Université Assane Seck de Ziguinchor

Système d'Exploitation

LES ALGORITHMES D'ORDONNANCEMENT DES PROCESSUS

SOMMAIRE

I. INTRODUCTION

Le SE d'un ordinateur gère plusieurs processus à la fois. L'efficacité théorique serait maximale si le nombre de processeurs était comparable à celui des processus. Le fait d'avoir plusieurs processeur peut impacter sur le prix de l'ordinateur. Il faudra alors faire le compromis entre performance et prix. Souvent une machine possède un seul processeur alors qu'en multiprogrammation, on a plusieurs processus encours d'exécution. Il ne faut pas perdre de vue que quand plusieurs processus sont encours d'exécution, un seul est à l'état d'exécution. Un problem possible est que plusieurs processus se partagent un seul processeur. Donc il faudra définir définir une politique d'accès ou politique d'ordonnancement au processeur. L'ordonnancement définit des critères selon lesquels les processus ont accès au processeur. Comme par exemple un carrefour routier partagé entre usagers (véhcules), les guichets d'une administration etc. Des

critères d'ordonnancement doivent être defines dans le but d'optimiser l'utilisation de l'UCT et de répondre aux besoins et priorités des applications. Dans tous les cas, une file d'attente va se former autour du processeur. Ainsi un processus pourra être alloué au processeur quand il sera plus prioritaire de la file d'attente, mais on peut également lui retirer le processeur à l'arrivée d'un processus prioritaire, et dans ce cas il retourne dans la file d'attente. Ce qu'on peut représenter par le schema d'ordonnancement de la **Figure1**.

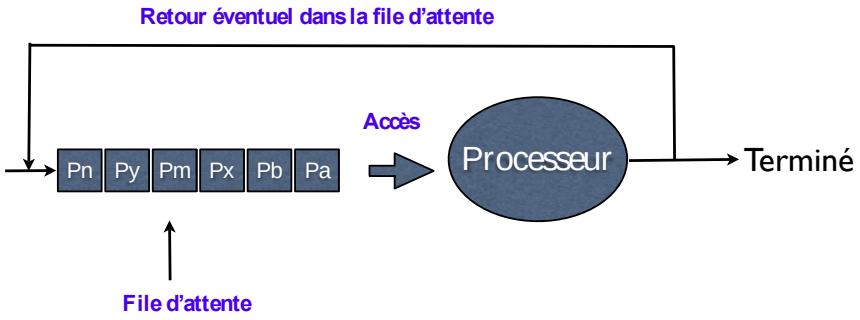

Figure1: Schéma d'ordonnacement

II. CRITERES D'ORDONNANCEMENT

II.1. L'optimisation des ressources

Pour optimiser l'utilisation de la ressource principale qui est le processeur, certains paramètres doivent être maximisé ou minimiser :

- Le pourcentage d'utilisation du processeur est à maximiser
- le débit ou le nombre de processus executes par unités de temps de temps est à maximiser
- Le temps de rotation qui est égal au temps de terminaison - temps d'arrivée (attente incluses) est à maximiser
- Minimiser le temps d'attente (temps passé dans la file d'attente)
- Minimiser le temps de réponse (pour les système interactif): temps entre une demande et première réponse

II.2. Priorité des utilisateurs

Les besoins des utilisateurs sont exprimés sous forme d'applications sur l'ordinateur. Ainsi la priorité des utilisateurs est liée à la priorité des applications. On peut noter trois paramètres importants et fondamentaux comme indicateurs de priorité.

- **L'Ordre d'arrivée**: le premier arrivé est le premier servi (caisses au RestoU, au supermarché, à la poste etc.
- **L'urgence :** le premier servi est celui dont le besoin d'accès rapide à la ressource est le plus grand (pompier)
- **L'importance :** le premier servi est celui dont l'accès à la ressource est le plus important (personne âgée dans les transports en commun)

II.3. Pénalisation

Compte tenu de l'indicateur de priorité utilisé par le système, certains processus peuvent se voir pénaliser. La pénalité d'un processus est liée à son temps d'attente, c'est le nombre d'unités de temps durant lesquelles le processus est présent dans la file d'attente (sans être exécuté). On peut mesurer la pénalité T en fonction du temps d'attente a et du temps d'exécution e par: $T = e/(e+a)$. Dans le cas idéal T=1=e/e, donc a=0

II.3. Types d'ordonnacement

On distingue généralement deux types d'ordonnacement : l'ordonnacement non prémptif (sans réquisition) et l'ordonnacement préemptif (avec requision).

