Gestion des processus et des ressources par un système d'exploitation

1. Système d'exploitation

Toute machine est dotée d'un **système d'exploitation** (OS, Operating System) qui constitue la couche intermédiaire entre la machine et les applications.

1.1. Rôle du système d'exploitation

Gestion des ressources matérielles - L'OS cache la complexité du matériel, arbitre les requêtes d'accès aux ressources, évite les conflits, traite les interruptions, les entrées-sorties et les erreurs, empêchant les usages impropres de la machine.

Mise à disposition de bibliothèques - L'OS facilite l'utilisation de la machine grâce à un ensemble de bibliothèques, qui rendent la programmation plus facile, plus simple et permet à un même programme de fonctionner sur des machines très différentes (processeurs, nombre de cœurs, mémoire, type de disque dur ou carte réseau) munies du même système d'exploitation.

Exécution des programmes - L'OS charge les programmes depuis la mémoire de masse et gère leurs exécutions simultanées (ou plutôt en alternance). On appelle ces programmes en mémoire des processus.

Gestion de la mémoire - Un gros programme avec ses données est stocké dans une mémoire virtuelle (sur le disque dur). L'OS s'assure que seule la partie utile à l'exécution du programme à l'instant *t* est chargée dans la RAM en effaçant les données non utilisées pendant un certain temps.

Sécurisation - L'OS assure la sécurité globale du système.

1.2. Structure en niveaux

Le système d'exploitation est structuré en niveaux.
Les niveaux positifs sont les niveaux logiciels. Le niveau 3 est constitué des applications. Il communique avec le niveau 2 (drivers) qui communique avec le niveau 1 (autre niveau de drivers) qui communique avec le niveau 0 (le noyau).
Le noyau (kernel en anglais) assure la communication entre les logiciels et le matériel et la gestion des différentes tâches de la machine. Les programmes agissent sur une mémoire virtuelle, pas sur la mémoire physique et le noyau fait le lien entre ces deux mémoires.
Les niveaux négatifs sont les niveaux matériels.

2. Gestion des processus et des ressources

Un processus est un programme en cours d'exécution avec un espace mémoire et des ressources dédiés.
Il est identifié par un numéro PID (Process Identifier). Il dispose de l'identifiant du processus parent qui l'a créé, PPID (Parent Process Identifier), est associé à son propriétaire et a des droits d'accès aux ressources.

2.1. Visualisation des processus

Windows
Le gestionnaire des tâches affiche les processus, avec PID, occupation mémoire et utilisation du processeur. Le bouton « Fin de tâche » permet d'arrêter un processus. Un clic droit permet de modifier la priorité d'un processus.
La commande tasklist renvoie un affichage similaire (tasklist /v pour plus de détail) et permet de chercher un processus particulier avec : tasklist /fi "pid eq 1560".

La commande wmic permet d'obtenir les processus parents (PPID) :

wmic process get processid, parentprocessid

On peut rechercher un PID/PPID particulier : wmic process get processid, parentprocessid | find "6300"

Informations système → Environnement logiciels → Tâches en cours affiche la liste des processus actifs avec leur priorité.

Linux

On affiche des informations sur les processus dans un terminal Linux avec par exemple :

- **ps -aef** ou **ps -aef | less** : avec les PID et PPID des processus actifs
- **pstree** : affichage sous forme d'arbre
- **top** : affichage en temps réel, puis h pour l'aide, q pour quitter, M pour trier par consommation de la mémoire, P par consommation du processeur, i pour le filtrage des processus inactifs ; k permet de mettre fin à un processus (kill), r de modifier la priorité…
- **kill** PID : supprimer un processus, deux options :
  - kill -15 demande la terminaison propre d'un processus en libérant les ressources allouées
  - kill -9 demande la terminaison immédiate (uniquement si kill -15 ne fonctionne plus)

Mini-TP

Utiliser top dans un terminal Linux et ps -aef dans un autre terminal pour observer les PID et les PPID des processus lors des manipulations suivantes.
Ouvrir un navigateur internet, repérer le PID et le PPID du navigateur. Quel est son processus père ?
Ouvrir une autre fenêtre du navigateur et observer le nouveau processus, vérifier son PPID.
Ouvrir différents onglets dans un navigateur, vérifier les nouveaux processus et leur PPID.
Que remarque-t-on ? Exécuter la commande pstree.
Utiliser kill pour terminer les différents processus associés au navigateur en terminant par le processus initial du navigateur.

2.2. Création d'un processus

On crée un processus par exemple en cliquant sur une icône, ce qui appelle la commande CreateProcess() sur Windows ou fork() sur Linux. On peut ainsi lancer plusieurs processus associés à un même programme.

Windows

La fonction CreateProcess() crée un nouveau processus.

Linux

Un processus peut créer un autre processus avec la commande fork(). On le transforme ensuite en un programme différent avec la commande exec().
Le tout premier processus, appelé processus 0 ou Swapper, PID = 0, est créé au démarrage de l'ordinateur. Il crée un processus init ou systemd, PID = 1, qui va créer les autres processus.

