Gestion des ressources et processus

5.1 Rappels : Système d'exploitation

a. Qu'est-ce qu'un système d'exploitation

Un **système d'exploitation** (Operating System en anglais, "OS" en abrégé) est une « couche » logicielle qui permet de gérer et de faire fonctionner les divers matériels qui composent l'ordinateur (processeur, mémoire centrale, horloges, terminaux, mémoires secondaires (disque, disquettes), périphériques (réseau, souris, etc.)

Le système d'exploitation sert d'interface entre les applications au sens large du terme et le matériel.

b. Les tâches du système d'exploitation

Le système d'exploitation s'occupe principalement de deux tâches.

- **le rôle de machine virtuelle** qui consiste à offrir une interface commune, indépendante du matériel, aux logiciels et aux utilisateurs.
- **le rôle de gestionnaire de ressources** qui consiste à ordonner et à contrôler l'allocation des processeurs, des mémoires et des périphériques entre les différents programmes qui y font appel.

c. Les processus

Il convient de bien faire la distinction entre un programme et un processus. Bien que proche ils representent deux réalités différents.

- **Un programme** est produit par un compilateur ou éditeur d'un langage de programmation. C'est un objet inerte correspondant au contenu stocké sous forme d'un fichier sur disque dur ou autre.
- **Un processus** est un objet dynamique évoluant dans le temps. Il s'agit d'une abstraction correspondant à l'exécution d'un programme et son contexte. Un processus, en plus du programme executé, contient :
  - ⋆ l'état du processeur à un instant donné ou **contexte d'exécution**. Ce contexte d'exécution est composé de l'ensemble des valeurs contenues dans les registres du processeur.
  - ⋆ Les données du programme ou **contexte mémoire**.

Les systèmes d'exploitation multi-tâches sont capables d'exécuter plusieurs proccessus simultanément. Cependant le processeur ne peut executer qu'un programme à la fois mais peut passer de l'un à l'autre en une infime fraction de temps donnant ainsi l'illusion d'une exécution simultanée. La partie du système d'exploitation en charge de coordonner l'allocation de ressources aux différents processus se nomme **l'ordonnanceur** (scheduler en anglais).

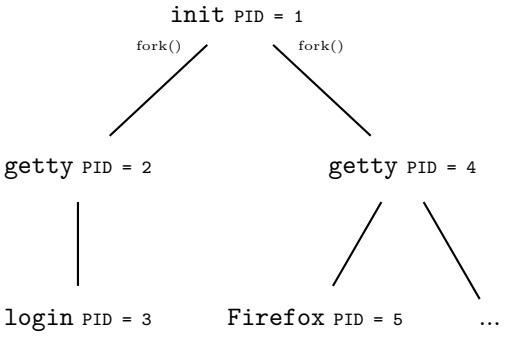
5.2 Gérer les proccessus

a. Création des proccessus

Les processus peuvent avoir deux origines distinctes.

⇒ **Les processus systèmes** sont créés par le système d'exploitation. On dit qu'ils sont exécutés en « kernel mode » et ils ont pour propriétaire le super-utilisateur du système `root`.

⇒ **Les processus utilisateurs** sont lancés par l'utilisateur et executés en « user mode » et ont pour propriétaire l'utilisateur en question.

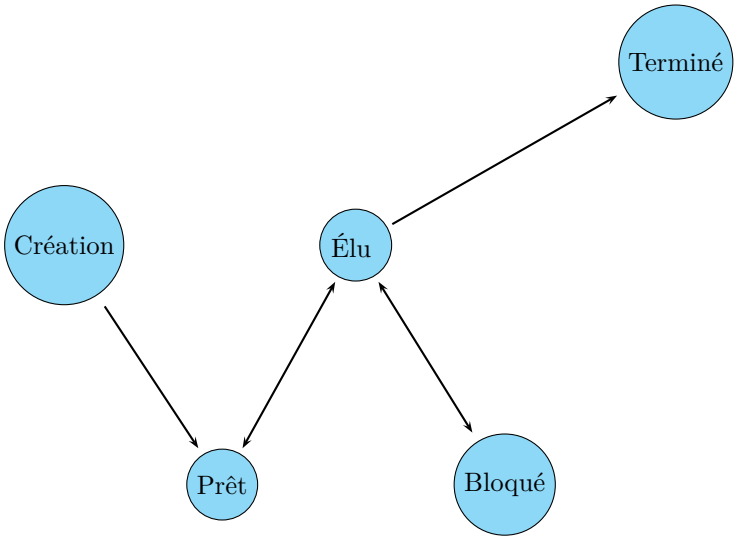


Dans les systèmes Unix, un processus est créé uniquement par **un autre processus.** Ainsi il existe une filiation entre les processus, que l'on peut représenter sous la forme d'arbre dont la racine est le processus `init`. Ensuite chaque processus peut avoir zéro, un ou plusieurs processus fils.

Lors du démarage du système UNIX, le système créé le processus `init`, ce sera le premier processus créé son `PID`, Processus IDentification, est 1. Par la suite un processus est créé par duplication/clonage du processus père par l'appel système `fork()`.
On peut d'ailleurs avoir un aperçu de cette arborecence à l'aide de la commande `pstree`.

b. État d'un processus

Quand un processus s'exécute, il est amené à changer d'état (voir illustration ci-dessous).



Il peut se trouver dans chacun des états suivants

Nouveau : le processus est en cours de création.

Prêt : en attente de processeur ;

Élu : en cours d'exécution ;

Bloqué : en attente de ressource ;

Terminé : le processus a fini l'exécution.

