TD 3 : Script Shell sous Unix

Exercice 1

1. Ecrire un script **script.sh** qui demande à l'utilisateur un nom de répertoire. Si un répertoire de ce nom n'existe pas déjà, il crée le répertoire avec ce nom. Dans tous les cas, il se déplace dans ce répertoire et enfin affiche un message "On est dans le répertoire" suivi du chemin absolu du répertoire.

   Réponse :

```
#!/bin/bash
read -p "Nom du répertoire : " rep
if [ ! -d $rep ]
then
    mkdir $rep
fi
cd $rep
echo "On est dans le répertoire `pwd`"
```

2. Ecrire une commande **protege** qui :

   - demande à l'utilisateur d'entrer un nom de fichier ;
   - vérifie que ce fichier existe ;
   - protège complètement ce paramètre (enlève tous les droits à tous les autres utilisateurs — y compris ceux du même groupe que le propriétaire).

   Réponse :

```
#!/bin/bash
read -p "Nom du fichier : " fich
if [ -e $fich ]
then
    chmod go-rwx $fich
fi
```

3. Ecrire une commande **effacer** qui :

   - demande à l'utilisateur d'entrer un nom de fichier ;
   - teste que ce fichier est un fichier ordinaire ;
   - vérifie que le répertoire **poubelle** existe à la racine de votre compte ;
   - le crée sinon ;
   - déplace le fichier de nom entré en début de programme dans le répertoire **poubelle**.

   Réponse :

```
#!/bin/bash
read -p "Nom du fichier : " fich
if [ -f $fich ]
then
    if [ ! -d ~/poubelle ]
```

```
    then
        mkdir ~/poubelle
    fi
    mv $fich ~/poubelle
fi
```

Exercice 2

Créer un programme capable de compter le nombre de fichiers dans un répertoire. La syntaxe du programme est : `nbfich [repertoire]`
où repertoire est le chemin du répertoire désiré. Si repertoire n'est pas spécifié, utiliser le répertoire courant.

Réponse :

```
#!/bin/bash
if [ $# -eq 0 ]
then
    rep=.
else
    rep=$1
fi
echo "Le répertoire $rep contient `ls $rep | wc -l` fichiers."
```

Exercice 3

Ecrire un script pour connaitre le plus grand nombre de trois nombres donnés en paramètres de la commande. Afficher un message d'erreur si le nombre d'arguments n'est pas suffisant.

Réponse :

```
#!/bin/bash
if [ $# -eq 3 ]
then
    max=$1
    if [ $max -lt $2 ]
    then
        max=$2
    fi
    if [ $max -lt $3 ]
    then
        max=$3
    fi
    echo "max($1,$2,$3) = $max"
fi
```

Exercice 4 : Utilisation de la commande `expr`

Dans le langage Bourne-Shell ou bash, il n'y a pas, à proprement parler, de variables numériques. Les manipulations arithmétiques sont réalisées au travers de la commande `expr` qui interprète certains de ses paramètres de position comme les représentations ASCII de nombres entiers, les arguments incorrects provoquant une erreur de la commande `expr` matérialisée par un code de retour égale à 2

1. Par quel moyen simple est-il possible de tester qu'une variable Shell est une valeur interprétable numériquement?

Réponse : L'idée est d'inclure la variable dans une opération arithmétique avec un nombre quelconque (dans cet exemple le nombre 0) et faire un test sur le code de retour de la commande `expr`.

```
#!/bin/bash
read -p "Donner une valeur : " x
if expr $x + 0 >/dev/null 2>&1 # vrai ssi $x est un nombre
                               # ==> code de retour de expr = 0
then
    echo "$x est numérique"
else # ici, le code de retour de expr est 2
    echo "$x n'est pas numérique"
fi
```

2. Écrire un script, en utilisant `case`, qui effectue les opérations mathématiques :

   - L'addition +,
   - La soustraction −,
   - La multiplication x,
   - La division /.

   Le nom de script doit être `script4` qui fonctionne comme suit `./script4 20 / 3`. Vérifier que le nombre d'arguments est correct et que le 1-er et le 3-ème arguments sont interprétables numériquement.

Réponse :

```
#!/bin/bash
if [ $# -eq 3 ]
then
    if expr $1 + $3 >/dev/null 2>&1
    then
        case $2 in
            +) echo "$1 + $3 = `expr $1 + $3`";;
            -) echo "$1 + $3 = `expr $1 - $3`";;
            x) echo "$1 * $3 = `expr $1 \* $3`";;
            /) if [ $3 -ne 0 ]
               then
                    echo "$1 / $3 =`expr $1 / $3`"
               fi;;
            *) echo "Opération non reconnue";;
        esac
        echo "$1 $2 $3 = $res"
    fi
fi
```

Exercice 5 : Utilisation de la fonction select

Vous allez à l'aide de la fonction select réaliser un menu à 4 options pour un utilisateur. Le script doit boucler tant que l'utilisateur n'a pas choisi de quitter.

- Option 1 : Afficher la liste des utilisateur connectés (en utilisant la commande `who`)
- Option 2 : Afficher la liste de tous les processus

- Option 3 : Afficher la date
- Option 4 : Terminer

Réponse :

```
#!/bin/bash
select choix in "Afficher la liste des utilisateur connectés"\
                "Afficher la liste de tous les processus"\
                "Afficher la date"\
                "Terminer"
do
    case $REPLY in
        1) who;;
        2) ps aux;;
        3) date;;
        4) break;;
        *) echo "Choix invalide";;
    esac
done
```

Exercice 6 : Utilisation de la commande `exit`

Ecrire un script Shell appelable soit sans arguments, soit avec trois arguments et telle que, appelée sans arguments, il réalise la lecture au clavier de trois chaînes de caractères. Disposant alors dans tous les cas de trois chaînes, il indique si les trois chaînes sont identiques, si deux de ces chaînes sont identiques ou si elles sont toutes différentes par un message sur la sortie-erreur-standard (utiliser la commande **exit**). Le code de retour de la commande sera égal à 0 (**exit 0**) si les trois chaînes sont égales, 1 (**exit 1**), 2 (**exit 2**) ou 3 (**exit 3**) selon que la chaîne en i-ème position est différente des deux autres (celles-ci étant identiques), 4 si les trois chaînes sont différentes et 5 si le nombre de paramètres d'appel est incorrect.

