Lycée Jean Moulin **Draguignan**
NSI T*erminale* Année .....

Systèmes d'exploitation et processus

Introduction

Un système d'exploitation gère l'ensemble des ressources et sa fonction peut se résumer en:

- la gestion des processus;
- la gestion des fichiers;
- la gestion de la mémoire;
- la gestion des périphériques;
- le traitement des entrées-sorties;
- la sécurisation du système.

Quand on lance l'exécution d'un programme à partir d'un OS en cliquant par exemple sur une icône, cela déclenche la création d'un **processus** pour exécuter le programme. D'autres processus peuvent être déclenchés à partir d'un processus (on dira que ce sont des processus-fils).

Certains fonctionnent en permanence en arrière-plan, comme un antivirus et ceux qui s'exécutent dès le démarrage de la machine (les services sous windows ou les daemons sous linux).

On peut dire qu'il y a deux grandes catégories de processus:

- ceux créés par un utilisateur;
- ceux créés par le système.

Programme ≠ Processus

Un programme est une description statique d'une tâche à exécuter.
Un processus est une instance (un exemplaire) d'une tâche en cours d'exécution.

On peut comparer un **programme** à **une recette de cuisine** listant les ingrédients et décrivant la méthode à utiliser pour la réaliser.
Le processus serait alors quelqu'un qui a récupéré la recette, acheté les ingrédients et qui serait en train de réaliser la recette.
De plus à un instant donné, on peut avoir plusieurs personnes dans la même cuisine. Chacun s'occupe d'une recette (la même ou pas...), chacun est sur son propre exemplaire de recette et dans un état d'avancement différent.
Avec cette "organisation" on peut:

- Désigner les tâches individuellement et indépendamment du programme qu'elles exécutent (même si on a plusieurs exemplaires du même programme).
- Agir de façon individuelle sur les différentes tâches sans impacter les autres.
  Par exemple:
  - en mettre certaine en pause.
  - en terminer d'autres.
  - gérer les tâches en fonction de priorités (peut-être que certaines sont plus importantes que d'autres).
  - ...

Comment ça marche ?

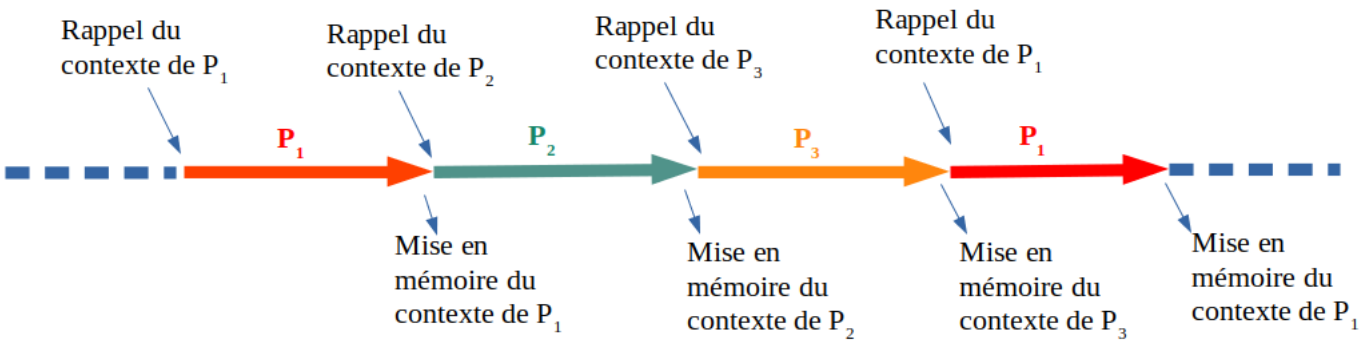
Lors de la création d'un processus, il sera **identifié** par :

- Un numéro (PID : Process Identification)
- Un PPID (le numéro du processus père)
- Un UID, l'identifiant de l'utilisateur

Il lui sera alloué de la **mémoire virtuelle** qui permettra de stocker:

- le code du programme à exécuter;
- les variables globales
- une pile pour stocker les paramètres des fonctions, variables locales, résultats, etc.
- une réserve de mémoire et de quoi stocker le code des bibliothèques si nécessaire...
- une capture de l'état de l'ensemble des registres du processeur sur lequel il s'exécute (**le contexte d'exécution**)

Si le processus est en cours d'exécution, le contexte est mis en mémoire dans les registres du processeur.
Si le processus est mis en pause pour qu'un autre s'exécute, le contexte est mis en mémoire quelque part et sera rappelé dans les registres lorsque le processus aura de nouveau la main.
Voici un schéma qui illustre l'ordonnancement de 3 processus (P_1, P_2, et P_3) avec un seul processeur. Chacun des processus occupe le processeur pendant un certain temps. La mise en mémoire et le rappel des contextes est très important pour que l'exécution d'un processus n'interfère pas avec les autres.

