TP Script sous Linux

Romain Raveaux

1. Les commandes de base

Testez les commandes suivantes et décrivez leurs fonctions :
id
whoami
pwd
cat
cp
mv
mkdir
ls
chmod
rm
head
tail
cut
uname -a

2. Les variables d'environnement

Comment initialiser une variable ?
Comment visualiser le contenu d'une variable ?
Comment supprimer une variable ?
Quelle est la valeur des variables standards HOME et PATH ?
Lister les variables connues par votre shell avec les commandes env et set ?
Expliquer la différence entre les variables exportées et non exportées ?

3. Meta-caractères de séparation et de terminaison de commandes

Expliquer ce que font les commandes suivantes :

1°) Quel est le résultat des commandes suivantes ?

touch fich ; **chmod** 123 fich.

2°) Quel est l'effet des commandes suivantes ?

mavaleur= « bonjour »
echo $mavaleur
echo $mavaleur>monfichier

3°) Pouvez vous écrire mavaleur dans fich ? Dans la négative, trouver une solution.

4°) Que fait la commande suivante : (ctrl-d pour finir la commande)

cat>montexte

5°) Que font les commandes suivantes :

ls –l /bin | more
ls -l | sort
ls -l | wc –l

6°) Quel est l'effet des commandes suivantes :

man id
man id > essai.txt

4. Méta-caractères de redirection

Tout processus UNIX prend par défaut ses informations sur l'entrée standard (clavier) et restitue par défaut ses résultats sur la sortie standard (écran du terminal). On peut modifier ce comportement par défaut en utilisant des métacaractères de redirection :

- Expliquer ce que font les commandes suivantes (Lorsqu'une commande attend une entrée clavier, tapez quelques lignes de texte puis terminez par Ctrl-d (symbole EOF=end-of-file) pour sauvegarder).

```
$ cat > essai.txt
$ cat  essai.txt
$ sort  < essai.txt
$ cat  >> essai.txt
$ sort  < essai.txt
$ sort  < essai.txt  > essai-tri.txt
$ cat essai-tri.txt
$ cat  essai.txt  essai-tri.txt
```

- Quel est l'effet de la commande suivante ? (essai.txt est le fichier créé précédemment)

```
$ wc -w  < essai.txt > mots.txt
```

Que se passe t-il si on enlève l'option -w ?

Quelle est la différence entre les deux commandes suivantes :

```
$ cp essai1 essai2
$ cat < essai1 > essai2
```

5. Caractère d'encadrement

Quel est la différence entre :
echo pwd
et
echo `pwd`

6. Programmation Shell

Un script est un fichier texte interprété par le Shell (l'interpréteur de commandes).
Il existe plusieurs interpréteurs de commandes et bash est l'un des plus connu.
Le programme bash est situé dans le répertoire /bin.

Pour qu'un fichier texte puisse être correctement compris par le système d'exploitation, il faut préciser quel interpréteur de commandes sera utilisé pour exécuter le fichier.

La première ligne d'un script définit l'interpréteur à utiliser :
#!/bin/bash

1°) Réaliser le script suivant :

```
$ cat fichier_script
x=12
y=14
if [ $x -ge $y ]
then echo superieur ou egal
else echo inferieur
fi
echo essai termine
$ chmod 700 fichier_script          -> rendre le fichier exécutable
```

Pour exécuter le script, il suffit de taper la commande : ./fichier_script

2°) Réaliser les scripts présentés en exemple :
Quelques éléments de syntaxe :

Instruction conditionnelle if then else :

```
format :if      liste_de_commandes1
                then    liste_de_commandes2
                else    liste_de_commandes3
                fi
```

Instruction for :

```
format :        for     paramètre       in      liste
                do      liste_de_commandes
                done
```

Un exemple de la boucle for :
Exemple 1 :

```
#!/bin/bash
for fichier in `ls`
do
echo "Fichier trouvé : $fichier"
done
```

Instruction while et Instruction until :

format :
```
while   liste_de_commandes1
do      liste_de_commandes2
done
```

2 exemples de la boucle while :
Exemple 1 :

```
#!/bin/bash
while [ -z $reponse ] || [ $reponse != 'oui' ]
do
echo –n 'Dites oui : '
read reponse
done
```

Exemple 2 :

```
a=0
while [ $a -ne 10 ]
do      a=$((a+1))
        echo $a
done
```

Définition d'une fonction :

deux formats sont autorisés pour définir une fonction :

```
function   nom_de_la_fonction  {script_de_la_fonction}
```

OU

```
nom_de_la_fonction ( ) {script_de_la_fonction}
```

Passage de paramètres :
Il est possible de passer des paramètres à une fonction ou à un script. Chaque paramètre est séparé par un espace.
Exemple : ./monscript.sh salut bonjour

Pour récupérer ces variables dans un script il suffit d'utiliser les méta-caractères suivants :
$1 : le premier paramètre (salut)
$2 : le 2ème paramètre (bonjour)
$3 : …
Le nombre de paramètres est contenu dans la variable $#

7. Expressions régulières

Avec un éditeur de texte, créer un fichier « essai.txt » qui contient les caractères suivants :
aaaa
bbbb
ccccx
abcdx
aefghxijklx
aijklx
4

1°) Essayer les commandes suivantes :
grep aaa essai.txt
grep [0-5] essai.txt
grep [a-d] essai.txt
grep a.*x essai.txt
grep –r apache /etc
2°) Conclure sur chacune des commandes.

8. Manipulations de base dans un shell concernant les processus :

● Visualisez les processus tournant sur votre machine avec la commande ps –ef et la commande top. Regarder les colonnes UID, PID et PPID.
● Visualisez l'affiliation des processus de votre machine avec la commande pstree.
● Afin de mieux voir les résultats, faites une redirection de la sortie de pstree dans un fichier que vous appellerez sortie.txt. Ouvrez ensuite le fichier avec un éditeur de texte. Faite la même chose avec la commande ps -ef dans un fichier qui s'appellera sortie2.txt.
● Utilisez ensuite la commande grep avec sortie2.txt comme canal standard d'entrée et affichez seulement la ligne contenant firefox.