TP 1 : Manipulation de processus

Février 2014

Objectifs :
- améliorer ses relations avec le système UNIX ;
- observer et manipuler les processus lourds UNIX.

1 Hiérarchie de processus

Lorsque vous vous êtes connecté sous UNIX, vous avez lancé un processus, c'est-à-dire l'exécution d'un programme particulier, le *shell de login* (interpréteur de commandes). Ce processus (comme tout processus) a un numéro d'identification appelé *Process-ID* (PID). Chaque nouvelle commande lancée par la suite nécessite un nouveau processus. Ce nouveau processus est alors considéré comme le « fils » du processus à partir duquel il a été créé, ici le shell de login.
Ainsi ce nouveau processus possède un processus père (repéré par son *Parent Process-ID* (PPID). Cette commande peut elle-même lancer une nouvelle comande, son processus devient le père du nouveau créé, et ainsi de suite . . .
L'ensemble des processus de la machine est donc organisé en une hiérarchie « père en fils ». Les processus possèdent tous le même ancêtre, le processus numéro 1. La commande `ps -f` (`f` pour *full*) permet de visualiser la hiérarchie de vos propres processus.
La colonne PID est le numéro du processus, la colonne PPID est le numéro du père de ce processus.

```
ermont@leia:~$ ps -f | more

ermont   1073   1071   0 22:48:20 pts/2     0:00 bash
ermont   1086   1073   0 22:48:52 pts/2     0:00 more
```

Ici la commande more possède comme père le processus de login (`bash`). La commande `ps -e` affiche des infos sur tous les processus du système. Faire `man ps` pour voir toutes les options de la commande `ps`.

▷ **Exercice 1 : Observation des processus**
Observez les processus que vous avez engendrés, explicitement ou non. Analysez certaines de leurs caractéristiques ainsi que leurs relation père/fils.

2 Destruction de processus

La commande `kill` permet d'envoyer un signal à un processus. Les signaux les plus utilisés sont par exemple :

`SIGINT` - interruption du processus ;
`SIGTERM` - terminaison du processus ;
`SIGKILL` - mort inconditionnelle du processus ;
`SIGSTOP` - arrêt du processus ;
`SIGCONT` - reprise, après arrêt, du processus.

La syntaxe est :

```
kill -<signal> <PID>
```

Remarque. L'utilisation des touches `Ctrl-C` provoque l'envoi du signal `SIGINT`. D'autres signaux peuvent être envoyés par une combinaison de touches, voir la commande `stty -a`.

▷ **Exercice 2 : Destruction d'un processus**
Dans une fenêtre, lancez un processus (par exemple avec la commande `sleep` *qui attend simplement durant le nombre de secondes qui lui est passé en paramètre).*
Dans une autre fenêtre, repérez ce processus et envoyez-lui des signaux.

3 Processus en arrière-plan et avant-plan

Lorsque vous lancez une commande vous « perdez la main », c'est-à-dire que vous devez attendre la fin de cette exécution pour pouvoir taper une autre commande. Le processus généré se déroule alors en *avant-plan* (ou *foreground*) et « bloque » le lancement de tout nouveau processus. UNIX est un système d'exploitation multitâche et permet donc de lancer plusieurs processus simultanément. Cependant un seul se déroulera en avant-plan, les autres devront se dérouler en *arrière-plan* (ou *background*).
Les processus d'arrière-plan se déroulent en parallèle du processus interactif courant. Ils sont actifs en même temps mais vous avez toujours « la main » pour taper de nouvelles commandes. Cependant tout affichage d'un processus d'arrière-plan se fera sur l'écran et viendra donc se mélanger avec l'affichage d'un processus interactif en avant-plan.
Le lancement d'une commande en arrière-plan se fait par la commande suivie du caractère « & ». Le numéro du processus créé est alors affiché.

```
ermont@leia:~$ sleep 100&

[1] 1499
lermont@leia:~$
  PID TTY          TIME CMD
  871 pts/1    00:00:00 bash
 1499 pts/1    00:00:00 sleep
 1500 pts/1    00:00:00 ps
ermont@leia:~$
```

On peut également faire passer en arrière-plan le processus se déroulant en avant-plan. Pour cela, il faut dans un premier temps obtenir la main sur le shell, en envoyant un signal `SIGSTOP` au processus en avant-plan (par exemple, avec les touches `Ctrl-Z`) puis demander au shell de faire passer le processus en arrière-plan grâce à la commande `bg` (`fg` permet de le faire revenir en avant-plan).
L'exemple suivant montre que l'exécution d'un processus en arrière-plan ne gène pas celle des processus d'avant-plan.

```
ermont@leia:~$ cat boucle
#!/bin/bash
for i in un deux trois quatre; do
   echo $i
   sleep 5
done

ermont@leia:~$ boucle
un

^Z [1] + 19385 Stopped boucle
```

```
ermont@leia:~$ bg

deux

ls -l

-rw-r--r--      1 ermont    teacher  2 fev 10:10 boucle
trois
ermont@leia:~$ kill 19385
   [1] + 19385 Killed boucle
```

▷ **Exercice 3 : Passage en arrière-plan**

Depuis une fenêtre, lancez la commande `xterm`*. Dans quelle fenêtre pouvez-vous taper des commandes ? Dans la fenêtre d'origine, tapez alors* `Ctrl-Z`*. Expliquez le changement ? Comment faire alors pour pouvoir taper des commandes dans les deux fenêtres ?*