Programmation en SHELL BASH sous Linux

Travaux pratiques

Avant de commencer

- Vous devez savoir au minimum comment est constitué le SGF de Linux et connaître les commandes suivantes :
- who, id
- ls, cd, mkdir, rmdir, rm, cp, mv, pwd, ln
- wc, tail, head, sort, more, cat, less, tar, gzip, grep, lpr
- chmod, chown, chgroup, umask, adduser, userdel, connaître la structure des fichiers /etc/group et /etc/passwd
- which, locate, find
- clear, date, echo, df, du, top, kill, killall, alias
- connaître les fichiers standards (stdin, stdout, stderr), les opérateurs de redirections (<, >, >>), le fonctionnement des pipes.
- set, ps, hostname, les principales variables environnement et les principaux fichiers "~/.bash_*". Déclarer une varaiable environnement, exporter cette variable.
    - .bash_history, .bash_profile,.bashrc, .bash_logout
- La commande "man"

Le quoting

- Tout ce qui est placé entre "" est lu littéralement sauf `, $ et "
- Tout ce qui est placé entre ' (accent aigu) est lu litééralement, sauf un autre accent aigu.
- Pour déspécialiser un caractère utiliser le \ (back slashe)

Les structures de contrôles

if, then elif, fi

```
if expression
  then
     intruction1...
     instructionN
   elif
   then
     ....
   else
     ...
fi
```

case

```
case chaîne

in
       regex)
              commandes
              ...
              ;;
              ...
       esac
```

for

```
for x [ in list ]
do
   commandes
done
for $i in * do
       pr $i > $i.tmp
done


for $i in `cat liste` do
       ls $i
done
```

let

```
Initialisation (ajouter 1 à i)
let i=i+1
let "i = i + 1"
```

read

```
Lecture d'une valeur au clavier
echo -n "Entrez votre nom :"
read nom
echo $nom
```

select

```
select nom [ in liste ; ]
do
        commandes
done
```

- Le numéro du choix est affecté à $REPLY, le nom du choix à $nom

```
select choix in  \
        "Choix A" \
        "Choix B";
        do
               case $REPLY  in
                        1)      echo $choix ;;
                 2)      echo $choix ;;
                 *)      echo "Vous avez tapé n'importe quoi !";;
                 esac
        done
```

Fonctions

```
function commande { ...}
        On accède aux paramètres avec $1...$n

function somme {
 resultat = `expr $1 + $2`
}
```

echo

- echo -n désactive le retour de chariot de fin de chaîne
- echo -e active l'interprétation des séquences d'échappement (\a bell).
- Exemple : echo -e 'foo \a '

test

- test expression ou alors [expression]. Retourne 0 si Vrai, une autre valeur dans les autres cas.
- Test sur les fichiers
- -d, si c'est un répertorie
- -e, si le fichier existe
- -f, si le fichier existe et si c'est un fichier standard
- Test sur les chaînes
- s1 == s2, si les chaînes s1 et s2 sont identiques
- s1 != s2, si les chaînes sont différentes.
- Test sur les entiers
-
- n1 -eq n2, si n1 est égal à n2
- opérateurs -ne, -eq, -gt, -lt, -le -ge

Répéter jusqu'à

```
until
        Commande_Test
do
        Commandes
done
```

Tant Que

```
while
        Commande_Test
do
        Commande
done
```

Exercices

TP1 utilisation de la fonction TEST

- Ecrivez un script qui dit si le paramètre passé est :

- un fichier
- un répertoire

- n'existe pas

- Ecrivez un script qui n'affiche que les répertoires

- Ecrivez un script qui n'affiche que les fichiers

- Ecrivez un script qui donne le nombre de fichiers et de répertoires

TP2 Utilisation de la fonction selon que (case)

- En utilisant la strucure case, écrire un script qui :

- affiche un menu
- demande à l'utilisateur de saisir une option du menu

- affiche à l'utilisateur l'option qu'il a choisi

- Exemple de ce qui doit s'afficher à l'écran :

- ********** Menu général **********

- <1> Comptabilité

- <2> Gestion commerciale

- <3> Paie

- <9> Quitter

- **

TP3 Utilisation de la fonction pour (for)

- En utilisant la structure for, écrire un programme qui donne les valeurs de y d'une fonction pour les valeurs de x allant de -10 à 10 avec un incrément de 1.
- Réalisez le traitement pour les fonctions y=x, y = x puiss2
- Réécrivez les programmes avec la strucure répéter ... jusqu'à
- Adapter le script afin que les bornes -x, +x soient passées en paramètres au script.
- Modifiez le script de façon à ce que l'on puisse passer en paramètres l'incrément.

TP4 Etude de la fonction si (if)

- *1)* En utilisant la strucure if, écrire un script qui :
- affiche un menu
- demande à l'utilisateur de saisir une option du menu
- affiche à l'utilisateur l'option qu'il a choisi
- Exemple de ce qui doit s'afficher à l'écran :
- ********** Menu général **********
- <1> Comptabilité
- <2> Gestion commerciale
- <3> Paie
- <9> Quitter
- **************************************
- 2) Vous allez utiliser un fichier dans lequel vous stockerez les informations suivantes :
- premier 3
- deuxième 10
- troisième 25
- quatrième 2
- cinquième 12
- Construisez un script qui permet de n'afficher que les enregistrements dont la valeur est supérieure à 10.

