Formation Unix/Linux (3)
Le Shell: gestion des processus, redirection d'entrée/sortie

Olivier BOEBION

Mars 2004

1 Définition

Un programme se compose d'un ou plusieurs fichiers composés d'instructions. Ces fichiers sont stockés sur un support comme un disque dur ou un CDRom. Un programme est une entité passive. Lorsque le programme est lancé, un processus est créé par le système d'exploitation. Un processus est une entité active et possède des caractéristiques (PID, PPID, UID...) qui peuvent varier dans le temps (priorité, adresse de l'instruction suivante, par exemples).

2 L'exécution d'une commande UNIX dans un shell

La gestion des processus par le système d'exploitation **UNIX** est dite hiérarchisée. Lorsque vous tapez une commande externe au shell, (cad qu'un fichier binaire correspondant à la commande est chargé ; par exemple : ls) un processus fils du shell est créé et la commande s'exécute dans celui-ci. Le processus père (le shell) attend que la commande soit exécutée pour redevenir actif. Deux appels systèmes sont en fait utilisés : **fork** et **exec** et on peut modéliser la procédure avec la figure 1.

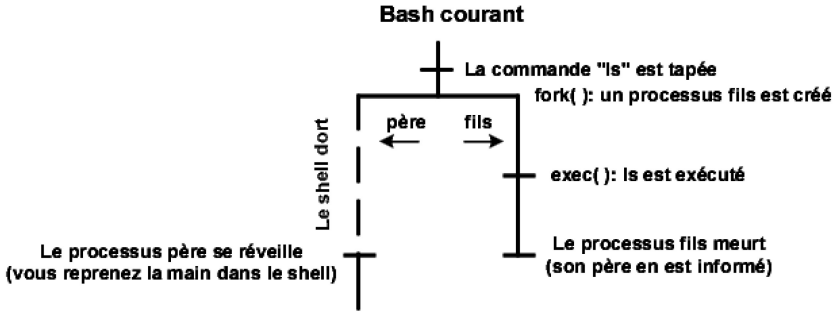

Fig. 1 – Lancement d'une commande

Il est possible d'obliger l'exécution d'un processus dans le shell courant. Pour cela, le lancement de la commande se fera à travers la commande interne **exec** : **exec sleep 10**, par exemple. De même, dans le cas d'un script, on peut utiliser le point ou la commande interne **source**. Par exemple, l'exécution du fichier **.bash_profile** dans le shell courant se fera de la manière suivante : **. .bash_profile**

3 Caractéristiques des processus

3.1 Caractéristiques générales

Tout comme les fichiers, un système **UNIX** identifie les processus grâce à un nombre. Cette information s'appelle le **PID** (**P**rocess **ID**entificator). Comme un fichier, un processus a un propriétaire. L'analogie s'arrête ici. Toutes les informations concernant les processus s'obtiennent avec la commande **ps**. Il existe une version *System V* et une version *BSD* de cette commande. Nous ne verrons que quelques options BSD (vous pouvez retrouver l'ensemble des options en consultant le manuel en ligne).

En tapant la commande **ps axl**, vous allez obtenir des informations concernant tous les processus :

- le propriétaire (**UID**) ;
- le numéro du processus (**PID**) ;
- le numéro du processus père (**PPID**) ;
- la priorité (**PRI**) ;
- la valeur de NICE (**NI**) ;
- l'état (**STAT**)
- le nom du device vers lequel il est dirigé (**TTY**) ;
- le temps passé dans la cpu (**TIME**) ;
- le nom de la commande exécutée (**COMMAND**).

Si vous ajoutez l'option "**f**, les liens de parenté entre les différents processus apparaitront plus facilement. Notez que toutes les processus sont lancées par le processus **init** dont le **PID** est **1**. Le processus père du processus **init** n'est autre que le noyau (PID égal à 0). Vous pouvez utiliser la commande **pstree** pour visualiser d'une seconde manière l'arborescence des processus.

Si vous n'êtes intéressé que par vos processus, la commande **ps fux** vous affichera ceux-ci.

