# Systèmes d'exploitation 1: Ordonnancement de Processus

## *1. Définition et objectifs de l'ordonnancement*

L'ordonnancement (Scheduling) règle les transitions d'un état à un autre des différents processus. Cet ordonnancement a pour objectifs :
1. de maximiser l'utilisation du processeur;
2. d'être équitable entre les différents processus;
3. de présenter aux utilisateurs un temps de réponse acceptable;
4. d'avoir un bon rendement;
5. d'assurer certaines priorités;

Si plusieurs processus sont dans l'état « prêt », c'est l'ordonnanceur qui choisit le processus à exécuter (le fait passer à l'état élu) en utilisant un algorithme qui implémente une politique d'ordonnancement. L'ordonnancement peut être préemptif ou non. Il est dit non préemptif si, une fois que le processeur a été alloué à un processus, il le gardera jusqu'à sa terminaison (temporaire ou définitive). Il est préemptif dans le cas contraire.

## *2. La politique FCFS (First-Come, First-Served: Premier arrivé, premier servi)*

La manière la plus simple de gérer la CPU est l'algorithme où le processus qui la sollicite en premier, sera le premier à être servi. Quand le processus en cours d'exécution se termine, le processeur est alloué au prochain processus. C'est un algorithme simple à utiliser et à implémenter (avec une file d'attente ou une liste chaînée), mais ses performances sont réduites.

## *3. La politique SJF(Shortest Job First: Le plus court d'abord)*

Cet algorithme associe à chaque processus un temps d'exécution estimé. Le processeur est alloué au processus dont le temps d'exécution estimé est minimal. Cette politique avantage les travaux courts. Elle est souvent utilisée dans les systèmes batch. La difficulté principale réside dans la connaissance à priori des temps CPU nécessaires aux différents travaux.
SJF peut être préemptif ou non. Il peut devenir préemptif et se transformer en SRTF (Shortest Remained Time First : Le temps restant le plus court d'abord). SRTF traite le problème suivant :
Si le nouveau processus qui arrive a un temps estimé d'exécution plus court que le temps restant pour terminer l'exécution du processus en cours (élu), ce dernier sera interrompu et la CPU est alloué au nouveau processus.

## *4 La politique du tourniquet (Round Robin ou RR) ou ordonnancement ciculaire*

Cet algorithme est l'un des plus utilisés et l'un des plus fiables et a été conçu pour des systèmes à temps partagé. Chaque processus prêt dispose d'un quantum de temps pendant lequel il s'exécute. Lorsqu'il a épuisé ce temps, qu'il s'est terminé avant, ou qu'il se bloque, par exemple sur une entrée-sortie, le processus suivant de la file d'attente est élu et le remplace. Le processus suspendu (qui a terminé son quantum mais pas son temps UC) est mis en queue du tourniquet.
Le seul paramètre important à régler, pour le tourniquet, est la durée du quantum. Il doit minimiser le temps de gestion du système et cependant être acceptable pour les utilisateurs. La part de gestion du système correspond au rapport de la durée de commutation sur la durée du quantum. Plus le

quantum est long plus cette part est faible, mais plus les utilisateurs attendent longtemps leur tour. Un compromis peut se situer, suivant les machines, de 100 à 200 ms.

### *5. La politique de la plus haute priorité*

Dans l'algorithme du tourniquet, les quantums égaux rendent les différents processus égaux. Il est parfois nécessaire de privilégier certains processus par rapport à d'autres. L'algorithme de priorité choisit le processus prêt de plus haute priorité (valeur la plus basse, en général).
L'ordonnancement avec priorité peut être préemptif ou non. Dans le cas préemptif, quand un nouveau processus arrive dans la file d'attente des processus prêts, sa priorité est comparée avec celle du processus élu et l'UC sera affectée au nouveau processus si sa priorité est pplus haute que celle de l'élu actuel. Dans le cas non préemptif, le nouveau processus sera simplement inséré dans la file d'attente des processus prêts (en tête s'il a la plus haute priorité).
Ces priorités peuvent être statiques ou dynamiques. Les processus du système auront des priorités statiques (non-modifiables) fortes. Les processus des utilisateurs verront leurs priorités modifiées, au cours de leur exécution, par l'ordonnanceur. Ainsi un processus qui vient d'utiliser l'UC verra sa priorité baisser.

### *6. La politique du tourniquet avec priorités*

On utilise généralement une combinaison des deux techniques précédentes. À chaque niveau de priorité correspond un tourniquet. L'ordonnanceur choisit le tourniquet non vide de priorité la plus forte et l'exécute.
Pour que tous les processus puissent s'exécuter, il est nécessaire d'ajuster périodiquement les différentes priorités.

