

La programmation Shell : passage de paramètres.

Nous avons vu en TP comment rédiger un premier programme shell. Voici pour rappel un exemple de programme qui affiche le nom de l'utilisateur connecté:

La première ligne indique le shell ou interpréteur qui sera lancé pour exécuter le programme.

#!/bin/bash

#On affiche maintenant le nom de l'utilisateur actuellement connecté à l'aide de la variable d'environnement \$USER. On passe une ligne après la première ligne du script

echo \$USER

Vous noterez que tout programme shell porte l'extension "sh". Nous allons voir aujourd'hui comment passer des paramètres aux scripts que l'on exécute.

Considérons le programme "**mystere.sh**" suivant:

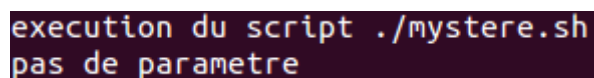
```
#!/bin/bash

echo  execution du script $0

if [ $# -ne 1 ]
then
    echo  pas de parametre
else
    if [ -f $1 ]
    then
        echo  $1 est un fichier
    else
        echo  $1 n est pas un fichier
    fi
fi
```

Exercice:

1. Recopiez et exécutez ce script
2. Que permet de faire ce script ?



```
execution du script ./mystere.sh
pas de parametre
```

3. Commentez chaque ligne du programme (caractère # pour commenter) afin d'expliquer son fonctionnement détaillé.
4. Complétez le tableau suivant:

| Opérateur / Symbole | [Exemple] | Signification |
|------------------------------|---------------------------------|--|
| \$# | if [\$# -ne 1] | Le nombre de paramètre |
| \$0 | echo execution du script \$0 | Nom du script |
| \$1 \$2 \$3 ... | if [-f \$1] | Le premier paramètre, le deuxième paramètre, le troisième paramètre |
| \$* | for param in \$* | Liste de tous les paramètres |
| Chaînes de caractères | | |
| = | \$var1 = \$var2 | renvoie vrai si les variables \$var1 et \$var2 ont le même contenu |
| != | \$var1 != \$var2 | renvoie vrai si les variables \$var1 et \$var2 sont différentes |
| -n | -n "\$var" | renvoie vrai si la variable \$var contient quelque chose |
| -z | -z "\$var" | renvoie vrai si la variable \$var est vide |
| Numériques | | |
| -eq | \$var1 -eq \$var2 | renvoie vrai si les variables \$var1 et \$var2 sont égales |
| -ne | \$var1 -ne \$var2 | renvoie vrai si les variables \$var1 et \$var2 sont différentes |
| -gt | \$var1 -gt \$var2 | renvoie vrai si la variable \$var1 est plus grande que \$var2 |
| -ge | \$var1 -ge \$var2 | renvoie vrai si la variable \$var1 est plus grande ou égale à \$var2 |
| -lt | \$var1 -lt \$var2 | renvoie vrai si la variable \$var1 est plus petite que \$var2 |
| -le | \$var1 -le \$var2 | renvoie vrai si la variable \$var1 est plus petite ou égale à \$var2 |
| Fichiers | | |
| -f | -f \$var | |
| -d | -d \$var | |
| -r [-w[-x]] | -r \$var | renvoie vrai si \$var est un fichier en mode lecture (respectivement en écriture et en exécution) pour l'utilisateur |

Les boucles

Pour comprendre le fonctionnement des boucles, recopiez les scripts suivants et testez les :

```
Applications Raccourcis
eleve@UBUNTU: ~
Fichier Édition Affichage Rechercher Terminal Aide
eleve@UBUNTU:~$ cat users.txt
bmurray 0601010101
eleve 0702020202
toto 0603030303
eleve@UBUNTU:~$ cat script1.sh
#!/bin/bash

cat users.txt | while read USER TEL
do

    echo "Le numéro de mobile de $USER est $TEL."

done
eleve@UBUNTU:~$
```

```
Applications Raccourcis
eleve@UBUNTU: ~
Fichier Édition Affichage Rechercher T
eleve@UBUNTU:~$ cat script2.sh
#!/bin/bash

CHEMIN="/etc/network/if-down.d"

for l in `ls $CHEMIN`
do

    ls -l $CHEMIN/$l

done
eleve@UBUNTU:~$
```

```
Applications Raccourcis
eleve@UBUNTU: ~
Fichier Édition Affichage Rechercher
eleve@UBUNTU:~$ cat script3.sh
#!/bin/bash

for i in `seq 1 5`
do

    if [ $i -lt 10 ]
    then

        echo "0$i"

    else

        echo $i

    fi

done
eleve@UBUNTU:~$
```

Exercices

1. Rédigez un script qui prend en paramètre le nom d'un fichier contenant une liste de noms de répertoires et qui créera dans le répertoire ~/DATA tous ces répertoires.

```
GNU nano 2.2.6      Fichier : Creati
#!/bin/bash

for rep in `cat repertoires.txt`
do
    mkdir -p /$rep
done
```

2. Rédigez un script qui prend en paramètre un nom d'utilisateur, et qui :
 - indiquera si cet utilisateur existe ;
 - indiquera le shell utilisé par cet utilisateur (/bin/sh, /bin/bash, /bin/ksh, ...) ;
3. Visionnez les vidéos fournies en prenant des notes et en me posant des questions pour les points qui posent problème.
4. Complétez le script sur l'inventaire du poste de travail.
5. Indiquez dans le fichier de log le moment d'exécution du script.
6. On organise tous les jours une sauvegarde des données sur un poste de travail. Ces données sont sauvegardées dans le répertoire /home/eleve/BACKUP tous les jours de l'année. Proposez une solution pour créer automatiquement l'arborescence complète suivante :
 - janvier
 - 01
 - 02
 - 03
 - ...
 - 31
 - février
 - 01
 - 02
 - ...
 - 28
 - mars
 - 01
 - 02
 - ...
 - 31
 - ...