10 - Les processus

Laurent Tichit

8 avril 2011

Plan

Les Processus

Les Processus

Toute exécution d'un programme est un processus.

- Programme = code machine sur le disque dur (aspect statique)
- Processus (tâche) = code en cours d'exécution sur le processeur (aspect dynamique)

Identification et création de processus

- Tout processus actif est identifié par un nombre, le **PID** (*Process IDentification*).
- Un processus ne peut être crée que par un autre processus.
- Chaque processus connaît donc le *PID* de son processus parent : le **PPID**.

La commande **ps**

La commande **ps -u $USER** indique un minimum d'informations sur tous les processus de l'utilisateur. Avec 4 colonnes :

- **PID** = Process IDentifier
- **TTY** = terminal auquel la commande est rattachée
- **TIME** = cumulative CPU time
- **CMD** = commande

La commande **ps**

La commande **ps -elf** indique un maximum d'information sur tous les processus du système. La colonne :

- **F** = indique (4 ou 5) que le processus est en mode super-utilisateur.
- **S** = D : attente d'entrée/sortie, R : running, S : sleeping, T : stopped, Z : defunct ($|R|$ <= nombre de CPU).
- **UID** = proprietaire effectif
- **PID** = Process IDentifier
- **PPID** = Parent PID
- **C** = % CPU utilisé
- **PRI** = priorité
- **NI** = nice (utilisation CPU réduite)
- **SZ** = taille totale en mémoire
- **WCHAN** = addresse de la fonction du noyau dans laquelle le processus est en attente
- **STIME** = date de création du processus.
- **TTY** = terminal auquel la commande est rattachée
- **TIME** = cumulative CPU time
- **CMD** = commande

Exécution de la commande **ps -lf**

Indique toutes les information sur les processus rattachés au terminal.

```
F S UID      PID PPID C PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 S tichit  3733 3731 0  80   0 -  4888 wait   15:12 pts/0    00:00:00 [bash]
0 S tichit  3745 3733 0  80   0 - 40108 select 15:12 pts/0    00:00:38 emacs Uni
0 S tichit  4978 3733 0  80   0 - 15532 poll   16:02 pts/0    00:00:02 xpdf Unix
0 S tichit  5398 3733 1  80   0 - 17470 poll   16:29 pts/0    00:00:17 xpdf Unix
0 R tichit  5833 3733 0  80   0 -  3793 -      16:56 pts/0    00:00:00 ps -fl
```

La commande **pstree**

Permet de représenter l'arborescence des processus.

- **-p** : afficher le PID
- **-n** : trier par PID
- **-u** : afficher le nom du propriétaire
- **-u login** : n'afficher que les processus de *login*

Faire communiquer les processus

- Il est possible de faire cohabiter (et s'exécuter) à un moment donné plusieurs processus.
- Ces processus peuvent s'échanger des données.
- La façon la plus courante est la suivante : le processus 2 va lire ce que le processus 1 écrit.
- Ce mode de communication s'appelle un *tube* (en anglais *pipe*). Sous le Shell, le symbole | permet de créer un *pipe*.
- Par exemple : **ls -aRl | wc -l** crée deux processus :
  - le premier (**ls -aRl**) liste de manière recursive le contenu du répertoire courant.
  - Sa sortie est redirigée vers l'entrée du second processus (**wc -l**) (*word count*) qui lui, compte les lignes.

Exécution concurrente des processus

Notons que les processus exécutés de manière concurente, qu'ils échangent ou non des informations, sont crées "simultanément" et synchronisés par le système :

- Suspendre le producteur lorsque le tube est plein.
- Suspendre le consommateur lorsque le tube est vide.

Fonctionnement :

1. Le Shell crée les 2 processus
2. Redirige la sortie standard du premier vers l'entree standard du second
3. Le système démarre l'exécution des 2 processus.

Contrôler ses programmes

La commande **kill -numero PID** permet d'envoyer un signal à un processus. Par défaut, le signal envoyé est le signal de terminaison.

- Tenter de détruire le processus 66434 :
  `tichit@iml230:~/Test$` **kill 66434**
- Forcer la destruction du processus 66434 :
  `tichit@iml230:~/Test$` **kill -9 66434**
- Tuer tous les processus de nom `firefox` :
  `tichit@iml230:~/Test$` **killall firefox**

Pour tuer tous les processus autorisés, on passe **-1** comme numéro de processus :

- Forcer la destruction de tous mes processus : **kill -9 -1**

Occupation mémoire et CPU

- La commande **free** affiche l'état de la mémoire (disponible, utilisée, libre ...) pour chacun des segments.
- La commande **uptime** affiche la charge du processeur sur les dernières minutes (1, 5, 15).

Suivi des processus

- La commande **top** affiche une page périodiquement remise à jour sur les processus qui tournent. Il faut taper **q** pour quitter). Cette commande permet :
  - De gérer les processus
  - D'être informé de la charge de travail du CPU
  - D'être informé de l'utilisation mémoire.

  Elle permet aussi de pouvoir tuer le processus (commande interactive **k**, de trier par colonne (> **ou** <), etc.