Réponse :

```
#!/bin/bash
if [ $# -eq 0 ]
then
    read -p "Donner trois chaines de caractères : " ch1 ch2 ch3
elif [ $# -eq 3 ]
then
    ch1=$1
    ch2=$2
    ch3=$3
else
    exit 5
fi
if [ $ch1 = $ch2 -a $ch1 = $ch3 ]
then
    exit 0
elif [ $ch2 = $ch3 ] # ici, ch1 est différente des deux autres chaines
then
    exit 1
elif [ $ch1 = $ch3 ] # ici, ch2 est différente des deux autres chaines
then
    exit 2
```

```
elif [ $ch1 = $ch3 ] # ici, ch3 est différente des deux autres chaines
then
    exit 3
else # ici, les trois chaines sont différentes
    exit 4
fi
```

Exercice 7 : Utilisation de la commande `getopts`

1. Réalisez un script appelé "**Convertisseur**" qui vous permettent de convertir en Euros une somme en Dirhams passée en argument. Le taux de conversion sera contenu dans un fichier nommé **Taux** sous la forme : Taux :10.64. Pensez à tester si ce fichier est existant et lisible par votre processus. Si ce n'est pas le cas, votre script doit afficher une erreur et quitter en indiquant un code retour 1.

2. Modifiez votre script afin de pouvoir traiter l'option `-e` qui permet d'inverser la conversion (Euros vers Dirhams). En cas d'erreur de syntaxe, le script affiche l'usage du programme et quitte avec un code retour égal à 2.

3. Modifiez votre script afin de pouvoir traiter l'option `-f FILE` qui permet d'indiquer un fichier dans lequel se trouve plusieurs sommes à convertir (une par ligne). Les résultats des conversions seront placés dans un fichier nommé "**Resultats**".

Réponse :

```
#!/bin/bash

# Convertisseur Euros/Dirhams et Dirhams/Euros
function usage
{
    echo "Convertisseur Euros/MAD"
    echo "Usage : Convertisseur [-e] [-f arg] argument"
    echo "Convertit des Dirhams en Euros par defaut"
    echo "-e : inverse la conversion"
    echo "-f : donne un fichier de somme ⌧ convertir"
    echo "argument : valeur ⌧ convertir"
    exit 2
}
CONV=0
F=0

if [ -e Taux -a -r Taux ]
then
    #lecture du Taux de conversion
  IFS=:
  read c TAUX < Taux
    #Traitement des arguments
    while getopts ":ef:" opt
    do
        case $opt in
            e) CONV=1;;
            f) F=1
                FILE=$OPTARG;;
            ?) usage
        esac
```

```
    done
    shift `expr $OPTIND - 1`
    case $F in
    # Y a t'il une somme ⌧ convertir
        0) if [ $# != 1 ]
                then usage
           fi
           case $CONV in
                0) RES=`echo $1/$TAUX | bc -l`
                   echo "$1 Dirhams est egal a $RES Euros";;
                1) RES=`echo $1*$TAUX | bc -l`
                   echo "$1 Euros est egal a $RES Dirhams";;
                *) ;;
            esac
            exit 0;;
    # Le fichier existe t'il ?
        1) if [ -e $FILE -a -r $FILE ]
            then
                for I in `cat $FILE`
                do
                    case $CONV in
                        0) RES=`echo $I/$TAUX | bc -l`
                           echo "$I Dirhams est egal a $RES Euros" >> Resultats;;
                        1) RES=`echo $I*$TAUX | bc -l`
                           echo "$I Euros est egal a $RES Dirhams" >> Resultats;;
                        *) ;;
                    esac
                done
                exit 0;
            else
                usage
                exit 2
            fi ;;
        *) ;;
    esac
else
    echo "!!! Erreur !!!\n"
    echo "\tLe fichier Taux est inaccessible !!!\n"
    exit 1
fi
```

Exercice 8 : Création de fonction shell

1. En utilisant les structures que vous connaissez, écrire un script qui affiche la table de multiplication d'un nombre donné en paramètre. Exemple `mul 4`, donne la table de multiplication de 4. Vous afficherez les résultats pour un multiplicateur allant de 1 à 10. L'affichage de la table de multiplication sera réalisé par une fonction `affTABLE()`.

2. Modifiez le script afin que les bornes du multiplicateur soient passés en paramètres.
   Exemple : `mul 3 25 35`. On veut la table de multiplication de `3*25` jusqu'à `3*35`

Réponse :

```
#!/bin/bash
```

```
 2
 3  function affTable() {
 4    a=$2
 5    b=$3
 6    while [ $a -le $b ]
 7    do
 8        echo "$1 * $a = `expr $1 \* $a`"
 9        a=`expr $a + 1`
10    done
11  }
12
13  if [ $# -eq 1 ]
14  then
15     affTable $1 1 10
16  elif [ $# -eq 3 ]
17  then
18     affTable $*
19  else
20     echo "Nombre d'arguments incorrect"
21     exit 1
22  fi
```