C'est l' `ordonnanceur` (scheduler en anglais) qui gère le statut (Prêt, Élu, Bloqué) des proccessus.

c. « Monitorer » les proccessus

Sous Linux il y a des commandes indispensables à connaître pour suivre l'état des processus.

- `ps` (process status), elle permet d'avoir un grand nombre d'informations (UID, PID, PPID, TIME ...). Ne pas hésiter à consulter la documentation à l'aide de la commande `man ps`.
- `pstree` l'arbre de filiation des processus.
- `top` évolution en temps réel avec la consommation mémoire et proccesseur.

Parfois on doit intervenir sur les statuts des processus :

- `nice` permet de modifier la priorité d'un processus. La priorité est un nombre, plus ce nombre est petit plus le processus est prioritaire.
- `kill` permet de mettre fin à un processus.

5.3 L'ordonnancement des processus

a. Les priorités

Le système UNIX est multitâche, il exécute les processus à tour de rôle en temps partagé. Le processeur est alloué au processus pour une fraction de temps. Lorsque le processus a écoulé ce temps, ou alors quand il est bloqué sur un évènement, il est suspendu (ses registres sont sauvegardés), mis dans la file des processus en attente et le premier de la file est exécuté.
Un processus suspendu est composé de son espace d'adressage et de son entrée dans la table des processus.
Le système utilise un algorithme d'ordonnancement pour gérer l'allocation du processeur basé sur un recalcul périodique de la priorité des processus en fonction de différents critères.

▶ Exercice 1 D'après bac 2021

Un processeur choisit à chaque cycle d'exécution le processus qui doit être exécuté. Le tableau ci-dessous donne pour trois processus `P1`, `P2`, `P3` :

- la durée d'exécution (en nombre de cycles),
- l'instant d'arrivée sur le processeur (exprimé en nombre de cycles à partir de 0),
- le numéro de priorité.

Le numéro de priorité est d'autant plus petit que la priorité est grande. On suppose qu'à chaque instant, c'est le processus qui a le plus petit numéro de priorité qui est exécuté, ce qui peut provoquer la suspension d'un autre processus, lequel reprendra lorsqu'il sera le plus prioritaire.

Processus	Durée d'exécution	Instant d'arrivée	Numéro de priorité
`P1`	3	3	1
`P2`	3	2	2
`P3`	4	0	3

Compléter le tableau ci-dessous et indiquer dans chacune des cases le processus exécuté à chaque cycle.

`P3`										
0	1	2	3	4	5	6	7	8	9	10

b. Interblocage ou Deadlock

Lors de son exécution, un processus utilise et occupe des ressources : de la mémoire, des fichiers, des disques ainsi que tout type de périphérique.
Un proccesus ne peut utiliser un périphérique uniquement si ce dernier n'est pas occupé. Dans l'exemple ci-dessous le processus `cheese` ne peut pas utiliser la webcam car un autre processus l'occupe.

```
(cheese:13393): cheese-WARNING **: 18:36:48.696:
Peripherique « /dev/ »video0 occupe: gstv4l2object.c(3770):
Call to S_FMT failed for YUYV @ 640x480: Peripherique ou ressource occupe
```

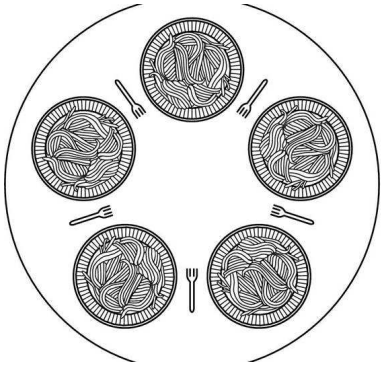
N.B : certaines ressources sont à usage exclusif et d'autres peuvent être à usage partagé.

Dans les faits, le proccesus demande à utiliser une ressource et si c'est possible l'utilise puis lorsqu'il a fini il la libère.
L'interblocage, deadlock ou encore étreinte fatal) est un phénomène qui peut avoir lieu lorsque plusieurs processus se bloquent les uns les autres. Lorsqu'un processus `A` refuse de libéré une ressource 1 car il a besoin d'une autre ressource 2 avant d'achever sa tâche et qu'un autre processus `B` refuse de libérer la ressource 2 car il a besoin de la ressource 1.

▶ Exercice 2 Le « diner des philosophes »

Le problème du « diner des philosphes » est un classique en culture informatique proposé par Dijkstra pour illustrer l'ordonnancement des proccessus et l'allocation des ressources.

Son principe est le suivant :



- cinq philosophes se trouvent autour d'une table ronde ;
- chacun des philosophes a devant lui un plat de spaghettis ;
- à gauche de chaque plat de spaghettis se trouve une fourchette.

Un philosophe n'a que trois états possibles :

- penser pendant un temps indéterminé ;
- être affamé pendant un temps déterminé et fini (sinon il y a famine) ;
- manger pendant un temps déterminé et fini.

Des contraintes extérieures s'imposent à cette situation :

- quand un philosophe a faim, il va se mettre dans l'état « affamé » et attendre que les fourchettes soient libres ;
- pour manger, un philosophe a besoin de deux fourchettes : celle qui se trouve à gauche de sa propre assiette, et celle qui se trouve à droite (c'est-à-dire les deux fourchettes qui entourent sa propre assiette) ;
- si un philosophe n'arrive pas à s'emparer d'une fourchette, il reste affamé pendant un temps déterminé, en attendant de renouveler sa tentative.

1. Que se passe-t-il si chaque philosophe se saisit de la fourchette devant lui ?

2. Le problème ici est de trouver un ordonnancement qui permettrait aux philosophes de manger chacun à leur tour.

(source fr.wikipedia.org)