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.

Solution 1 : script.sh

```
#!/bin/bash
read -p "Donnez un nom de répertoire : " repertoire
if [ ! -d $repertoire ]
then
   mkdir $repertoire
fi
cd $repertoire
echo "On est dans le répertoire : `pwd`"
```

Solution 2 : script.sh

```
#!/bin/bash
read -p "Donnez un nom de répertoire : " repertoire
[ -d $repertoire ] || mkdir $repertoire
cd $repertoire
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).

Solution 1 : protege.sh

```
#!/bin/bash
read -p "Donnez un nom de fichier : " fichier
if [ -e $fichier ]
then
   chmod go-rwx $fichier
fi
```

Solution 2 : protege.sh

```
#!/bin/bash
read -p "Donnez un nom de fichier : " fichier
[ -e $fichier ] && chmod go-rwx $fichier
```

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`.

Solution 1 : effacer.sh

```
#!/bin/bash
read -p "Donnez un nom de fichier : " fichier
if [ -f $fichier ]
then
   if [ ! -d ~/poubelle ]
   then
       mkdir ~/poubelle
   fi
   mv $fichier ~/poubelle
fi
```

Solution 2 : effacer.sh

```
#!/bin/bash
read -p "Donnez un nom de fichier : " fichier
[ -f $fichier ] && ([ -d ~/poubelle ] || mkdir ~/poubelle) && mv $fichier ~/poubelle
```

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.

Solution 1 : nbfich.sh

```
#!/bin/bash
if [ $# -eq 0 ]
then
    rep=.
elif [ $# -eq 1 ]
then
    if [ -d $1 ]
    then
        rep=$1
    else
        echo "$1 n'est pas un répertoire"
        exit 1
    fi
else
    echo "Nombre de paramètres incorrect"
    exit 2
fi
echo "Nombre de fichiers du répertoire $rep est `ls $rep | wc -l`"
```

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.

Solution 1 : max.sh

```
#!/bin/bash
if [ $# -eq 3 ]
then
    max=$1
    if [ $2 -gt $max ]
    then
        max=$2
    fi
    if [ $3 -gt $max ]
    then
        max=$3
    fi
    echo "Max($1,$2,$3)=$max"
else
    echo "Il faut exactement trois paramètres"
    exit 1
fi
```

Solution 2 : max.sh

```
#!/bin/bash
if [ $# -eq 3 ]
then
    max=$1
    [ $2 -gt $max ] && max=$2
    [ $3 -gt $max ] && max=$3
    echo "Max($1,$2,$3)=$max"
else
    echo "Il faut exactement trois paramètres"
    exit 1
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?
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 2-ème arguments sont interprétables numériquement.

Exercice 5 : Utilisation de la commande 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

Solution 1 : script_select.sh

```
#!/bin/bash
select opt 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 "Il faut choisir 1, 2, 3 ou 4";;
    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.

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 script. 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".

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 3×5 jusqu'à 3×35