Correction TP4 : Programmation Shell (1)

Exercice 1 :

Ecrivez un shell script appelé « mon_script » qui affiche son nom, le nombre de paramètres fournis, la liste des paramètres, la valeur du premier paramètre, ainsi que les valeurs de deux variables : l'une initialisée et l'autre saisie.

Il existe plusieurs shells, chacun disposant de spécificités propres. Le Bourne Shell (sh) est le shell le plus connu et le plus courant sur les Unix. Le C-Shell (csh) reprend la structure du langage C. Le Korn Shell (ksh) est une évolution du Bourne Shell. Le Z-Shell (zsh) est lui-même une évolution du Korn Shell. Le shell de référence sous Linux se nomme le Bourne Again Shell (bash), etc.

Il existe deux méthodes pour exécuter un script shell :

1) Accorder le droit x à l'utilisateur : **chmod u+x f1 (ou f1.sh)**

   puis l'exécuter avec **./f1 (ou f1.sh)**

2) forcer l'exécution du script à partir d'un shell : **sh f1 (ou f1.sh)**

```
# !/bin/sh
echo $0
echo $#
echo $*
echo $1
a=5
echo $a
read b
echo $b
```

Exercice 2 :

Ecrivez un script shell qui teste si le paramètre passé :

- Est un fichier
- Est un répertoire

Si aucun paramètre n'est passé, on affichera un message d'erreur indiquant le mode d'emploi du script.

```
# !/bin/sh
If [$# -eq 0]
then
echo vous devez saisir un paramètre
elif [$# -eq 1]
then
      If [ - f $1]
      then
      echo $1 est un fichier
      elif [ - d $1]
      then
      echo $1 est un répertoire
```

```
      fi
else
echo trop de paramètre
fi
```

Exercice 3 : (pour introduire la structure case)

Ecrire un script Shell «**Compta**» permettant de compter le nombre de caractères ou de mots ou de lignes d'un fichier donné en argument. Le type d'éléments à compter est introduit interactivement par l'utilisateur.

```
#!/bin/sh
echo "si vous voulez compter les caractères du fichier tapez 1 "
echo "si vous voulez compter les mots du fichier tapez 2 "
echo "si vous voulez compter les lignes du fichier tapez 3 "
read choix
case $choix in
1) echo le nombre de caractere est `wc -c  $1` ;;
2) echo le nombre de mots est `wc -w  $1`;;
3) echo le nombre de lignes est `wc -l  $1`;;
*) echo " erreur de parametre"
esac
```

Exercice 4 : (pour introduire la boucle for et les commandes expr et let)

Ecrire un script shell qui calcule la somme de ses paramètres :

./somme 1 2 3 4 5

total = 15

```
#!/bin/sh
if [ $# -gt 0 ]
then
    total=0
    for fich in $*
     do
       total=`expr $total + $fich `
       # let total=total+fich
       # total=$(expr $total + $fich)
            done
   echo "total = $total"
else
     echo "erreur de paramètres"
fi
```

Exercice 5 : (pour introduire les commandes head, tail et shift)

1. Ecrire un script shell qui affiche la nième ligne (premier paramètre du script) du fichier donné comme 2ième paramètre du script (proposer deux solutions).

```
#!/bin/sh
if [ $# -eq 2 ]
then
 i=0
 for lg in `cat $2`
 do
     i=`expr $i +1`
```

```
      if [ $i -eq $1 ]
            echo le $i ème élément de cette liste est :
            echo $lg
      fi
 done
else
 echo erreur de paramètres
fi
******* ou *******
#!/bin/sh
if [ $# -eq 2 ]
then
head -$1 $2 |tail -1
else
 echo erreur de paramètres
fi
```

2. Modifier le script précédent pour pouvoir accepter plus d'un fichier.

```
#!/bin/sh
if [ $# -gt 2 ]
then
        num=$1
        shift
        for fich in $*
        do
            head -$num $fich |tail -1
        done
  else
        echo "erreur de paramètres"
  fi
```

