Grégoire Roumache

Laboratoires de Bash

01-variables-environnement.sh

```
# Réaliser un script qui affiche le contenu de 2 variables.
# La variable utilisateur private est interne au script et
# la variable pub est une variable globale définie dans le
# shell avant d'exécuter le script. Vérifier selon que l'on
# définisse pub comme variable globale ou non, elle est
# accessible par le script ou pas.

# this is my private variable
PRIVATE="Hello ^w^"

# print both variables
echo "public variable = $PUB"
echo "private variable = $PRIVATE"

# NOTE: the public variable has to be exported to be used in the script
#       e.g. $PATH can be used with no problem

# ex - create a public variable:
# 1. PUB="bonjour ^o^"
# 2. export PUB
# --> or : export PUB="bonjour ^o^"
```

02-parametres-1.sh

```
# Créer un script qui compte le nombre de paramètres
# qui lui ont été passé

# affiche un message personnalisé selon votre nom
# d'utilisateur entré en paramètre au script:

# robert = bonjour robert
```

```
# test = attention ceci est un compte de test
# root = Bienvenue administrateur
# nom inconnu = Erreur (+ retourner le code d'erreur)

# Variante : pour chaque paramètre entré, le script fait le test

exit=0

if [$1 == "robert"]
    then echo "bonjour robert"
elif [$1 == "test"]
    then echo "attention ceci est un compte de test"
elif [$1 == "root"]
    then echo "Bienvenue administrateur"
else
    echo "Erreur !! >_<"
    exit=1
fi

echo $exit

# faire: echo $?, pour avoir le code de sortie du script !!
```

02-parametres-2.sh

```
# idem que: '02-parametres-1.sh'

exit=0

echo "Nombre de paramètres = $#"

case $1 in
    "robert") echo "bonjour robert"
    "test") echo "attention ceci est un compte de test"
    "root") echo "Bienvenue administrateur"
    *)
        echo "Erreur >_<"
        exit=1
esac

echo $exit
```

03-options-et-code-de-retour.sh

```
# Réalisez un script qui accepte les option a b c. En cas
# d'option inconnue, il doit retourner le code d'erreur 1

while getopts ":abc" option
do
```

```
    case $option in
        a) echo "commande a";;
        b) echo "commande b";;
        c) echo "commande c";;
        \?)
            echo $OPTARG "option invalide"
            exit 1;;
    esac
done

echo "Option analysis over"
exit 0

# you have to use run the script like this: ./<script> -<option> -<option>
# e.g. ./03-script.sh -a -hello
```

04-lister-les-logs-systemes.sh

```
# Réaliser un script qui affiche le contenu de tous les fichiers
# log qui se trouvent dans le répertoire /var/log avec l'option
# -a (y compris dans les sous répertoires).
# Sans options, il affichera le listing des différents fichiers
# de log. Avec -l, il listera le contenu du fichier de log demandé.

if [[ $# -eq 1 ]]
then
    echo "listing des fichiers ^^"
    ls -l /var/log
    exit 0
fi

while getopts ":al" option
do
    case $option in
        a)
            echo "listing des fichiers récursif ^^"
            ls -R -l /var/log;; # -R = recursive
        l)
            echo "affichage du log demandé ^^"
            cat /var/log/$2;;
        \?)
            echo "erreur: option inconnue ^^"
            exit 1;;
    esac
done

exit 0
```

05-determiner-le-type-du-fichier.sh

```
# Réaliser un script qui affiche "le fichier existe" dans
# le cas ou il existe et si oui le type de fichier
# (fichier, dossier, exécutable, etc.)

FILE=$1

if test $FILE;
then
    echo "le fichier existe ^^"

    file $FILE

    exit 0
fi

exit 1
```

06-lister-les-comptes-utilisateurs.sh

```
# Réaliser un script qui liste les comptes utilisateurs normaux
# du système, ç à d dont le UID est supérieur à 500 (3ème
# colonne dans le fichier /etc/passwd). Le script doit également
# indiquer le shell par défaut employé par cet utilisateur.

awk -F: '{if ($3 >= 500) { print $1, "-", $3, "-", $7 }}' /etc/passwd
```

07-creer-des-comptes-utilisateurs.sh

```
# Réaliser un script qui permet de créer des comptes
# utilisateurs à partir d'un fichier comprenant : login
# et mot de passe, séparé par des virgules.

# L'option –h permet d'afficher une aide indiquant l'ordre
# dans lequel les éléments doivent être présent dans le
# fichier et l'option -o. Le séparateur par défaut est ";".
# L'option –s permet de choisir un autre séparateur.

function create_logins() {
    file=$1
    separator=$2

    echo "file ='$file', separator = '$separator'"

    while read -r line
    do
        login=$(echo $line | cut -d $separator -f 1)
        password=$(echo $line | cut -d $separator -f 2)
        command="useradd $login --password $password"
```

```
        echo $command
        $command
    done < "$file"
}

separator=';'

while getopts ":hos" option
do
    case $option in
        h)
            echo "format fichier = <login>;<mdp>"
            echo "use -s to change the separator"
            echo "use -o to specify the file";;
        o)
            file=$2
            create_logins $file $separator;;
        s)
            separator=$2;;
        \?)
            echo "erreur: option incorrecte";;
    esac
done
```