Priorité

Les processus ont une priorité allant de 0 (plus prioritaire) à 39 (moins prioritaire).
On peut régler la priorité des processus avec **nice** au lancement : nice *-niceness commande* va lancer une commande (donc un nouveau processus) en ajustant sa priorité d'une valeur *niceness* allant de -20 à 19.
Par défaut, cette valeur d'ajustement vaut 0 ; la priorité du processus est alors égale à 20.
Avec une valeur *niceness* de 19 (nice -19 *commande*), la commande va s'exécuter avec une priorité de 39, donc très faible.
On doit être en mode administrateur pour exécuter la commande nice avec une valeur négative, ce qui correspond à une priorité au-dessus de la moyenne : sudo nice --15 *commande* correspond à une priorité égale à 20 – 15 = 5.

On modifie la priorité d'un processus en cours d'exécution avec **renice** *niceness PID*.

2.3. Ordonnancement des tâches (scheduling)

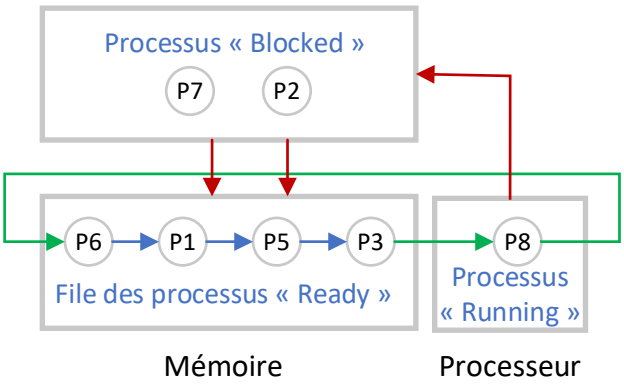
Tous les systèmes d'exploitation actuels sont multitâches : ils gèrent l'exécution de plusieurs processus simultanément. En réalité, ce n'est pas du vrai simultané, mais de l'alternance. Un cœur ne peut exécuter qu'une tâche à la fois et le système partage les cœurs disponibles entre les tâches prêtes.

L'ordonnanceur (scheduler en anglais) est la partie de l'OS qui sélectionne le processus à exécuter.

2.3.1. Etats d'un processus

Un processus actif peut être dans trois états principaux :

- Prêt ou En attente (Ready) : le processus est prêt et attend son tour
- Élu ou Exécution (Running) : le processus est en cours d'exécution par le processeur
- Bloqué (Blocked) : le processeur est bloqué en attendant une ressource, il repasse à Ready dès que la ressource est disponible



Un processus peut aussi s'endormir en attendant un évènement (réaction de l'utilisateur, arrivé d'un message réseau, accès disque…). Il est alors en attente passive sur une file d'attente spéciale. Quand l'évènement se produit, il revient dans la file des processus prêts (Ready).
Cette attente passive est bien mieux qu'une attente active, avec une boucle potentiellement infinie, qui consomme du temps CPU et gaspille des ressources.
Il existe aussi un état Initialisation (avant Prêt) et Terminé.

Le système peut prendre la main de force sur un processus qui consomme trop de temps CPU (préempter) pour permettre aux autres processus de s'exécuter.

2.3.2. Commutation de contexte (context switch)

Le **contexte**, ou l'état, d'un processus est l'ensemble des informations qui lui sont associées : adresse mémoire de l'instruction en cours, valeurs des registres, données en mémoire...

Le changement de processus actif est accompagné d'un changement de contexte avec une sauvegarde du contexte du processus évincé, qui passe de Running à Ready ou Blocked, et un chargement du contexte du nouveau processus qui passe de Ready à Running.

2.4. Risque d'interblocage

Exemple
Un processus P1 détient une ressource R1 et a besoin d'une ressource R2 pour poursuivre. Un processus P2 détient la ressource R2 et a besoin de la ressource R1 pour poursuivre. Les deux processus sont bloqués. Il s'agit d'un interblocage (deadlock en anglais).

Cette situation s'étend à davantage de processus : P1 détient R1 et nécessite R2, mais R2 est détenue par P2 qui nécessite R3, qui est détenue par P3 qui nécessite R1 …

Le système est alors bloqué. On parle d'interblocage (deadlock).

Les algorithmes d'ordonnancement de l'OS doivent gérer ces situations pour éviter l'interblocage.

Stratégies contre l’interblocage

Afin d’éviter ces situations, un processus peut libérer les ressources lorsqu’il est en attente, ou bien ne pas bloquer l’accès aux ressources qu’il utilise, ou alors en cas de blocage, l’OS peut mettre fin à un processus bloqué, ce qui libérera automatiquement les ressources. Un processus peut aussi réserver simultanément toutes les ressources dont il a besoin, puis les relâcher simultanément quand il a terminé.