ISS - Initiation aux Systèmes d'exploitation et au Shell
LU2IN020

TP 09 – Les signaux

Julien Sopena

décembre 2022

Le but de cette neuvième semaine est d'étudier la communication par signaux. On y étudiera l'envoi de signaux, mais aussi la redéfinition des , de signaux.

Exercice 1 : Highlander

Pour arrêter un processus, vous avez l'habitude de faire un *ctrl+c*. Derrière cette combinaison de touches se cache en fait l'envoi d'un signal `SIGINT` à tous les processus du terminal s'exécutant actuellement au premier plan. Les processus lancés en arrière plan avec un `&` ne reçoivent donc pas le signal.

Question 1

Implémentez un script `highlander.sh` qui affiche chaque seconde `"il ne doit en rester qu'un"` et refuse de mourir lorsque l'on fait un `ctrl-c`, même lorsqu'il est lancé au premier plan.

Question 2

Comment l'arrêter tout de même ? On vous rappelle que la commande `ps` permet d'obtenir les *pid* des processus en cours d'exécution.

Question 3

Peut-on modifier ce script pour qu'il résiste à la réception de tous les signaux, dont la liste est disponible `kill -L`. Vous vous aiderez du `man` de la commande `trap` pour répondre à cette question (il est possible que le `man` de cette commande *buildin* ne soit pas installé sur votre machine personnelle, vous devrez installer un paquet souvent nommé *"manpages-posix"*).

Exercice 2 : Dede

Dans cet exercice, nous allons utiliser la commande `dd` qui permet de copier des données par blocs d'octets, indépendamment de la structure du contenu du disque en fichiers et en répertoires. Cette commande à de multiples applications dont : copie d'un bluray, création d'une clé bootable, création de disques virtuels, recherche de fichiers effacés, analyse de la mémoire vive, ...

Question 1

Pour commencer, lancez la commande suivante et étudiez le résultat.

```
moi@pc /home/moi $ dd if=/dev/urandom of=/tmp/iss bs=1b count=5000000
```

Question 2

Cette commande utilise les signaux pour offrir à la demande un affichage de son état d'avancement. Ouvrez maintenant deux terminaux, dans le premier lancez la commande, puis dans le deuxième envoyez-lui régulièrement un signal `SIGUSR1`.

Pour trouver le *pid* du processus `dd` vous utiliserez la commande `pgrep -n dd`. Vous devriez au passage regarder le man de `pgrep` qui est une commande très puissante.

Question 3

Écrivez maintenant un script qui lance en tâche de fond la commande `dd`, puis une seconde plus tard lui envoie un signal `SIGUSR1`. Pour ne pas polluer le répertoire `/tmp/`, votre script devra attendre la fin de la commande `dd`, puis supprimer le fichier `/tmp/iss` généré.

Question 4

Implémentez pour finir un script qui lance la commande `dd` et produise un affichage de son avancement toutes les secondes jusqu'à ce qu'elle est terminée.

Exercice 3 : Ola

Question 1

Pour commencer, implémentez un script `chain.sh` qui prend en paramètre un entier `n` et va créer `n-1` autres instances de lui-même en cascade : *i.e.*, une instance va en créer une autre qui à son tour va en créer une autre, et ainsi de suite. Vous veillerez à ce que votre script fonctionne, quel que soit l'emplacement d'où il est appelé.

Chacune des instances devra afficher la chaîne `"Ola : O"`. Les espaces entre le caractère `:` et le caractère `O` devront correspondre au nombre de processus restant à créer : `n+1` espaces pour le premier processus, `n` pour son fils (le deuxième dans la chaîne), `n-1`, ..., 1 espace pour le dernier processus.

Question 2

Peut-on, en utilisant la commande wait et sans recréer de nouveau processus, obtenir l'affichage périodique suivant (les lignes identiques étant affichées par le même processus suivant la règle de la première question) ?

```
moi@pc /home/moi $ ./chain.sh 3
Ola :   0
Ola :  0
Ola : 0
Ola :   0
Ola :  0
Ola : 0
Ola :   0
Ola :  0
Ola : 0
....
```

Question 3

L'idée pour obtenir l'affichage périodique voulu est d'utiliser la chaîne de processus créée précédemment en la faisant parcourir par un signal `SIGCONT` circulairement. À savoir que la réception d'un signal `SIGCONT` redémarre (en le mettant à l'état *prêt*) un processus préalablement suspendu (placé à l'état *bloqué*) par la réception d'un signal `SIGSTP`.

Donnez le code d'un script shell `ring.sh` qui reprend le principe du script `chain.sh` pour réaliser le scénario suivant :

- les `n-1` premiers processus devront s'auto suspendre après avoir lancé leur successeur ;
- le dernier processus s'endormira 1 seconde puis lancera un signal `SIGCONT` au premier processus et se stoppera à son tour ;
- à son réveil, le premier processus affichera le message de *Ola* désiré puis enverra un signal `SIGCONT` au deuxième processus avant de se re suspendre ;
- et ainsi de suite pour tous les processus : affichage, réveil du suivant, auto suspension. Le successeur du dernier processus créé étant le premier processus.

Afin de simplifier le développement, l'anneau ne sera parcouru que 10 fois. Notez que l'implémentation de cette dernière consigne repose entièrement sur un compteur local à chaque processus.