3.2 Etat d'un processus

L'état d'un processus peut être les suivants :

- **D** : le processus est ininterruptible ;
- **R** : le processus est en train de s'exécuter ;
- **S** : le processus est endormi ;
- **T** : le processus est stoppé ;
- **Z** : le processus est zombi : il est mort mais son père ne le sait pas.

D'autres informations peuvent aussi apparaître :

- **W** : le processus a été swappé ;
- **<** : le processus possède une priorité haute ;
- **N** : le processus possède une priorité basse ;
- **L** : le processus possède des pages mémoires verrouillées.

3.3 Les entrée/sorties d'un processus

Chaque création de nouveau processus s'accompagne de la création d'une table de descripteurs de fichier. Trois entrées sont créé par défaut : **STDIN**, **STDOUT** et **STDERR**. Ces descripteurs correspondent respectivement à l'entrée standard (numéro logique **0**), à la sortie standard (numéro logique **1**) et à la sortie d'erreur standard (numéro logique **2**). Par défaut, ces fichiers sont affectés au clavier pour les entrées et à un TTY pour les sorties mais il est possible de les rediriger.

La redirection d'entrée se fait avec le symbole <, celle de sortie avec les symboles > et >> et celle de sortie d'erreur avec les symboles 2 > et 2 >>. Le tableau 1 vous donne quelques exemples possibles.

Commande	**Action**
$ mail user3 < fic	envoi du fichier fic à l'utilisateur user3 par un mail
$ ls > fic	envoi du résultat de la commande ls dans le fichier fic
$ date >> fic	ajout du résultat de la commande date dans le fichier fic
$ find / -name toto 2>/dev/null	recherche d'un fichier et envoi des erreurs retournées dans le fichier null (poubelle)
$ cp *.* ~/tmp 2>>erreur.log	ajout des erreurs retournées par cp dans le fichier erreur.log (archivage d'erreur)

Tab. 1 – Redirection d'entrée/sortie

Bien entendu, il est possible de cumuler plusieurs redirections :

$ **find /home -name .bash_profile > resultat 2> erreur**

3.4 Le code retour d'un processus

L' exécution de toute commande **UNIX** se termine par l'envoi d'un code retour au processus père. Celui-ci indique le bon déroulement ou non de la commande. En général, un code retour égal à **0** indique un bon déroulement alors que**1** indique une erreur de syntaxe et **2** une erreur d'emploi de la commande.

La variable $? du shell permet d'afficher le code retour de la dernière commande exécuté.

4 Gestion des processus

4.1 Background et Foreground

Vous avez remarqué que lorsque vous lancez une commande UNIX, vous ne récupérez la main que lorsque l'exécution de cette commande est terminée. Grâce au multi tâche, il est possible de faire exécuter celle-ci en tâche de fond et ainsi de récupérer le prompt immédiatement. Pour cela, vous utiliserez le caractère **&** à la fin de la commande. Toutefois, il faudra dans ce cas rediriger la sortie standard dans un fichier :

$ find /home -name .bash_profile > resultat.txt &

La commande **jobs** vous permet d'obtenir la liste des processus en tâche de fond. Il est possible de passer un processus de l'arrière plan au premier plan avec la commande **fg**, de la stopper en tapant **CTRL-Z** ou encore d'utiliser la commande **bg** pour passer un job en background.

Le paramètre des commandes **bg** et **fg** est soit un **PID** soit un numéro retourné par la commande **jobs**. Dans ce dernier cas, le numéro sera précédé du symbole %.

4.2 Les signaux

Le système communique avec les processus à l'aide de signaux. Par exemple, si vous lancez une commande et que vous tapez les touches **CTRL-Z**, le processus en cours recevra le signal numéro **24** (**SIGSTOP**) et stoppera son traitement. Une déconnexion provoquera l'envoi du signal **1** (**SIGHUP**) à tous les processus. Lorsque vous tapez **CTRL-C**, vous envoyez un signal **2** (**SIGINT**) au processus courant.

Si un processus reçoit un signal auquel aucun traitement n'est associé, il meurt. Il est possible de définir ce traitement associé ou d'ignorer un signal. Ceci reste cependant impossible pour certains signaux.