En **Ordonnancement non préemptif** (sans réquisition): le processus à l'état élu est traité jusqu'à sa fin quelque soit sa durée. L'importance, l'urgence etc. ne sont pas prise en compte. Le diagramme d'états des processus se presente comme le montre la **Figure2.**

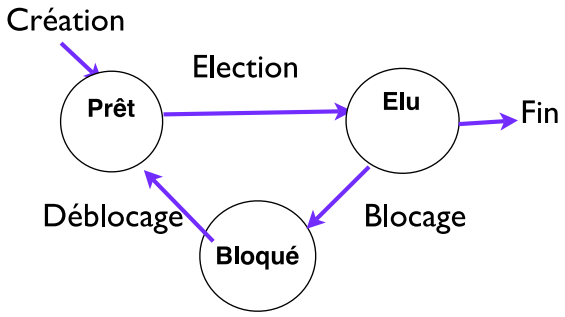

Figure2: Diagramme d'états pour un ordonnancement non préemptif

Ordonnancement préemptif (avec réquisition): en fonction des critères d'ordonnancement, le SE remet le processus en file d'attente avant la fin de son exécution (voir **Figure3**)

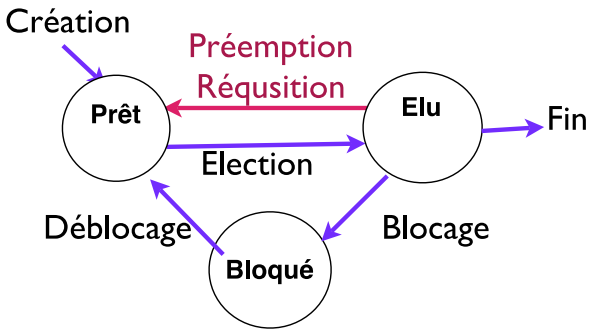

Figure2: Diagramme d'états pour un ordonnancement préemptif

III. LES ALGORITHMES D'ORDONNANCEMENT

III.1. Ordonnancement non préemptif

L'ordonnacement non premptif permet à un processus d'accéder au processeur via la file d'attente. Une fois que le processus a le processeur, le traitement s'effectue jusqu'à terminaison, il n'est jamais interrompu quelque soit sa durée, l'importance ou l'urgence ne sont pas pris en compte.

III.1.1. FCFS (First Come First Served)

C'est l'agorithme du Premier Arrivée, Premier servi (PAPS). Le traitement est sequentiel, le Premier processus arrivé en file d'attente, sera le premier servi. Les processus courts sont pénalisés au profit des processus longs à l'image d'une photocopieuse utilisée par une personne qui photocopie un livre avant celle qui photocopie une page.

Soient les trois processus P1, P2, P3 don't l'ordre d'arrivée est P1, P2, P3 et le temps d'exécution ou de traitement est résumé dans le **Tableau 1.**

Processus	P1	P2	P3
Durée d'exécution	24	3	3

Tableau1: Temps d'exécution des processus

Le schéma d'exécution se présente comme indiqué sur la Figure3 et le schéma d'attente sur la **Figure4**.

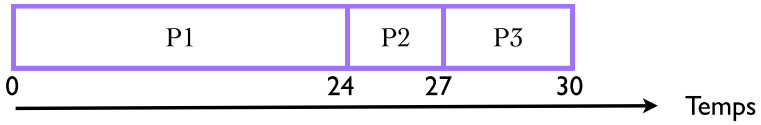

Figure3: Schéma d'exécution des processus selon FCFS

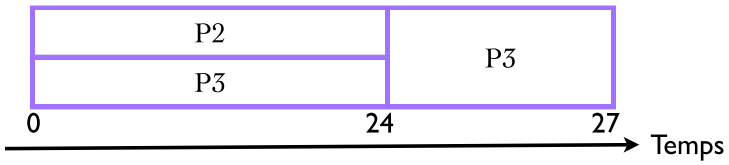

Figure4: Schéma d'attente des processus selon FCFS

Ainsi le temps d'attente pour P1 est 0 unité de temps, 24 pour P2 et 27 pour P3. Le temps d'attente moyen est (0 + 24 + 27)/3 = 17.

