TP1&2 Script M1105

Table des matières

1) Affichage

1.1) commande echo pour afficher en une seule commande :

```
echo -e "bonjour \ntout le monde"
```

1.2) commande echo pour afficher en trois commandes :

```
echo "bonjour"
echo -n "tout"
echo " le monde"
```

1.3) commande echo pour afficher en deux commandes :

```
echo -n "bonjour"
echo " tout le monde"
```

2) Variables

2.1) Les variables contenant le nom de l'utilisateur et le nom de la machine sont USERNAME et HOSTNAME.

```
HOSTNAME=205-2
USER=samuel.laforge01
USERNAME=samuel.laforge01
```

2.2) Avec geany :

```
user="$USERNAME"
machine="$HOSTNAME"
echo "Vous etes l'utilisateur $user travaillant sur la machine $machine"
```

Affichage dans le terminal :

```
samuel.laforge01@205-2 ~ $ bash Q2.2
Vous etes l'utilisateur samuel.laforge01 travaillant sur la machine 205-2
```

TP1&2 Script M1105

2.3) Avec geany :

```
nom='laforge'
prenom='samuel'
echo "l'utilisateur $USERNAME s'appel $nom $prenom"
```

Affichage dans le terminal :

```
samuel.laforge01@205-2 ~ $ bash Q2.3
l'utilisateur samuel.laforge01 s'appel laforge samuel
```

2.4) Avec geany :

```
cmd_ls=$(ls)
echo -e "$cmd_ls \n\n$cmd_ls"
```

Affichage dans le terminal :

```
Desktop
Q1.1
Q1.2
Q1.3
Q2.2
Q2.3
Q2.4
tp

1.2
Desktop
Q1.1
Q1.2
Q1.3
Q2.2
Q2.3
Q2.4
```

TP1&2 Script M1105

```
tp
```

2.5) Avec geany :

echo -n 'nom : '

read nom

echo -n 'prenom : '

read prenom

echo -e "\n$prenom $nom"

Affichage dans le terminal :

```
nom : Laforge
prenom : Samuel
Samuel Laforge
```

3) Analyse de la ligne de commande

3.1) Pour afficher l'ensemble de ligne de commande d'un script qu'on écrit il faut utiliser $*. Ensuite en utilisant la commande : bash Nom_du_fichier ecrire ce que l'on veut et entrée on aura tout ce que l'on aura entré affiché dans le script.

3.2) Pour afficher la commande utilisée juste avant il faut faire echo $0 :

```
samuel.laforge01@205-2 ~ $ echo $0
bash
```

3.3)

Avec geany :

creerdossier=$(mkdir script)

droit=$(chmod 0777 script)

verification=$(ls -all script)

echo $creerdossier $droit $verification

Affichage dans le terminal :

```
samuel.laforge01@206-9 ~ $ bash Q3.3
total 8 drwxrwxrwx 2 samuel.laforge01 eleves 4096 oct. 11 11:45 . drwx--x--x 11
samuel.laforge01 eleves 4096 oct. 11 11:45 ..
```

TP1&2 Script M1105

3.4) La variable permettant d'afficher le statut du script est **$?**:

Ainsi, la console renvoi un code d'erreur 1 car le répertoire a déjà été créé à la question précédente.

3.5) En gardant le script d'avant je crée un répertoire dans mon dossier personnel dans lequel j'ai les droits :

```
samuel.laforge01@205-12 ~ $ bash script script2
0
0
```

3.6) Dans /root, on ne possède pas de droits donc lorsque l'on exécute le script à l'intérieur la console nous retourne ;

```
samuel.laforge01@205-12 ~ $ bash script /root/test
mkdir: impossible de créer le répertoire «/root/test»: Permission non accordée
1
chmod: impossible d'accéder à '/root/test': Permission non accordée
1
```

3.7) Pour créer un répertoire dans un dossier en utilisant le même script il faut utiliser latouche **‘** qui permet de garder un seul argument pour créer deux dossiers en même temps :

```
samuel.laforge01@205-12 ~ $ bash script133 'script6 script6/script7'
0
0
```

3.8) Lors des trois question précédentes on a obtenu un code « 0 » lorsque tout fonctionnait et un code « 1 » lorsqu'il y avait une erreur. Ainsi on peut en déduire que la variable **$?** permet d'afficher le code d'erreur de la commande.

4) Utilisation du shebang

4.1) Le shebang (#!) est la première ligne d'un fichier texte qui indique au système que ce fichier est un script. On l'utilise dans la première ligne du fichier suivi de l'interpréteur permettant d'exécuter le script.