Commande **jobs**

Permet de connaître le status des commandes lancées depuis le Shell. Elle permet de connaître :

- Leur numéro de job
- Leur état (Running, Stopped, Sleeping)
- Leur nom

Avant- et arrière-plan

- Par défaut une commande est lancé en *avant-plan* : le Shell attend la fin de son exécution, puis réaffiche le *prompt*.
- Il est possible de lancer une commande en *arrière-plan*. Celle-ci sera donc exécuté *en parallèle*, et le Shell continuera son exécution.

Avant-plan

Une commande en avant-plan (*foreground*) :

- Bloque (endort) le Shell
- Peut communiquer de manière interactive avec le clavier (peut *lire* au clavier)

Pour faire passer l'exécution d'une commande en arrière-plan, il faut :

- l'endormir : signal **CTRL-Z** au clavier
- lancer la commande **bg** (ou la commande **bg numéro-de-job**).

Arrière-plan

Une commande en arrière-plan (*background*) :

- Ne peut pas lire à partir du clavier
- Ne bloque pas le Shell (c'est lui qui continuera à communiquer de manière interactive avec le clavier)
- Pour qu'une commande exécutée en arrière-plan passe en avant-plan, il exite la commande **fg** (ou la commande **fg numéro-de-job**).
- Pour lancer une commande directement en arrière-plan, il faut la faire suivre de l'opérateur **&**. Ex :
  `tichit@iml230:~$` **nedit &**

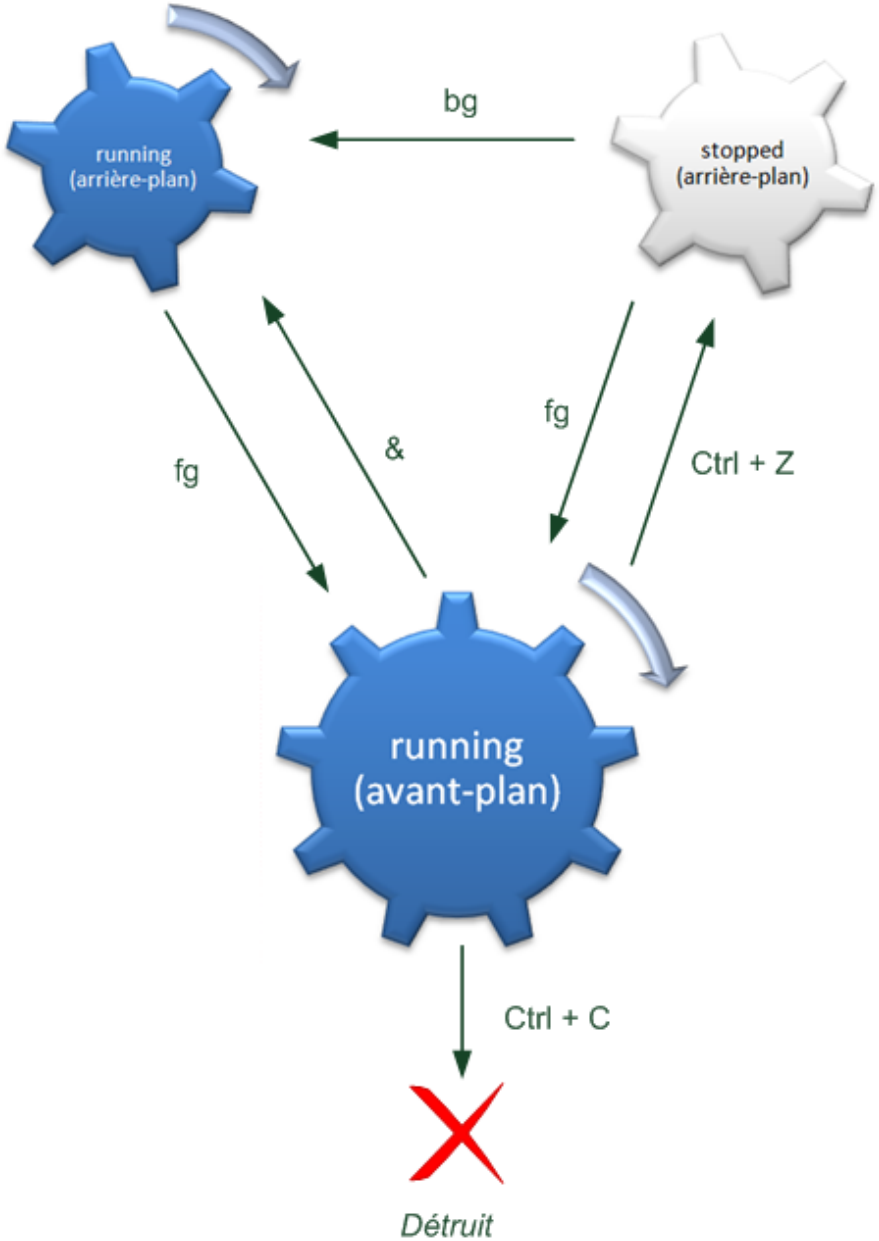
Signaux clavier

Il y a deux signaux que l'on peut envoyer à un processus à partir du clavier :

- Le signal *sleep* grâce à **CTRL-Z**, gèle le processus
- Le signal *term* grâce à **CTRL-C**, tue le processus.

