CHAPITRE 3 :

L'ORDONNANCEMENT DES PROCESSUS

Objectifs spécifiques

- Comprendre la problématique du multitâche
- Connaître la notion d'ordonnancement des processus et l'utilité d'un ordonnanceur
- Connaître les différents critères d'ordonnancement
- Connaître les algorithmes d'ordonnancement préemptifs
- Connaître les algorithmes d'ordonnancement non préemptifs

Eléments de contenu

I. Multitâche et ordonnancement des processus

II. Critères et types d'ordonnancement

III. Algorithmes d'ordonnancement

Volume Horaire :

Cours : 1 heure 30

TD : 3 heure

3.1 Introduction

Un ordinateur possède forcément plusieurs processus en concurrence pour l'obtention du temps processeur, cette situation se produit lorsque 2 ou plusieurs processus sont en état prêt simultanément. L'Ordonnanceur (planificateur, scheduler) est la partie (un programme) du système d'exploitation responsable de régler les états des processus (Prêt, Actif,…etc.) et de gérer les transitions entre ces états ; c'est l'allocateur du processeur aux différent processus, il alloue le processeur au processus en tête de file des Prêts.

3.2 Objectifs d'un Ordonnanceur

Les objectifs d'un Ordonnanceur sont :

- Maximiser l'utilisation du processeur
- Présenter un temps de réponse acceptable
- Respecter l'équité entre les processus selon le critère d'ordonnancement utilisé.

3.3 Critères d'ordonnancement

L'objectif d'un algorithme d'ordonnancement consiste à identifier le processus qui conduira à la meilleure performance possible du système. Certes, il s'agit là d'une évaluation subjective dans laquelle entrent en compte différents critères à l'importance relative variable. La politique d'ordonnancement détermine l'importance de chaque critère. Un certain nombre d'algorithmes ont fait leur preuve dans la mise en œuvre d'une politique d'ordonnancement.

La liste qui suit passe en revue des critères d'ordonnancement fréquemment utilisés.

- **Utilisation de l'UC :** Pourcentage de temps pendant lequel l'UC exécute un processus. L'importance de ce critère varie généralement en fonction du degré de partage du système.
- **Utilisation répartie :** Pourcentage du temps pendant lequel est utilisé l'ensemble des ressources (outre l'UC, mémoire, périphérique d'E/S…)
- **Débit :** Nombre de processus pouvant être exécutés par le système sur une période de temps donnée.
- **Temps de rotation :** durée moyenne qu'il faut pour qu'un processus s'exécute. Le temps de rotation d'un processus comprend tout le temps que celui-ci passe dans le système. Il est inversement proportionnel au débit.
- **Temps d'attente :** durée moyenne qu'un processus passe à attendre. Mesurer la performance par le temps de rotation présente un inconvénient : Le temps de production du processus accroît le temps de rotation ; Le temps d'attente représente donc une mesure plus précise de la performance.
- **Temps de réponse :** Temps moyen qu'il faut au système pour commencer à répondre aux entrées de l'utilisateur.
- **Equité :** degré auquel tous les processus reçoivent une chance égale de s'exécuter.
- **Priorités :** attribue un traitement préférentiel aux processus dont le niveau de priorité est supérieur.

3.4 Types d'ordonnancement

Il existe 2 types d'ordonnancement

a. ***Ordonnancement préemptif :*** Avec réquisition où l'Ordonnanceur peut interrompre un processus en cours d'exécution si un nouveau processus de priorité plus élevée est inséré dans la file des Prêts.
b. ***Ordonnancement coopératif :*** Ordonnancement jusqu'à achèvement : le processus élu garde le contrôle jusqu'à épuisement du temps qui lui a été alloué même si des processus plus prioritaires ont atteint la liste des Prêts

3.5 Les algorithmes d'ordonnancement

a. **L'algorithme FIFO (**First In First Out)

L'ordonnancement est fait dans l'ordre d'arrivée en gérant une file unique des processus sans priorité ni réquisition : chaque processus s'exécute jusqu'à son terme ; le processus élu est celui qui est en tête de liste des Prêts : le premier arrivé. Cet algorithme est facile à implanter, mais il est loin d'optimiser le temps de traitement moyen

b. L'algorithme SJF (Shortest Job First)

L'ordonnancement par ordre inverse du temps d'exécution (supposé connu à l'avance) : lorsque plusieurs travaux d'égale importance se trouvent dans une file, l'Ordonnanceur élit le plus court d'abord (les travaux les plus cours étant en tête de la file des prêts).

