Gestion des processus et des ressources

Dans les années 1980 les ordinateurs personnels n'étaient pas capables d'exécuter plusieurs tâches à la fois : on lançait un programme et on y restait jusqu'à ce que celui-ci plante ou se termine.

Les systèmes d'exploitation récents (Windows, Linux ou osX par exemple) permettent d'exécuter plusieurs tâches simultanément - ou en tous cas, donner l'impression que celles-ci s'exécutent en même temps.

A un instant donné, il n'y a donc pas un mais plusieurs programmes qui sont en cours d'exécution sur un ordinateur : on les nomme processus. Une des tâches du système d'exploitation est d'allouer à chacun des processus les ressources dont il a besoin en termes de mémoire, entrées-sorties ou temps processeur, et de s'assurer que les processus ne se gênent pas les uns les autres.

Vocabulaire

Programme exécutable : un fichier binaire contenant les instructions machines directement exécutable par le processeur de la machine.

Processus : « Programme en cours d'exécution ». Un processus est un phénomène dynamique qui correspond à l'exécution d'un programme. Le système d'exploitation identifie les processus par un numéro unique. Un processus est décrit par :

L'ensemble de la mémoire allouée par le système pour l'exécution de ce programme (code en mémoire et données).

Thread ou tâche : exécution d'une suite d'instructions démarrées par un processus.

Exemple : Un navigateur web lance un processus, il peut y avoir un thread qui affiche une image et un autre thread qui télécharge un fichier.

La différence fondamentale entre le processus et le thread est que les processus ne partagent pas leur mémoire, alors que les threads, issus d'un même processus peuvent accéder aux variables globales du programme et occupent le même espace mémoire.

Exécution concurrente : deux processus s'exécutent de manière *concurrente* si les intervalles de temps entre le début et la fin de leur exécution ont une partie commune.

Exécution parallèle : deux processus s'exécutent en parallèle s'ils s'exécutent au même instant. Il faut plusieurs processeurs sur la machine.

États d'un processus

Tous les systèmes d'exploitation "modernes" (Linux, Windows, macOS, Android, iOS...) sont capables de gérer l'exécution de plusieurs processus en même temps. Mais pour être précis, cela n'est pas en véritable "en même temps", mais plutôt un "chacun son tour". Pour gérer ce "chacun son tour", les systèmes d'exploitation attributs des "états" au processus.

Voici les différents états :

Lorsqu'un processus est en train de s'exécuter (qu'il utilise le microprocesseur), on dit que le processus est dans l'état **"élu"**.

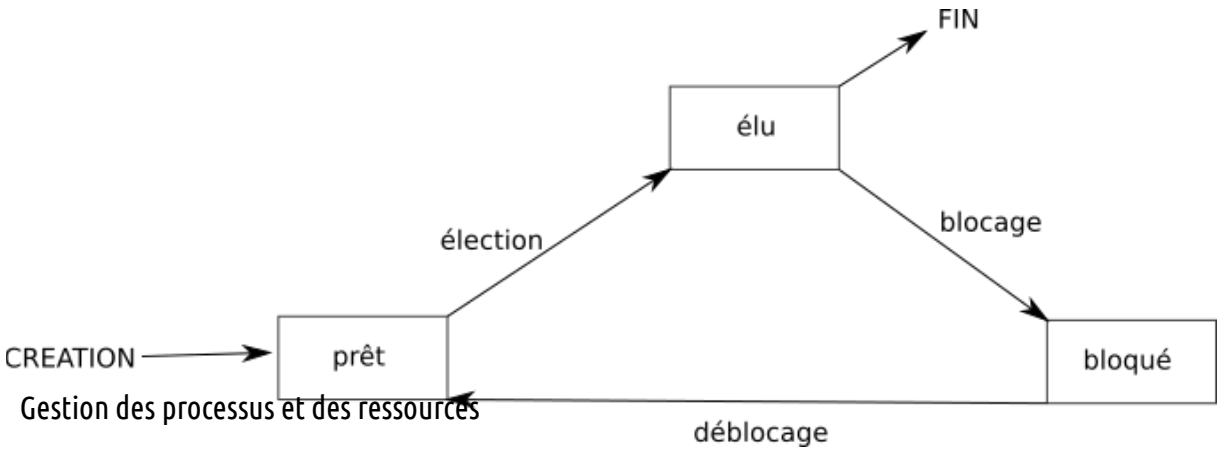
- Un processus qui se trouve dans l'état élu peut demander à accéder à une ressource pas forcément disponible instantanément (par exemple lire une donnée sur le disque dur). Le processus ne peut pas poursuivre son exécution tant qu'il n'a pas obtenu cette ressource. En attendant de recevoir cette ressource, il passe de l'état **"élu"** à l'état **"bloqué"**
- Lorsque le processus finit par obtenir la ressource attendue, celui-ci peut potentiellement reprendre son exécution. Mais comme nous l'avons vu ci-dessus, les systèmes d'exploitation permettent de gérer plusieurs processus **"en même temps**", mais un seul processus peut se trouver dans un état **"élu"**

