Shell(s) et langage(s) de script : Script

Formation continue — Université Lille 1
Pour toutes remarques : Alexandre.Sedoglavic@univ-lille1.fr

Licence trimestre 3 — 2008-09

Un fichier correspondant à un script doit être exécutable par un interpréteur ; on doit donc systématiquement lui assigner les droits d'exécution :

```
% chmod u+x monscriptamoi
```

Un script shell commence par la ligne

```
#!/chemin/acces/moninterpreteur
```

qui indique au système que le fichier en cours d'exécution est composé de commandes devant être interprétées par l'intrepréteur `moninterpreteur` dont l'exécutable se trouve dans le répertoire `/chemin/acces`.

Les paramètres de position dans un script correspondent aux arguments de la ligne de commande invocant ce dernier.

On peut bien sur utiliser tous les paramètres spéciaux et définir des variables.

Enfin, toutes les commandes, mot-clefs et expressions sont à disposition.

Un retour chariot est un terminateur de commande comme l'indique l'exemple suivant :
`if [ $# -ne 2 ] ; then echo "pb" ; fi` à comparer avec la conditionnelle du script qui suit.

```
# Ceci est un commentaire
#!/bin/bash
NBPARAM=2 # nombre de param\‘etre du script

usage()
# Comment utiliser ce script
{
    echo "Usage: ‘basename $0‘ firstparam secondparam"
    echo "Print \"’firstparam’ and ’secondparam’\"".
    return # juste pour dire que ca existe
}

# Test le nombre de param\‘etre et ex\’ecute
# la fonction usage() s’il est different de~$2$
if [ $# -ne $NBPARAM ]
then
    usage # l’appel de fonction se fait sans ()
    exit false
fi

# le script fait son job
echo $1 and $2
# et se termine bien
exit true
```

La commande true retourne le code 0. Les 2 points : en sont un synonyme (destiné sans doute à rendre du code illisible :-)

Le commande interne

- continue *n* reporte l'exécution au début de la première commande composée itérative la contenant ou de la *n*ième commande composée la contant si *n* est spécifié. Cette commande composée est alors réexécutée.
- break *n* provoque la fin de la première commande composée itérative la contenant ou de la *n*ième commande composée la constante si *n* est spécifié. L'exécution du code reprend à la suite de cette commande composée.

```
% for i in ‘ls /‘ ;  do \
> if [ -d "/$i" ] ; then continue ; fi ; \
> echo "$i n’est pas un repertoire" ; done
% cd / ; for i in ‘ls .‘ ; do if [ -d "/$i"] ;
>  then cd $i ; else break ; fi ; done
%
```

Un script peut être interprété de la façon suivante :

```
% bash -[opt] nomduscript
```

afin de pouvoir utiliser les options décrites dans

```
% bash -c "help set"
```

On peut notamment suivre l'évaluation de toutes les commandes par le biais de l'option `-x`.

Pour parfaire sa connaissance du shell, on peut s'adonner à l'exercice plaisant de lire des scripts.

Par exemple dans un système d'exploitation de type UNIX, chaque service (impression, réseaux, etc.) est controlé par un script présent dans le répertoire `/etc/init.d`. Ils sont soit lancés au besoin par l'utilisateur, soit lors du boot par le processus init sous la direction du fichier `/etc/inittab` et des liens dans les répertoires `/etc/rc[1-6].d`.

Il est instructif d'y jeter un coup d'oeil. Par exemple, on peut consulter le script killall.

```
#! /bin/bash
# Bring down all unneeded services that are still running
# (there shouldn't be any, so this is just a sanity check)
case "$1" in
   *start) ;;
   *) echo $"Usage: $0 {start}"
        exit 1 ;;
esac
for i in /var/lock/subsys/* ; do
        # Check if the script is there.
        [ -f "$i" ] || continue
        # Get the subsystem name.
        subsys=${i#/var/lock/subsys/}
        # Networking could be needed for NFS root.
        [ $subsys = network ] && continue
        # Bring the subsystem down.
        if [ -f /etc/init.d/$subsys.init ]; then
                /etc/init.d/$subsys.init stop
        elif [ -f /etc/init.d/$subsys ]; then
                /etc/init.d/$subsys stop
        else
                rm -f "$i"
        fi
done
```

Définition de fonction dans le shell

Afin d'utiliser ses propres fonctions dans le shell, on peut les définir dans un script

```
% cat > /tmp/foo << fin
> mafct()
> {
> echo "La vie est belle"
> }
> fin
%
```

Ceci fait, on peut utiliser la commande interne source bar qui lit et exécute le fichier bar dans l'intrepréteur courant :

```
% source /tmp/foo
% mafct
La vie est belle
```