L'ordonnanceur

Dans un système multitâches à temps partagé, plusieurs processus peuvent être présents en mémoire centrale en attente d'exécution. Si plusieurs processus sont prêts, le système d'exploitation doit gérer l'allocation du processeur aux différents processus à exécuter. **L'ordonnanceur** s'acquitte de cette tâche.

1. Notion d'ordonnancement

Le système d'exploitation d'un ordinateur peut être vu comme un ensemble de processus dont l'exécution est gérée par un processus particulier : l'**ordonnanceur** (scheduler en anglais).
Un ordonnanceur fait face à deux problèmes principaux :

- ✗ le choix du processus à exécuter.
- ✗ le temps d'allocation du processeur au processus choisi.

Les objectifs d'un ordonnanceur d'un système multitâches sont, entre autres :

- ✗ s'assurer que chaque processus en attente d'exécution reçoive sa part de temps processeur ;
- ✗ minimiser le temps de réponse ;
- ✗ utiliser le processeur à 100% ;
- ✗ utiliser d'une manière équilibrée les ressources ;
- ✗ prendre en compte les priorités ;
- ✗ être prédictible.

Systèmes multitâches préemptifs

Un système d'exploitation multitâche est **préemptif** lorsque celui-ci peut arrêter (réquisition) à tout moment n'importe quelle application pour passer la main à la suivante. Dans les systèmes d'exploitation préemptifs on peut lancer plusieurs applications à la fois et passer de l'une à l'autre, voire lancer une application pendant qu'une autre effectue un travail.

L'ordonnancement

Il existe plusieurs politiques d'ordonnancement afin d'optimiser la réactivité de la machine ainsi que le taux d'utilisation des processeurs. Parmi les principales techniques, on peut citer :

- ✗ Premier arrivé, premier servi (FIFO) : simple, mais peu adapté à la plupart des situations.
- ✗ Plus court d'abord : très efficace, mais la plupart du temps il est impossible d'anticiper le temps d'exécution d'un processus.
- ✗ Priorité : le système alloue un niveau de priorité aux processus. Cependant des processus de faible priorité peuvent ne jamais être exécutés.
- ✗ Tourniquet : un quantum de temps est alloué à chaque processus. Si le processus n'est pas terminé au bout de ce temps, il est mis en bout de file en état prêt.
- ✗ Un système hybride entre tourniquet et priorité qu'on retrouve dans les systèmes UNIX.

2.Interblocage

Le fonctionnement multitâche apporte malheureusement de nouveaux problèmes comme celui de **l'interblocage**.
Imaginez deux processus, le premier s'est accaparé une ressource A et le second une ressource B ; chacun convoite ensuite la ressource de l'autre et n'est prêt à libérer sa propre ressource que si, celle de l'autre devient accessible. On est là face à une situation de blocage mutuel, un interblocage (**deadlock**).
Celle-ci se traduit par la mise en sommeil définitif des deux processus, dans l'attente d'une libération de ressource qui ne surviendra jamais. La figure 1 illustre une situation simple d'interblocage.

Dans un article de 1971, Jr Coffman a établi les conditions d'un interblocage :

- ✗ Au moins une ressource doit être conservée dans un mode non partageable.
- ✗ Un processus doit maintenir une ressource et en demander une autre.
- ✗ Une ressource ne peut être libérée que par le processus qui la détient.
- ✗ Chaque processus doit attendre la libération d'une ressource détenue par un autre qui fait de même.

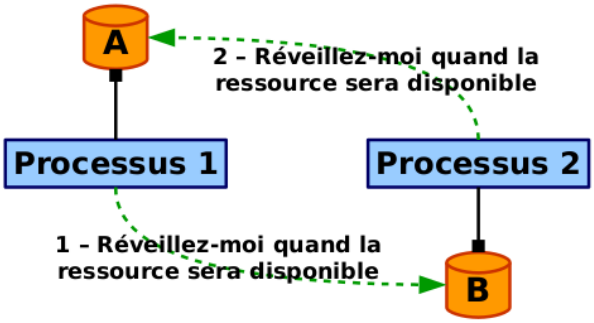

Figure 1: Situation d'interblocage

3.Processus Légers (Thread)

Parallèlement aux processus, il existe des processus dits légers qui partagent le même espace d'adressage, hormis la pile qui est propre à chacun. On les appelle des threads.
Ils sont d'un usage très courant dans le pilotage des interfaces graphiques, l'encodage vidéo, les calculs scientifiques... car ils permettent de tirer le meilleur parti des micro-processeurs actuels (multithreading) tout en **réduisant le coût de la commutation de contexte**. Du point de vue de l'ordonnanceur ces threads sont considérés comme des processus (tâche) à part entière.