Ne pas confondre ces caractères de contrôle qui envoient des signaux et qui fonctionnent avec TOUS les processus avec le caractère de fin de fichier **CTRL-D** qui implique la fin d'un traitement et parfois, aussi, la fin d'un processus.
On ne peut envoyer un signal par l'intermédiaire d'un contrôle-clavier qu'aux processus s'éxecutant en avant-plan.

Avant- et arrière-plan



Exécution Parallèle / Séquentielle

- Par extension, pour lancer plusieurs commandes en parallèle :
  commande1 & commande2 & commande3 & commande4 &
- Pour lancer plusieurs commandes séquentiellement :
  commande1 ; commande2 ; commande3 ; commande4

Propriétaire d'un processus

Tout processus possède deux propriétaires :

- le propriétaire **réel** (celui à qui appartient le fichier)
- le propriétaire **effectif** (celui qui exécute le programme du fichier)

... et deux groupes propriétaires :

- le groupe propriétaire **réel** (celui auquel appartient le fichier)
- le groupe propriétaire **effectif** (le groupe du propriétaire effectif)

Exemple : Soit le fichier exécutable suivant (obtenu par ls -lg)

```
-rwx--x--x 2 tichit enseign 12098 Aug 12 11:57 executable
```

Si le programme est exécuté par *marc* du groupe *eleve*, le processus appartient :

- réellement à *tichit* du groupe *enseign*
- effectivement à *marc* du group *eleve*

Privilèges

Deux cas se présentent :

- Par défaut, le bit *setuid* (respectivement *setgid*) n'est pas positionné : le processus sera exécuté avec les droits de son propriétaire (respectivement groupe propriétaire) **effectif**, celui qui demande l'exécution.
- Le bit *setuid* (respectivement *setgid*) est positionné. Le processus sera exécuté avec les droits de son propriétaire (respectivement groupe propriétaire) **réel**, celui qui possède le fichier.

La commande **chmod u+s fichier** (**chmod g+s fichier**) permet de positionner le bit setuid (setgid). Il faut au préalable que le fichier soit en mode exécutable (**u+x** ou **g+x**). On peut aussi utiliser directement **chmod 4000 fichier** (**chmod 2000 fichier**).
Attention, positionner le bit setuid peut créer une faille de sécurité (en particulier sur des scripts Shell).

Les bits "setuid" et "setgid"

Comme il est dangereux de conserver des fichiers exécutables pour lesquels les bits setuid ou setgid sont positionnés, on peut essayer de rechercher de tels fichiers grâce à **find** :

find /usr/bin -type f -a \(-perm -4000 -o -perm -2000 \) -exec ls -l \;

- **-type -f** : Ne prendre en compte que les fichiers standards.
- **-a** : conjonction entre **-type** et le contenu de la parenthèse.
- **\(-perm -4000 -o -perm -2000 \)** : disjonction entre les tests sur suid et sgid.

Exemple

```
-rwxr-sr-x 1 root tty 14632 2008-11-04 23:11 /usr/bin/bsd-write
-rwsr-xr-x 1 root root 41784 2009-04-04 07:50 /usr/bin/chfn
-rwsr-xr-x 1 root root 15168 2007-12-10 20:03 /usr/bin/arping
-rwsr-xr-x 1 root lpadmin 14632 2009-04-23 13:32 /usr/bin/lppasswd
-rwsr-xr-x 2 root root 131040 2009-02-17 04:24 /usr/bin/sudoedit
-rwsr-xr-x 1 root root 28208 2009-04-04 07:50 /usr/bin/newgrp
-rwxr-sr-x 1 root utmp 371936 2009-02-11 13:29 /usr/bin/screen
-rwxr-sr-x 1 root shadow 23984 2009-04-04 07:50 /usr/bin/expiry
-rwsr-xr-x 1 root root 42776 2009-04-04 07:50 /usr/bin/passwd
-rwxr-sr-x 1 root tty 14744 2009-02-18 22:42 /usr/bin/wall
-rwxr-sr-x 1 root ssh 93536 2009-01-28 22:25 /usr/bin/ssh-agent
-rwxr-sr-x 1 root utmp 404208 2009-02-27 05:23 /usr/bin/xterm
-rwxr-sr-x 1 root mail 15128 2008-11-18 11:46 /usr/bin/dotlockfile
-rwsr-sr-x 1 daemon daemon 48376 2009-04-17 09:52 /usr/bin/at
-rwsr-sr-x 1 root root 10472 2009-04-03 09:44 /usr/bin/X
-rwxr-sr-x 1 root mlocate 35352 2008-11-04 23:08 /usr/bin/mlocate
-rwsr-xr-x 1 root root 51256 2009-04-04 07:50 /usr/bin/gpasswd
-rwsr-xr-x 1 root root 32952 2009-04-04 07:50 /usr/bin/chsh
-rwxr-sr-x 1 root shadow 58904 2009-04-04 07:50 /usr/bin/chage
```