'entrée / sortie

T1 et T2 sont créées à l'instant 0, T3 est créée à l'instant 140. Chaque entrée / sortie dure 10 ms. La valeur du quantum (pour le temps partagé uniquement) est de 100 ms.

2.1.

Représentez sur un diagramme de Gantt l'exécution des 3 tâches en mode batch et en temps partagé. Calculez le temps total d'exécution T et le taux d'occupation du processeur τ_P.

On considère maintenant deux tâches identiques T_1 et T_2, soumises simultanément à l'instant t = 0 et 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)

2.2.

La tâche T_1 est soumise seule à t = 0. Représentez sur un diagramme de Gantt l'exécution de la tâche T_1. Calculez le temps total d'exécution T, le taux d'occupation du processeur τ_P et de l'unité d'échange τ_U.

Que peut-on en déduire pour T, τ_P et τ_U lorsque les tâches T_1 et T_2 sont exécutées en monoprogrammation ? Application numérique : e = 5 ms, c = 7 ms, l = 8 ms, n = 4.

2.3.

Représentez sur un diagramme de Gantt l'exécution des tâches T_1 et T_2 en supposant maintenant une multiprogrammation en mode batch. On supposera c < l et c > e. Calculez T, τ_P et τ_U avec les mêmes valeurs numériques qu'à la question précédente et comparez les résultats.

2.4.

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

3. TEMPS DE REPONSE EN MODE BATCH

On suppose qu'il y a n utilisateurs exécutant des commandes d'édition. Chaque commande consomme c secondes de temps processeur. Chaque utilisateur attend r secondes (temps de réponse) entre l'instant où il soumet une commande et l'instant où il obtient la réponse, il s'écoule ensuite t secondes (réflexion et frappe) avant qu'il soumette la commande suivante.

3.1.

Tous les utilisateurs commencent à travailler en même temps. Exprimez le temps de réponse en fonction des autres paramètres en régime permanent. Calculez r pour n = 10, t = 2 ms et c = 0,1 ms ; puis pour n = 30, t = 2 ms et c = 0,1 ms.

On suppose qu'il y a une autre classe d'utilisateurs effectuant des compilations. On note N, C, R, T les paramètres correspondant à cette classe. On suppose t = T, t <= (n-1)c+NC et T <= nc+(N-1)C.

3.2.

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. Montrez que r = R. Calculez r pour n = 10, t = 2 ms, c = 0,1 ms, N = 5 et C = 3 ms.

3.3.

Est-il raisonnable d'envisager un travail interactif en mode batch?

4. TEMPS DE REPONSE EN TEMPS PARTAGE

Dans cette partie, on note q le quantum et on suppose qu'il n'y a pas d'overhead. On considère à nouveau n utilisateurs effectuant des commandes d'édition et N utilisateurs effectuant des compilations. Pour simplifier l'analyse, on suppose que t = T = 0 et que c et C sont multiples de q.

4.1.

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 ms, N = 5 et C = 3 ms.

4.2.

Conclusion ?