Un utilisateur peut envoyer un signal à un processus avec la commande **kill**. Celle-ci envoie par défaut le signal numéro **15**. La syntaxe est :

$ **kill -*numéro* PID**
ou
$ **kill -*nom_du_signal* PID**

Vous pouvez aussi utiliser une syntaxe utilisant le symbole % suivi d'un numéro de processus issu de la commande **jobs**. Pour tuer un processus, la commande **kill** avec le **PID** comme seul argument suffit normalement. Afin d'être sûr d'éliminer un processus, l'envoi du signal **9** est requis.

4.3 Détachement d'un processus

Si vous lancez un long traitement sur une machine **UNIX**, le processus sera tué si vous vous déconnectez. Il existe un moyen de laisser s'exécuter un programme après une déconnexion et de visualiser les résultats stockés dans un fichier. Pour cela, vous utiliserez la commande **nohup** avant votre commande et vous redirigerez les résultats de votre processus dans une fichier.

4.4 La commande trap

Elle permet d'ignorer des signaux ou de leur associer un traitement particulier. Le tableau 2 donne quelques exemples. La syntaxe de cette commande est :

$ trap '*commande*' *numéro_signal*

Commande	**Action**
$ trap " 2	ignore le signal 2
$ trap 2	restaure le traitement par défaut
$ trap	liste les signaux piégés
$ trap 'echo bonjour' 2	exécute echo bonjour dès réception du signal 2

TAB. 2 – Exemples de la commande trap

4.5 Priorité des processus

Il est possible de jouer sur la priorité d'exécution des processus avec la commande **nice**. Celle-ci se place avant le programme à exécuter en précisant une valeur. Ces valeurs sont comprises entre 0 et 19 pour un simple utilisateur du système et entre -20 et 19 pour le super utilisateur. La valeur **0** correspond à la valeur par défaut, **19** à la priorité la plus basse et **-20** à la priorité la plus haute.

Si le processus est déjà chargé en mémoire, il reste possible de modifier la valeur de "nice" avec la commande **renice**. Le tableau 3 vous donne quelques exemples.

Commande	**Action**
$ nice 19 find / -name pdf	lance la commande find avec priorité la plus basse
$ renice −20 2332	place le processus dont le PID est 2332 en priorité la plus haute (seul root peut le faire)

TAB. 3 – Exemples des commandes nice et renice

Vous pouvez visualiser l'évolution du taux d'occupation dans la CPU des processus en temps réél avec la commande **top**. Une autre commande utile est la commande **time** qui permet, en la placant devant une commande, de vous indiquer les temps d'exécution de celle-ci. Trois temps vous sont délivrés :

- le temps réel passé entre le début et la fin d'exécution (real) ;
- le temps passé en mode utilisateur (user) ;
- le temps passé en mode noyaux (sys).

Exercices

Compréhension générale

Tapez la commande "exec sleep 2". Que se passe t'il et pourquoi ? Affichez ensuite tous les processus vous appartenant. Lancez une commande "sleep 600". Expliquez ce qui se passe :

- *pour le processus père (le shell) ?*
- *pour le processus "sleep" ?*

Processus en tâche de fond

Si le processus "sleep 600" précédent n'est pas encore terminé, comment faites vous pour reprendre la main dans le shell et placer le processus "sleep" en background ? Quelle commande aurait il fallu taper pour obtenir directement ce résultat ? Tuez ensuite le processus "sleep".

Redirection de sortie et code de retour

*Comment stocker dans un fichier les résultats donnés par la commande "**find / -size +500k**" ? Que s'affiche t'il à l'écran ? Comment stocker ces lignes affichées dans un seconf fichier ?*

Comment expliquez vous que le code retour de la commande find soit égal à 1 ?

Détournement de signaux

Trouvez une commande qui permette de lancer une commande ps lorque vous tapez CTRL-C dans votre shell courant. Comment faites vous pour lancer la commande ps à partir d'un autre terminal ?

Détachement

Lancez une commande "sleep 600" de manière à ce que son exécution continue après votre déconnexion. Que constatez-vous lorque vous vous reconnectez ?