Processus

Un **programme** est une réalisation concrète d'algorithme, dans un certain langage de programmation. Il se traduit généralement par un **fichier programme** exécutable par la machine.

Un processus est une instance d'exécution de programme.

Contexte

Un processus a un contexte géré par le système d'exploitation : date de démarrage, durée de vie, espace mémoire utilisé, accès au temps de calcul des processeurs...
À la création, un processus reçoit quelques attributs qui permettent à l'OS de l'identifier :

- un identifiant : **PID**,
- les processus sont créés par d'autres processus, si bien que chaque processus a un père et connaît l'identifiant de son père : **PPID**,
- un utilisateur est à l'origine de la création du processus. Son identifiant est l'**UID**.

Certains processus sont lancés par le système – souvent appelés services, qui s'exécutent en tâche de fond, en permanence - d'autres par les utilisateurs – plutôt des applications.

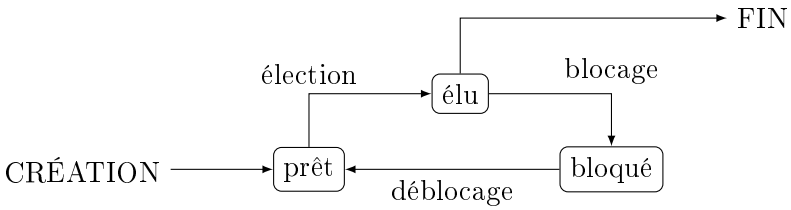
Visualisation des processus

Sous Windows, on dispose d'application avec interface graphique telle que le gestionnaire des tâches, accessible avec la combinaison CTRL+ALT+SUPR.

On peut aussi utiliser un terminal et la commande `tasklist`.

La commande `tskill <pid>` permet de tuer un processus en indiquant son PID ; `tskill <name>` fait la même chose mais avec le nom.

États



- Un processus prêt se présente à l'exécution,
- Le processus élu est celui en cours d'exécution,
- Si le processus élu demande une ressource qui n'est pas disponible, il passe dans l'état bloqué. Le système peut donc élire un autre processus.
- Lorsque la ressource se libère, le processus bloqué ne recommence pas son exécution tout de suite. Il revient dans l'état prêt, attendant d'être élu par le système.

Séquencement

Le **système d'exploitation** est chargé de gérer l'exécution de ces processus ainsi que les **ressources** qu'ils utilisent.

En général, une machine exécute plusieurs processus *à peu près* en même temps. Mais le processeur ne peut faire qu'une chose à la fois. La machine donne l'illusion de faire deux choses en même temps en exécutant les processus rapidement à tour de rôle.

On appelle **ordonnancement des processus** la gestion de ces tours.

Définitions

- **Soumission :** moment auquel est demandé l'exécution du processus. On soumet au système d'exploitation une demande d'exécution, c'est le système d'exploitation qui décide.
- **Durée :** temps d'exécution nécessaire pour le processus. Difficile à déterminer à priori.
- **Priorité :** certains processus peuvent être jugés plus urgents que d'autres et donc passer devant les autres.
- **Temps d'attente :** délai entre l'instant de soumission et l'instant de la première exécution.
- **Temps d'exécution :** délai entre l'instant de soumission et l'instant où le processus est complètement terminé.

cf. exercices pour les exemples.

FIFO : Premier arrivé premier servi

- Pas de notion de priorité sur les processus,
- non **préemptif** - *un processus démarré n'est pas interrompu pour en exécuté un autre*,
- pas de connaissance sur la durée des processus,
- temps d'attente augmente vite

On peut utiliser une stratégie FIFO avec priorité : Chaque fois que le système est libre, le système d'exploitation liste les processus qui lui ont été soumis. Parmi ceux là, il sélectionne le plus prioritaire. À priorité égale, il sélectionne celui soumis en premier.

SJF : Shortest Job First - Plus court en premier

Chaque fois que le système est libre, le système d'exploitation liste les processus qui lui ont été soumis. Parmi ceux là, il évalue les temps d'exécutions et il sélectionne celui dont le temps d'exécution sera **le plus court**.

- Pas de notion de priorité sur les processus,
- non préemptif,
- nécessite une évaluation de la durée des processus,
- risque de **privation** : si des processus court se présentent constamment, un processus long ne pourra jamais s'exécuter !

RR : Round Robin - Tourniquet

- On découpe l'exécution en **quantum** – *par ex. quantum de 1 unité de temps.*
- Quand un processus se présente, il se met à la fin de la file d'attente,
- au début d'un nouveau quantum, on sélectionne le processus en tête de file,
- le processus sélectionné se place aussitôt à la fin de la file.

- pas de notion de priorité sur les processus,
- préemptif - *un processus démarré s'interrompt pour faire place à un autre*,
- ne nécessite pas de connaissance sur la durée des processus.

Accès aux ressources

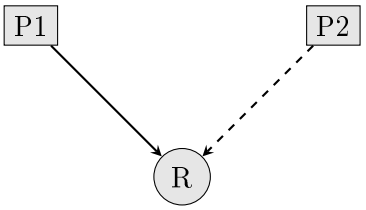
Certains processus ont besoin d'accéder à des ressources : *fichiers, ports E/S, lecteurs...*
Le système d'exploitation doit donner accès aux ressources.
Problème : deux processus ne peuvent pas modifier en même temps un même fichier ou écrire sur un même port.

Le **système d'exploitation** a pour rôle de gérer ces **accès multiples** par exemple en **bloquant** temporairement **un processus** en **attendant** que la **ressource** dont il a besoin soit **libérée**.

cf. Problème du dîner des philosophes.

Interblocage - cas sans problème

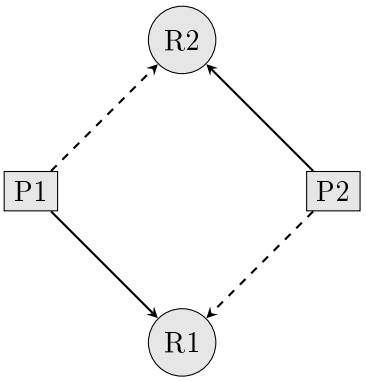
P1 et **P2** ont besoin de **R**. **P1** prend la main en premier, **P2** attend que **P1** ait terminé pour prendre son tour.



Interblocage - deadlock

Cas d'interblocage : **P1** et **P2** ont besoin de **R1** et **R2**. Chacun a pris la main sur une des deux ressources et attend l'autre.

C'est le problème des philosophes.



Prévoir les interblocages est possible mais c'est difficile voire irréaliste. *cf. algorithme du banquier*

On peut les éviter en programmant de façon plus sûre.

Exemple : poser les verrous toujours dans le même ordre.
Dans le cas précédent le deadlock intervient parce que P1 verrouille R1 puis R2 alors que P2 verrouille R2 puis R1.

En général, le système d'exploitation ne cherche pas à empêcher les interblocages. Si un interblocage survient, le système le détecte, tue un des processus puis relance le processus tué une fois la situation débloquée.