Table des matières

1) Affichage

1) Je fais la commande suivante pour retourner Bonjour «retour à la ligne» tout le monde en une commande :

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

2) Pareil mais en 3 commandes :

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

3) Pareil mais en 2 commandes

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

2) Variables

1) J'utilise la commande set pour récupérer les variables d'environnement correspondant au nom de l'utilisateur et de la machine :

```
test@202-19:~$ set |grep HOST
HOSTNAME=202-19
```

```
test@202-19:~$ set |grep USER
USER=test
```

2) Je fais un script pour pouvoir afficher "vous êtes l'utilisateur x utilisant la machine y"

```
user=$USER
machine=$HOSTNAME

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

3) J'utilise le script pour afficher "l'utilisateur x s'appelle $nom $prenom" avec des variables

```
nom='LAFORGE'
prenom='Samuel'

echo "L'utilisateur $USER s'appelle $nom $prenom"
```

4) J'assigne la commande ls à une variable et je l'affiche deux fois

```
ls=$(ls)

echo -e "$ls \n$ls"
```

5) Ce script permet de demander à l'utilisateur son nom et son prénom puis les affichent dans le terminal

```
echo "entrez votre nom"
read nom
echo "entrez votre prenom"
read prenom

echo "Vous etes $nom $prenom"
```

3) Analyse de la ligne de commande

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.

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

3) Au lancement du script il faut juste faire "bash [nom-du-script] [argument]" ce qu sera donné en argument sera le nom du dossier crée

```
mkdir $1
chmod o+rwx $1
```

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

```
mkdir $1
chmod o+rwx $1
echo "$?"
```

5) Je vais dans mon répertoire /home et j'exécute mon script :

```
test@202-19:~$ bash nom-prenom coucou
0
```

6) En allant dans mon répertoire /root je ne peux pas exécuter le script car je n'ai pas les droits

```
test@202-19:/home$ ./nom-prenom dossier
mkdir: impossible de créer le répertoire « /root/dossier »:
Permission non accordée
chmod: impossible d'accéder à '/root/dossier': Permission non
accordée
1
```

7) Le but ici est de créer un dossier (2ème argument) dans un autre dossier (1er argument) et de leur donner les droits.

```
mkdir -p $1/$2
chmod -R o+rwx $1
echo "$?"
```

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

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.

2) Pour pouvoir exécuter le script nous devons disposer au minimum du droit d'exécution

3) Pour rendre le premier script exécutable on fait la commande suivante :

```
chmod +x [fichier]
```

4) Dans /home/test je crée un dossier /bin et dans ce dossier j'y place mon script auto exécutable

A partir de maintenant je dois faire un export PATH='…...' et ajouter le chemin jusqu'à mon script pour pouvoir l'exécuter de n'importe où :

```
test@202-19:~/bin$ export PATH='/usr/local/bin:/usr/bin:/bin:/usr/
local/games:/usr/games:/snap/bin:/var/lib/snapd/snap/bin:/home/
test/bin'
```

5) J'ai seulement besoin d'écrire dans le terminal le nom de mon script pour l'exécuter de n'importe où (comme si c'était une commande Linux). Je n'ai pas besoin de donner le chemin vers mon script.