le microprocesseur ne peut "s'occuper" que d'un seul processus à la fois.

- Quand un processus passe d'un état **"élu"** à un état **"bloqué"**, un autre processus peut alors **"prendre sa place"** et passer dans l'état **"élu"**. Le processus qui vient de recevoir la ressource attendue ne va donc pas forcément pouvoir reprendre son exécution tout de suite, car pendant qu'il était dans à état "bloqué" un autre processus a "pris sa place". Un processus qui quitte l'état bloqué ne repasse pas forcément à l'état "élu", il peut, en attendant que "la place se libère" passer dans l'état "prêt" (sous entendu "j'ai obtenu ce que j'attendais, je suis prêt à reprendre mon exécution dès que la "place sera libérée"").

Le passage de l'état **"prêt"** vers l'état **"élu"** constitue l'opération **"d'élection"**. Le passage de l'état élu vers l'état bloqué est l'opération de **"blocage"**. Un processus est toujours créé dans l'état "prêt". Pour se terminer, un processus doit obligatoirement se trouver dans l'état **"élu"**.

On peut résumer tout cela avec le diagramme suivant :



Gestion des processus - Ordonnanceur du système d'exploitation

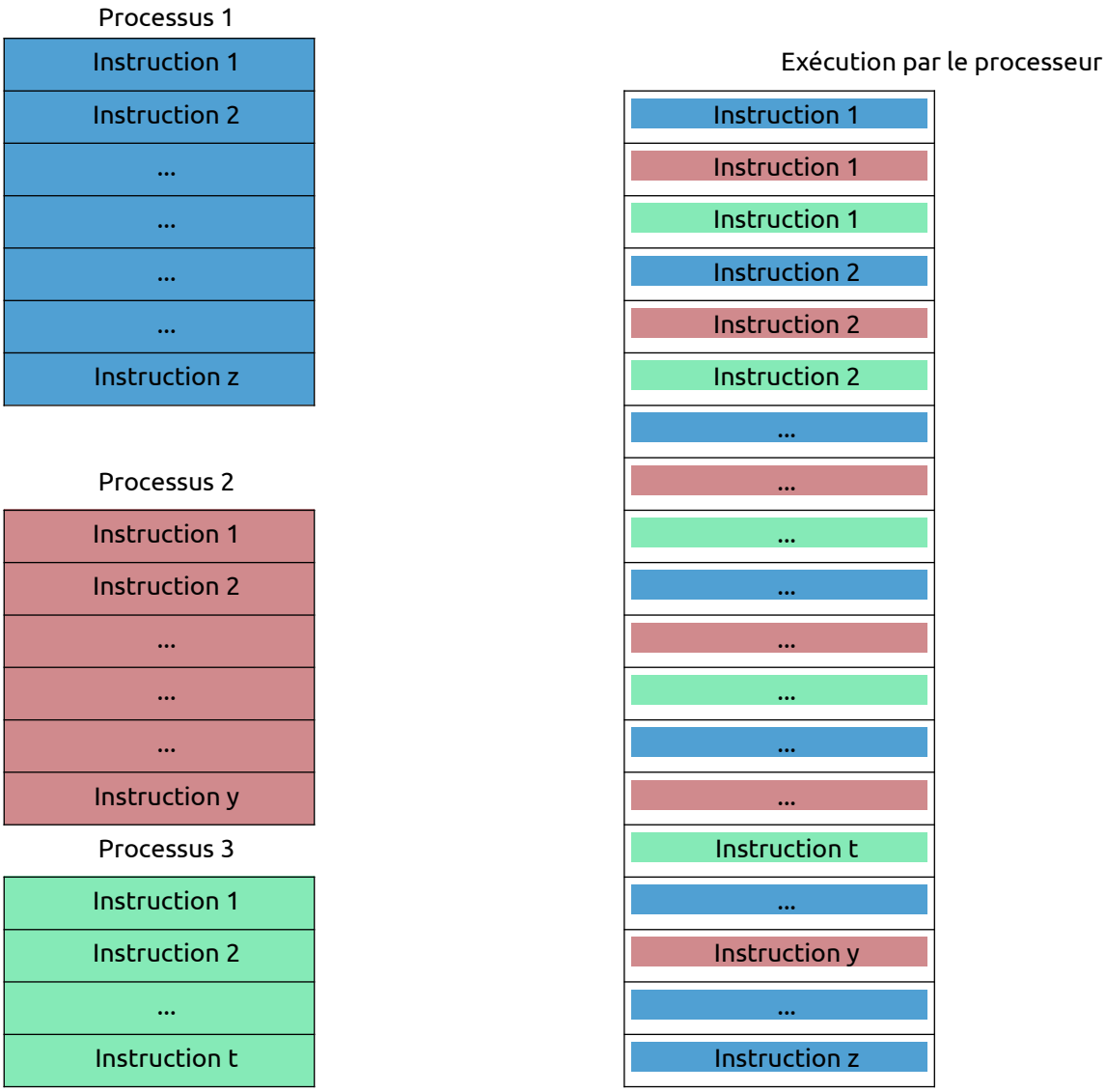
Le système d'exploitation doit permettre à toutes les applications et tous les utilisateurs de travailler en même temps. Cette gestion complexe des processus est réalisée par une partie du noyau : **l'ordonnanceur**.

L'ordonnanceur permet :

- de minimiser le temps de traitement du processus d'un utilisateurs
- de garantir l'équité entre les différents utilisateurs
- d'optimiser l'utilisation de la ressources
- d'éviter les interblocages

Plusieurs algorithmes d'ordonnancement sont possibles. Par exemple :