Si les processus arrivent au temps 0 dans l'ordre: P_2, P_3, P_1. Le schéma d'exécution se présente comme indiqué sur la **Figure5** et le schéma d'attente sur la **Figure6.**

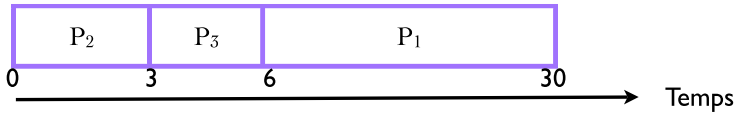

Figure5: Schéma d'exécution des processus selon FCFS

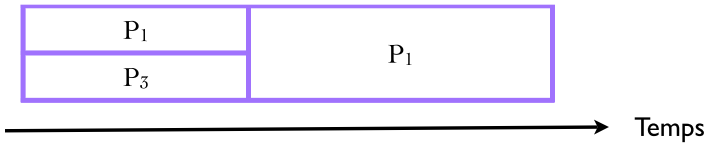

Figure4: Schéma d'attente des processus selon FCFS

Ainsi le temps d'attente pour P1 est 6 unité de temps, 0 pour P2 et 3 pour P3. Le temps d'attente moyen est (6 + 0 + 3)/3 = 3. On peut noter que le temps d'attente peut varier en fonction de l'ordre d'arrivée.

III.1.1. SJF (Shortest Job First)

L'alogorithme du plus court travail d'abord est une évolution de la stratégie précédente. La file d'attente est ordonnée non plus de façon chronologique mais en fonction du temps d'exécution nécessaire (on fait passer en tête les travaux courts). Un processus qui arrive est classé dans la file d'attente en fonction de sa durée. Les travaux longs sont pénalisés, au pire ils risquent de ne jamais être exécutés tant que de petits travaux sont soumis au système. En cas d'égalité de temps d'exécution, on applique FIFO

III.2. Ordonnancement préemptif

L'ordonnacement préemptif consiste à décider en fonction de certains critères, de remettre le processus en file d'attente avant la fin de son execution. Un processus peut dans certains cas faire plusieurs passages dans la file d'attente.

III.2.1. SRTF (Shortest Remaining Time First)

L'algorithme du plus court temps restant d'abord favorise le temps restant le plus court. Si un processus qui dure moins que le restant du processus courant se présente plus tard, l'UCT est enlevée au processus courant et donnée à ce nouveau processus. Il favorise les travaux courts et les travaux avec un temps de traitement restant plus court.

III.2.2. RR (Round Robing)

Dans l'algorithme du tourniquet le SE attribue à chaque processus un temps (quantum de temps) pendant lequel il est autorisé à s'exécuter. L'UCT est enlevée au processus courant P_x si le quantum s'achève avant la fin du processus et est donnée au premier processus P_y de la file d'attente. Le processus P_x se trouve ainsi en queue de liste. La gestion de la file d'attente est faite selon FIFO. Les travaux assez courts sont vite servis. Le tourniquet garanti que les travaux longs sortiront du système au bout d'un temps fini. L'efficacité du système dépend de la valeur du quantum. Si le changement de contexte s'opère en *c* unités de temps et le quantum fixé à *q* unités de temps. La surcharge introduite par le système est: $c/(c+q)$. Un quantum trop petit provoque trop de changements de contexte et augmente la surcharge, ce qui ralentit la machine. Tandis que pour un quantum trop long, la réactivité du système diminue, les petits travaux sont penalizes.

IV. Algorithme Priorité (avec ou sans préemption)

L'algorithme de priorité affectation une priorité à chaque processus (p.ex. un nombre entier) souvent les petits chiffres dénotent des hautes priorités (exemple: 0 la plus haute)

L'UCT est donnée au processus prêt avec la plus haute priorité **avec** ou **sans** preemption. A priorité égale, on applique FIFO. Cet algorithme peut engendrer des problèmes comme par exemple la famine où les processus moins prioritaires n'arrivent jamais à executer. L'ajustement dynamique de la priorité qui consiste à modifier la priorité d'un processus en fonction de son âge et de son historique d'exécution peut être une solution. Par exemple,

On peut augmenter la priorité des processus penalisés (plus âgés dans le système) qui peinent à terminer leur execution à cause des processus plus prioritaires. On peut diminuer la priorité d'un processus en fonction du temps passé dans le CPU (nombre de passages). Un processus P1 qui passe à 2 reprises dans le CPU peut voir sa priorité diminuée au profit d'un processus P2 plus ancien et qui n'a jamais occupé le CPU.