Processus

Une commande sous linux crée un processus en mémoire, ce processus est sous la responsabilité du kernel. Tous les traitements effectués par une commande sont en fait traitées par le processus qui est créé lors de l'exécution de cette commande.
Un processus est exécuté (en général) par un utilisateur, il disposera donc des mêmes droits que l'utilisateur qui est responsable de son exécution. Et en général, un utilisateur ne pourra pas agir sur un processus lancé par un autre utilisateur, exception faite de root, évidemment.
La commande qui permet de lister les processus est PS :

Lbd1PtoqQp1cEJPq-image-1560342140897.png

PID : Numéro du processus (unique)
TTY : Nom du terminal dans lequel se trouve le processus
TIME : Le temps processeur utilisé par le processus
CMD : La commande qui a créé le processus
On peut déduire de cet exemple, qu'une même commande peut créer plusieurs processus.

Avec la commande ps x on peut voir dans quel état sont les processus

JxdXA3quFVYKyFvM-image-1560342169855.png

S: en sommeil
D : Sommeil irréversible
R : En cours d'exécution
T : Stoppé
Z : Zombie (processus terminé, mais pas encore totalement libéré)

Pour l'instant, on ne visualise que les processus de l'utilisateur, pour obtenir la liste complète, on va utiliser plutôt la commande :
ps -ef (-e tous les processus, et -f pour full détails)

LQNKttBSBRO32vyF-image-1560342216008.png

Processus Foreground

Lorsque vous exécutez une commande comme ps -ef | less, le processus créé est foreground ou d'avant-plan. C'est-à-dire que le shell est bloqué durant l'exécution de la commande. L'utilisateur n'a donc plus la possibilité de faire autre chose. C'est pratique si le traitement est court, mais pas bon du tout si le traitement est long et coûteux en ressource.
Lorsqu'un processus a besoin de lancer un autre processus pour ses traitements, le premier processus est appelé processus parent et le ou les processus créés par le parent sont appelés processus enfant.

Lorsqu'un parent lancé en foreground crée un enfant, le processus enfant bloquera le parent jusqu'à la fin de ses traitements, puis à sa mort, le parent reprendra le relais.

Par exemple, le bash est un processus. Quand on lance une commande dans le bash, le processus parent bash est bloqué jusqu'à la fin des traitements de son processus enfant (par exemple le ls)
Si vous voulez éviter de bloquer le shell (ou le parent). Il est possible de lancer une commande en tâche de fond, dans ce cas, l'enfant rendra immédiatement les droits de continuer les opérations au parent, tout en continuant de faire ses traitements.
Pour se faire, on utilisera le caractère &, par exemple :

zQbTuQ1BXXx8yiXd-image-1560342239358.png

ici :
[1] représente le numéro du travail ou "job".
1342 représente le PID créé pour ce travail
Ces deux informations apparaissent automatiquement lors de la création d'un processus en tâche de fond
[1] Fini signifie que le job 1 est terminé et bash rappel quelle commande avait été exécutée.

Déplacer un processus

Si l'on fait un script, dont une commande demande à dormir 10.000 secondes, mais malheureusement, j'ai oublié de demander à la commande du script de se lancer en tâche de fond

xj0waeDMUsroxv2j-image-1560342261378.png

- Le ctrl+z permet d'interrompre le processus (suspend)
- on fait la commande bg, qui passe le processus en tâche de fond
- on le remet en foreground avec la commande fg
- le ctrl+c permet de lancer une interruption logiciel qui envoi un SIGTERM au processus en foreground

Si vous voulez connaître la liste des jobs en cours de traitement, vous pouvez utiliser la commande jobs

hiojVFSCGr3bJ3sS-image-1560342288221.png

Signaux

Un signal est un message envoyé à un processus pour altérer son état (par exemple stop, start, pause)
Certains signaux peuvent s'envoyer avec un raccourcis clavier (ctrl+z stoppe un processus)
Pour consulter la liste des signaux, kill -l

62mUMypgw0MpcZp6-image-1560342318478.png

Donc la commande kill permet d'envoyer un signal à un processus de plusieurs façons :
- Kill -2 → envoi le signal 2 (SIGINT) à un processus

- Kill -SIGINT → envoi le signal SIGINT à un processus
Le SIGINT est l'équivalent du CTRL+Z (stoppe le processus mais ne le détruit pas)
Pour tuer (détruire) un processus, on utilisera plutôt le SIGTERM (15)

ZlJtruDbGTgTRRBc-image-1560342340711.png

Killall - tuer un processus par user
La commande killall -u permet de demander à tuer tous les processus appartenant à un utilisateur.

killall -u root
Supprimera tous les processus lancés par le compte root.

Conserver un processus en quittant

Quand un utilisateur se déconnecte de sa session, tous les processus rattachés à l'utilisateur reçoivent un SIGHUP (1) qui va finalement tuer
tous les processus rattachés.
Si l'opération de sauvegarde est très longue et que l'administrateur doit se déconnecter, sa sauvegarde s'arrêtera par exemple.
Pour éviter ce phénomène, on utilisera la commande nohup

Priorité d'un processus

Un processus a besoin de temps processeur pour faire ses calculs. Certains d'entre eux ont droits à plus de CPU que d'autres, ceci se règle
avec un mécanisme de priorité. Les processus lancés par le système auront en général plus de priorité que les autres.
C'est le kernel qui s'occupe d'ajuster dynamiquement la priorité et essaie de fournir les bonnes priorités selon les besoins.
L'utilisateur pourra influencer dans une certaines mesure cette priorité en faisant varier une valeur de "gentillesse" niceness !
niceness -20 => Donne la priorité la plus haute possible
niceness 0 => valeur par défaut
niceness 19 => priorité la plus basse
Un utilisateur standard aura le droit de régler l'indice entre 19 et 0. Seul le root peut faire des réglages entre -1 et -20.
Pour affecter un indice de gentillesse, on utilise la commande nice avec l'option -n (régler la gentillesse) et une valeur.

LaEZTEmqkU47cYC6-image-1560342369540.png

Sur les captures précédentes, on a vu des + et des - à côté des jobs, les jobs avec un + est le dernier job à avoir été placé en tâche de fond.
Celui avec un - est l'avant-dernier.

Vous pourrez utiliser la commande TOP pour lister les tâches en cours.

Revision #1
Created 2019-10-31 12:06:46 UTC by Cécile
Updated 2019-10-31 12:07:03 UTC by Cécile