1- Round-robin (ou méthode du tourniquet)



2- Rate-monotonic scheduling - L'ordonnancement à taux monotone

Il attribue la priorité la plus forte à la tâche qui possède la plus petite période.

3- Fifo – First In First Out

Gestion du premier entré, premier sorti (ex : file d'attente d'une imprimante)

4- l'algorithme du plus court d'abord

l'algorithme du plus court d'abord, mais il n'est pas toujours facile d'évaluer le temps d'exécution d'une tâche.

Interblocage (ou deadlock)

En informatique également, l'interblocage peut se produire lorsque des processus concurrents s'attendent mutuellement. Les processus bloqués dans cet état le sont définitivement. Ce scénario catastrophe peut se produire dans un environnement où des ressources sont partagées entre plusieurs processus et l'un d'entre eux détient indéfiniment une ressource nécessaire pour l'autre.

Cette situation d'interblocage a été théorisée par l'informatitien Edward Coffman (1934-) qui a énoncé quatre conditions (appelées conditions de coffman) menant à l'interblocage

PID, PPID

Un processus est caractérisé par un identifiant unique : son PID (Process Identifier). Lorsqu'un processus engendre un fils, l'OS génère un nouveau numéro de processus pour le fils. Le fils connaît aussi le numéro de son père : le PPID (Parent Process Identifier). ps -ef > processus.txt

Sous linux on peut afficher les processus à l'aide de la commande **ps** (processus)
On peut tuer un processus à l'aide de la commande kill. → kill n°processus

```
$ kill 9653
```

kill -9 n°processus permet de tuer le processus sans que ce dernier ne puisse l'intercepter.
Sous windows ; le gestionnaire de tâches joue un rôle similaire (Ctrl-Alt-Supp)