4.2) Pour exécuter un script de cette manière il faut au minimum les droits : rwxr-xr-x

4.3) Pour rendre le premier script exécutable, il suffit d'utiliser la commande **chmod +x script** et rajouter en premier **#!/bin/bash**.

```
#!/bin/bash
echo -e bonjour «\n» tout le monde
```

TP1&2 Script M1105

4.4) Pour créer un dossier bin dans le home directory, il faut d'abord se situer dans l'emplacement home. Ensuite on crée le dossier bin, le script et on le rend exécutable.

```
cd
mkdir bin
cd bin
touch script
chmod +x script
geany script
```

Ensuite on entre les commandes suivantes :

```
#!/bin/bash
echo $PATH
```

le terminal nous retourne

```
usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

réponse différente car manipulation faite sous Linux Kali

4.5) Cette technique est utilisée afin d'invoquer un script avec n'importe quel interpréteur.

SCRIPT TP2

1 Test

IF

1) If dans un script permet de créer un algorithme : il correspond au «si» ceci, faire cela. Il s'utilise dans le choix si, alors, sinon.

2) Ce script permet de vérifier si une note a bien était saisie :

```
#!/bin/bash
echo 'Merci de saisir une note'
read note
if [ $note ];
then
        echo $note
else
        echo 'Merci de saisir une note'
fi
```

TP1&2 Script M1105

3) Ce script permet de vérifier qu'une note a bien était saisie et si la note est comprise entre 0 et 20

```
#!/bin/bash
echo 'Merci de saisir une note entre 0 et 20'
read note
if [ $note -ge 0 ] && [ $note -le 20 ];
then
        echo «Note : $note»
fi
if [ $note -gt 20 ];
then
        echo «Note non comprise entre 0 et 20»
        echo «Merci de saisir une note ≤ 20»
fi
if [ $note -lt 0 ];
then
        echo «Note non comprise entre 0 et 20»
        echo «Merci de saisir une note ≥ 0»
fi
```

Case

1) La commande **case** permet d'effectuer des tests. Elle oriente la suite du programme en fonction d'un choix de différentes valeurs. Lorsque le nombre de choix est important la commande **case** est plus appropriée que la commande **if**. Elle permet de remplacer plusieurs **if**

2) Ce script permet de vérifier si nous sommes bien un Dalton

```
#!/bin/bash
read dalton
case $dalton in
«William») echo «Bonjour Monsieur Dalton»;;
«Averell») echo «Bonjour Monsieur Dalton»;;
«Jack») echo «Bonjour Monsieur Dalton»;;
«Joe») echo «Bonjour Monsieur Dalton»;;
```

```
*) echo «Vous n’etes pas un Dalton»;;
esac
```

3) Même programme mais qui a été raccourci

```
#!/bin/bash
read dalton
case $dalton in
«William»  | «Averell» | «Jack» | «Joe») echo «Bonjour Monsieur Daloton»;;
*) echo «Vous n’etes pas un Dalton»;;
esac
```

2) Boucle

FOR

1) La commande **for** permet de parcourir une liste de valeur dans une boucle.

2) Ce script permet de lister le contenu du répertoire personnel (question faite sur machine virtuelle à la maison)

```
#!/bin/bash
for i in /home/test ;
do
        ls -l
done
```

3)

4) Ce script permet d’afficher la suite des puissances de 4 jusqu’à 4^10

```
#!/bin/bash
for i in `seq 0 10`;
do
        echo $((4**$i))
done
```

While

1) La boucle **while** permet d'exécuter les commandes présentes entre le **do** et le **done** tant que la commande1 placée à droite du **while** retourne un code **vrai**.

TP1&2 Script M1105

2) Ce script permet d'afficher les 10 premières puissances de 2

```
#!/bin/bash
x=1
y=0
while ((y<10));
do
        echo 2^$y=$x
        ((y+=1))
        ((x=2**y))
done
```

3) Exercice

1) Ce script permet de demander à l'utilisateur de saisir une note entre 0 et 100

```
#!/bin/bash
echo 'Merci de saisir une note entre 0 et 100'
read note
if [ $note -ge 0 ] && [ $note -le 100 ];
then
        echo «Note : $note»
fi
if [ $note -gt 100 ];
then
        echo «Note non comprise entre 0 et 100»
        echo «Merci de saisir une note ≤ 100»
fi
if [ $note -lt 0 ];
then
        echo «Note non comprise entre 0 et 100»
        echo «Merci de saisir une note ≥ 0»
fi
```

TP1&2 Script M1105

2) Ce script permet de générer un nombre aléatoirement compris entre 0 et 100

```
#!/bin/bash
ECHELLE=100
nombre=$RANDOM
let «nombre %= $ECHELLE»
        echo «$nombre»
```

3) Ce script permet de dire si la note saisie est inférieure, supérieure ou égale au nombre aléatoire

```
#!/bin/bash
ECHELLE=100
nombre=$RANDOM
let «nombre %= $ECHELLE»
        echo «Merci de saisir une note entre 0 et 100»
read note
if [ $note -lt $nombre ];
then
        echo «note inferieure au nombre aleatoire»
fi
if [ $note -gt $nombre ];
then
        echo «note superieure au nombre aleatoire»
fi
if [ $note -eq $nombre ];
then
        echo «note egale au nombre aleatoire»
fi
```

TP1&2 Script M1105

4) Ce script demande à l'utilisateur de saisir un nombre jusqu'à ce qu'il trouve la bonne réponse

```
#!/bin/bash
ECHELLE=100
nombre=$RANDOM
let «nombre %= $ECHELLE»
        echo «Merci de saisir une note entre 0 et 100»
read note
while [ $note -lt $nombre ] || [ $note -gt $nombre ];
do
read note
if [ $note -lt $nombre ];
then
        echo «note inferieure au nombre aleatoire»
fi
if [ $note -gt $nombre ];
then
        echo «note superieure au nombre aleatoire»
fi
done
        echo «Felicitation»
```