### *7. L'ordonnancement des fils d'exécution*

Les fils d'exécution sont soumis à un ordonnancement. Dans Windows NT, les fils d'exécution ont 32 niveaux de priorité qui sont soit fixes soit dynamiques. La priorité la plus haute est toujours celle qui s'exécute. Dans le cas d'une priorité dynamique, les valeurs de cette priorité varient entre deux bornes. Elle augmente, par exemple, lors d'une attente d'entrée-sortie. On peut changer la priorité des fils d'exécution dans Windows NT par la fonction SetThreadPriority().
Avec Java, les niveaux de priorité varient entre Thread.MIN_PRIORITY et Thread.MAX_PRIORITY. La priorité normale est Thread.NORM_PRIORITY. Comme avec NT, la priorité la plus haute est toujours celle qui s'exécute. Certains interprètes utilisent un tourniquet qui gère les fils de même priorité sans limite de temps pour le fil qui s'exécute. D'autres interprètes limitent l'exécution d'un fils à un quantum de temps, puis passent à un autre fil d'une même priorité. Le modèle d'exécution peut être assez différent suivant les machines, par exemple entre l'IBM 580 et le Sun E3000.

# Série d'exercices SE1: Ordonnancement de Processus

**Exercice 1**

Soit l'ensemble des processus suivants :

| Processus | Temps estimé d'exécution | Priorité |
|---|---|---|
| P1 | 10 | 3 |
| P2 | 1 | 1 |
| P3 | 2 | 3 |
| P4 | 1 | 4 |
| P5 | 5 | 2 |

On suppose que les processus sont arrivés dans cet ordre approximativement à l'instant T=0 et que la priorité de valeur minimale est la plus haute. On définit le temps de réponse d'un processus comme étant la durée qui sépare son instant d'arrivée de celui de la fin de son exécution. Déterminer le temps de réponse de chaque processus et le temps moyen de réponse dans chacun des cas suivants :

1) FCFS 2) SJF 3) Tourniquet (quantum=1 puis =2) 4) Priorité sans préemption.

**Exercice 2**

Soit l'ensemble des processus suivants :

| Processus | Instant d'arrivée | Temps estimé d'exécution | Priorité |
|---|---|---|---|
| P1 | 0 | 8 | 3 |
| P2 | 1 | 4 | 1 |
| P3 | 2 | 1 | 2 |

Calculer le temps de réponse de chaque processus et le temps moyen de réponse dans chacun des cas suivants :

1) FCFS 2) Priorité sans préemption 3) Priorité avec préemption 4) SJF 5) SRTF

**Exercice 3**

Cinq travaux notés de A à E, demandent à être exécutés au même instant. Leurs temps d'exécution respectifs sont estimés à 10, 6, 2, 4 et 8 minutes. Leurs priorités, déterminées de manière externe, sont respectivement 3, 5, 2, 1 et 4. Déterminer les temps moyens d'attente et de réponse pour chacun des algorithmes d'ordonnancement suivants: "Tourniquet" (avec quantum = 1), "Avec priorités", "Premier arrivé, premier servi", "le plus court d'abord".

Pour le premier mécanisme d'ordonnancement, on considère que le SE est multiprogrammé et que le temps processeur est équitablement réparti entre les différents travaux. Pour les autres, on suppose que chaque travail est exécuté jusqu'à ce qu'il se termine. Dans tous les cas, les travaux n'effectuent pas d'E/S.

**Exercice 4**

Simulez l'exécution de 2 processus P1 et P2 sur une machine dont le gestionnaire possède les caractéristiques suivantes:

Les priorités varient dynamiquement de 1 à 5, où 5 dénote la plus forte priorité: quand un processus ne consomme pas entièrement son quantum, sa priorité est décrémentée. Les quantums varient dynamiquement de 1 à 5 unités de temps: quand un processus ne consomme pas entièrement son quantum, son quantum est incrémenté. Les priorités varient de manière inversement proportionnelle aux quantums: quand la priorité d'un processus est décrémentée, son quantum est incrémenté. Le temps de commutation de contexte est de 1 unité de temps, le temps d'exécution de l'algorithme d'ordonnancement est négligeable. Les processus ont les caractéristiques suivantes:

- Ils sont soumis en même temps
- Les quantums initiaux sont de 2 pour P1 et de 5 pour P2
- Les priorités initiales respectives sont 2 et 3
- Chacun des deux processus requièrent 16 unités de temps, E/S non inclues
- P1 lance une E./S au $4^{ème}$ et $10^{ème}$ instants de son exécution, qui durent toutes les deux une unité de temps.
- P2 lance une E./S au $3^{ème}$ et $11^{ème}$ instants de son exécution, qui durent respectivement 7 et 3 unités de temps.

Donner le temps de réponse des deux processus, en montrant le scénario d'exécution de ces processus.