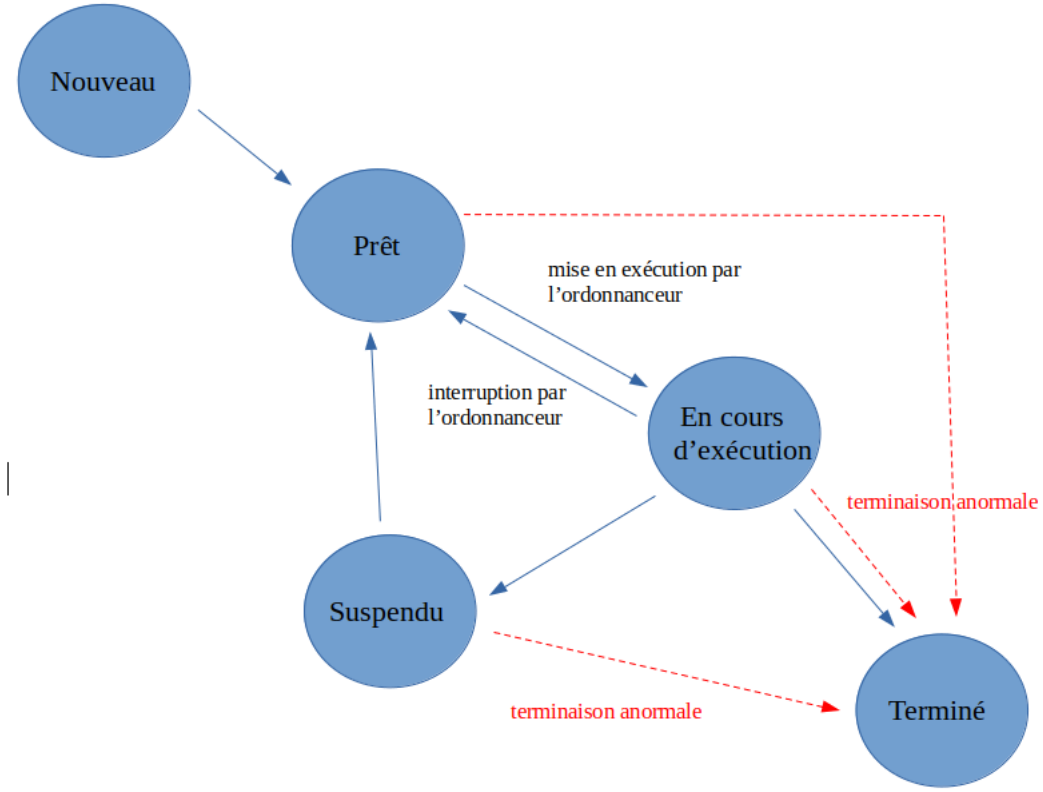


Comme l'exécution est très rapide, on a l'impression que les tâches s'exécutent simultanément, on parle alors de "pseudo-parallélisme". Si la machine possède 3 processeurs, on peut alors imaginer que les processus s'exécutent chacun sur un processeur simultanément...
Pour un utilisateur, quelque soit la machine, il n'a pas à se soucier des quantités de ressources nécessaires, **c'est l'OS qui gère les processus et les ressources nécessaires**.

Chaque processus possède également un **état logique** qui permet de savoir dans quel état est le processus à tout moment:

- l'état **nouveau** (vient d'être créé)
- l'état **prêt** (prêt à s'exécuter)
- l'état **en cours d'exécution**
- l'état **suspendu ou bloqué** (il est en pause car par exemple il attend que l'utilisateur fasse une entrée clavier...)
- l'état **terminé**

Voici un schéma résumant les passages entre états logiques. On peut supposer que le passage de l'état suspendu à l'état terminé est un arrêt anormal...



Terminaison anormale...

Considérons la situation suivante :
Soit 2 processus P1 et P2, soit 2 ressources R1 et R2. Initialement, les 2 ressources sont "libres" (utilisées par aucun processus).
Le processus P1 commence son exécution (état "en cours d'exécution"), il demande la ressource R1. Il obtient satisfaction puisque R1 est libre, l'ordonnanceur le met dans l'état "prêt" pour passer la main au processus P2.
Pendant ce temps, l'ordonnanceur a passé P2 à l'état "en cours d'exécution" : P2 commence son exécution et demande la ressource R2. Il obtient immédiatement R2 puisque cette ressource était libre. P2 repasse immédiatement à l'état "en cours d'exécution" et poursuit son exécution (P1 lui est toujours dans l'état "prêt").
P2 demande la ressource R1, il se retrouve dans un état "suspendu" puisque la ressource R1 a été attribuée à P1. P1 est dans l'état "prêt", il n'a pas eu l'occasion de libérer la ressource R1 puisqu'il n'a pas eu l'occasion d'utiliser R1 (pour utiliser R1, P1 doit être dans l'état "en cours d'exécution").
P2 étant "suspendu" (en attente de R1), l'ordonnanceur passe P1 dans l'état "en cours d'exécution" et avant de libérer R1, il demande à utiliser R2.
Problème : R2 n'a pas encore été libéré par P2, R2 n'est donc pas disponible, P1 se retrouve "suspendu" (les deux processus P1 et P2 ne pourront plus sortir de l'état "suspendu").

Cette situation est ce que l'on appelle une situation **d'interblocage** (deadlock en anglais)
Il existe plusieurs solutions permettant soit de mettre fin à un interblocage (cela passe par l'arrêt d'un des 2 processus fautifs) ou d'éviter les interblocages.

Un peu de pratique...

À faire 1:
Chercher un moyen de visualiser les processus en cours sur un ordinateur.

À faire 2:
Ouvrir le bloc note, trouver le processus correspondant, noter son PID et l'interrompre depuis l'interface de visualisation des processus.
Recommencer l'opération

À faire 3:
Chercher un moyen de visualiser les processus en cours sur votre tablette.

À faire 4:
On considère 3 Processus P1, P2 et P3 et 3 ressources R1, R2 et R3 tels que:

- P1 : demande R1, demande R2, libère R1, libère R2
- P2 : demande R2, demande R3, libère R2, libère R3
- P3 : demande R3, demande R1, libère R3, libère R1

Si les processus sont exécutés l'un après l'autre, d'abord P1 puis P2 et enfin P3 y-a-t-il interblocage ?

.....
.....
.....

Décrire une exécution des 3 processus qui conduit à une situation d'interblocage

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....