Cet algorithme possède l'inconvénient de la nécessité de connaissance du temps de service à priori et le risque de privation des tâches les plus longues. Afin de résoudre ces problèmes on pourra attribuer aux travaux une priorité croissante avec leur temps de séjour dans la file (temps d'attente). A temps d'attente égale, le travail le plus court est prioritaire. Toutefois, cette solution nécessite le calcul des priorités périodiquement et un réarrangement de la FA.

c. L'algorithme du temps restant le plus court (SRT : Shortest Remaining Time)

L'algorithme du temps restant le plus court, est la version préemptive de l'algorithme SJF. Chaque fois qu'un nouveau processus est introduit dans la file des processus à ordonnancer, l'Ordonnanceur compare la valeur estimée du temps de traitement restant à celle du processus en cours d'ordonnancement. Si le temps de traitement du nouveau processus est inférieur, le processus en cours d'ordonnancement est préempte. Tout comme l'algorithme SJF, l'algorithme SRT favorise les travaux courts : les travaux longs en revanche peuvent être victimes de famine.

d. L'algorithme Round Robin

Il s'agit d'un algorithme ancien, simple et fiable. Le processeur gère une liste circulaire de processus. Chaque processus dispose d'un quantum de temps pendant lequel il est autorisé à s'exécuter. Si le processus actif se bloque ou s'achève avant la fin de son quantum, le processeur est immédiatement alloué à un autre processus. Si le quantum s'achève avant la fin du processus, le processeur est alloué au processus suivant dans la liste et le processus précédent se trouve ainsi en queue de liste.

La commutation de processus (overhead) dure un temps non nul pour la mise à jour des tables, la sauvegarde des registres. Un quantum trop petit provoque trop de commutations de processus et abaisse l'efficacité du processeur. Un quantum trop grand augmente le temps de réponse en mode interactif. On utilise souvent un quantum de l'ordre de 100 ms.

e. L'algorithme HPF(Highest Priority First)

Le modèle du tourniquet suppose tous les processus d'égale importance. C'est irréaliste. D'où l'attribution de priorité à chaque processus. L'Ordonnanceur lance le processus prêt de priorité la plus

élevée. En cas de priorités égales on utilise l'algorithme FIFO. L'ordonnancement des priorités peut être préemptif ou non.

Les mécanismes d'attribution des priorités sont très variables ; la priorité est basée sur la caractéristique du processus (utilisation de la mémoire, fréquence des E/S), sur l'utilisateur qui exécute le processus, les coûts d'utilisation (Le temps de l'UC pour les tâches de priorité supérieure est par exemple plus coûteux) ou sur un paramètre que l'utilisateur peut spécifier. Certains mécanismes produisent des priorités qui varient de manière dynamique : volume du temps d'exécution ; alors que d'autre sont statiques (la priorité associée à un utilisateur).

f. Les files d'attente Rétroactives

Cet algorithme met en œuvre N files (N>1) d'attentes classées de 1 à N, un processus qui vient d'être crée est placée dans la file du niveau le plus élevé, une fois sélectionnés les processus de la file reçoivent une tranche de temps relativement courte. A l'expiration de cette dernière, le processus est déplacé dans la file de niveau inférieur. Les tranches de temps associées aux files s'allongent au fur et à mesure que le niveau décroît. Si un processus est bloqué avant d'avoir utilisé la totalité de sa tranche de temps, il passe à la file de niveau supérieur. Le processus sélectionné par cet algorithme est le prochain processus de la file la plus élevée contenant des processus. La sélection au sein d'une file se fait suivant la stratégie FIFO. Si un processus entre dans une file alors qu'un autre processus d'un niveau inférieur est en cours d'exécution, ce dernier peut être préempté.

3.6 Performance des algorithmes d'Ordonnancement

Les performances d'un algorithme pour un ensemble de processus donné peut être analysée si les informations appropriées relatives aux processus sont fournies. Par exemple, des données sur l'arrivée du processus et sur l'heure d'exécution de ce processus sont nécessaires pour évaluer l'algorithme SRT.

Temps de rotation=Temps fin d'exécution - Temps d'arrivée

Temps d'attente=Temps de rotation – Durée d'exécution

Temps moyen d'attente= $\frac{\sum Temps\ attente}{nbre\ de\ processus}$

Rendement= $\frac{\sum Temps\ d'exécution}{nbre\ de\ processus}$