Exercice 6 : (introduire les commandes tr et sed)

1. Ecrire un script Shell "**traA**" permettant de transformer tous les caractères "a" en "A" des noms d'une série de fichiers passés en argument.

```
#!/bin/sh
if [ $# -gt 0 ]
then
      for fich in $*
      do
              if [ -f $fich ]
              then
                      echo $fich | tr 'a' 'A'
              fi
     done
  else
      echo "erreur de paramètres"
    fi
```

2. Modifier ce script pour pouvoir transformer toutes les chaînes de caractères "aba" en "Aba".

```
#!/bin/sh
if [ $# -gt 0 ]
then
      for fich in $*
```

```
        do
                if [ -f $fich ]
                then
                        echo  $fich >> F
                fi
            done
        sed  s/aba/Aba/g  F
         rm F
    else
        echo "erreur de paramètres"
    fi
```

Exercice 7 : (introduire la commande grep)

1. Ecrire un script Shell "**ListeFich**" qui permet d'afficher toutes les informations des fichiers ordinaires d'un répertoire passé en argument. Traiter, par défaut, le cas du répertoire courant.

```
   #!/bin/sh
   if [ $# -gt 1 ]
   then
        echo Trop de paramètres
   else
    if [ $# -eq 1 ]
    then
         echo la liste de fichiers est :
         ls -Rl $1| grep '^-'
    elif [ $# -eq 0 ]
    then
         ls -Rl | grep '^-'
    fi
fi
********************ou ****************
   #!/bin/sh
   if [ $# -gt 1 ]
   then
        echo Trop de paramètres
   else
    if [ $# -eq 1 ]
     then
       rep=$1
    elif [ $# -eq 0 ]
    then
       rep=.
    fi
         ls -Rl $rep| grep '^-r'
fi
```

2. Modifier ce script pour n'afficher que les fichiers accessibles en lecture.

```
   #!/bin/sh
   if [ $# -gt 1 ]
   then
        echo Trop de paramètres
   else
```

```
 if [ $# -eq 1 ]
 then
    rep=$1
 elif [ $# -eq 0 ]
 then
    rep=.
 fi
ls -Rl $rep| grep '^-r'| tr -s ' ' | cut -d' ' -f9
fi
```

3. Modifier ce script pour n'afficher que les fichiers dont le nom est composé au plus de cinq caractères.

```
#!/bin/sh
if [ $# -gt 1 ]
then
     echo Trop de paramètres
else
 if [ $# -eq 1 ]
 then
    rep=$1
 elif [ $# -eq 0 ]
 then
    rep=.
 fi
ls -Rl $rep| grep '^-r'| tr -s ' ' | cut -d' ' -f9 | grep
'^.\{1,5\}$'
 fi
```

4. Afficher cette liste triée et en majuscule.

```
#!/bin/sh
if [ $# -gt 1 ]
then
     echo Trop de paramètres
else
 if [ $# -eq 1 ]
 then
    rep=$1
 elif [ $# -eq 0 ]
 then
    rep=.
 fi
ls -Rl $rep| grep '^-r'| tr -s ' ' | cut -d' ' -f9 | grep
'^.\{1,5\}$'| sort | tr 'a-z' 'A-Z'
 fi
```

5. Afficher le nième fichier dans cette liste.

```
#!/bin/sh
#!/bin/sh
if [ $# -gt 1 ]
then
     echo Trop de paramètres
else
 if [ $# -eq 1 ]
```

```
   then
      rep=$1
   elif [ $# -eq 0 ]
   then
      rep=.
   fi
  echo Donner le numéro de la ligne :
  read num
ls -Rl $rep| grep '^-r'| tr -s ' ' | cut -d' ' -f9 | grep
'^.\{1,5\}$'| sort | tr 'a-z' 'A-Z'| head -$num |tail -1
fi
```