TP5 Etude de la fonction répéter jusqu'à (until...do...done)

- -*A*- En utilisant la strucure until...do...done, écrire un script qui :

- demande à un utilisateur de saisir une commande
- exécute la commande ou affiche un message d'erreur si la commande ne s'est pas déroulée.

- répète cette opération tant que l'utilisateur le désire.

- Exemple de ce qui doit s'afficher à l'écran :

- **********

- Saisisez une commande, commande <Q> pour quitter.

- **********

TP6 Etude de la fonction tant que (while)

- En utilisant la structure while, écrire un script qui :

- Tant que l'utilisateur n'a pas tapé 9

- affiche un menu
- demande à l'utilisateur de saisir une option du menu

- affiche à l'utilisateur le résultat de sa commande

- Exemple de ce qui doit s'afficher à l'écran :

- ********** Menu général **********

- <1> Afficher la date (date)

- <2> Afficher le nombre de personnes connectées (who)

- <3> Afficher la liste des processus (ps)

- <9> Quitter

- **********

TP7 Etude de la fonction select

- -*A*- 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
- Option 2 : Afficher la liste des processus

- Option 3 : Afficher à l'utilisateur son nom, son UID, son GID, son TTY1

- Option 4 : Terminer

- -*B*- Modifier le script afin de rajouter un Option. Cette option doit vous permettre de vous envoyer un mail. Le contenu du mail sera le résultat d'une commande. Vous utiliserez la commande : mail VotreCompte < `laCOMMANDE`. Exemple : mail mlx < `who` Les commandes sont celles saisies dans les choix 1, 2 ou 3

TP8 Création de fonction shell

- -*A*- 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().

- -*B*- 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

- -*C*- Modifier le programme de façon à écrire une calculatrice. L'utilisateur saisi un nombre (par exemple 3). Ensuite il saisira des couples opérateur nombre (exemple + 3). Le programme réalisera les opérations jusqu'à ce que l'utilisateur tape l'opérateur "=" et affichera le résultat final.

TP9 Traitement des options de la ligne de commande

- -*A*-Vous utiliserez la fonction getopts pour vérifier la saisie de l'utilisateur. Réaliser un script d'archivage qui utilisera les options :

- -a (archive)
- -d (désarchive)

- -c (compresse)

- -x (décompresse)

Le fichier ou le répertoire à archiver sera passé en paramètre : exemple archive -a -c travail. Attention archive -a -d est invalide.

- -*B*-Modifier le script en utilisant des fonctions. Vous factoriserez au maximum ce qui est répétitif.

- -*C*- Modifiez le script afin de pouvoir passer plusieurs fichiers à compresser. Exemple archive -a -c fichier1 fichier2 fichier3

Remarque

Pour archiver vous exploiterez la commande tar (uniquement sur les répertoires car il est inutile d'archiver un fichier). Pour compresser gzip.

Corrections

TP 1 - Utilisation de TEST

```
#-------------------------------------------------------
#Ecrire un script qui dit si le fichier passé
#en paramètre et un fichier, un répertoire ou autre chose
#-------------------------------------------------------

# Si le paramètre est nul on envoie un message
if [ -z $1 ] ; then                     # voir aussi -n
        echo "Vous n'avez pas entré de paramètres"
        echo "Le mode d'utilisation du script est $0 NomDuFichier"
        exit 0
fi

if [ -f $1 ] ; then                             # alors c'est un fichier
        echo "$1 est un fichier"
        exit 0
fi

if [ -d $1 ] ; then                     #c'est un répertoire
        echo "$1 est un répertoire"
        exit 0
fi

echo "Il semble que $1 ne soit ni un fichier ni un répetoire ou alors
n'existe pas."
#-------------------------------------------------------
# TP 1 deuxième partie
# On utilisera la commande `ls` qui retourne les fichiers
#-------------------------------------------------------

#Donner la liste des fichiers fichiers
for i in `ls` ; do
        echo $i
done


# Ne donner que les fichiers standards
j=0
for i in `ls` ; do
        if [ -f $i ]; then
         j=`expr $j + 1 `
        fi
done
echo "Il y a $j fichiers standards dans le répertoire"


# Ne donner que les répertoires
j=0
for i in `ls` ; do
        if [ -d $i ]; then
         j=`expr $j + 1 `
        fi
done
        echo "Il y a $j  répertoires dans le répertoire"
```

```
# Donner le nombre de fichiers, première solution
# on utilise un indice
j=0
for i in `ls` ; do
         j=`expr $j + 1`
done
echo "Il y a $j fichiers dans le répertoire"


# Donner le nombre de fichiers deuxième solution
# on utilise les commandes systèmes
j=`ls | wc -l`
         echo "Il y a $j fichiers dans le répertoire"
```

TP 2 - Utilisation de CASE

```
clear

echo
"------------------------------------------------------------------------"
echo "<1>               Comptabilité"
echo "<2>               Gestion commerciale"
echo "<3>               Paie"
echo ""
echo ""
echo ""
echo "Taper une option du menu 1, 2 ou 3, <q> pour quitter"

read saisie

case $saisie
        in
         1)
                echo "Vous avez choisi l'option 1 Comptabilité" ;;
        2)
                echo "Vous avez choisi l'option 2 Gestion Commerciale" ;;
        3)
                echo "Vous avez choisi l'option 3 Paie" ;;

        q|Q)
                echo "Vous avez choisi de quitter l'application" ;;

        *)      # Tous les autres cas
                echo "Vous avez saisi un peu n'importe quoi" ;;
esac
```

TP 3 - Utilisation de la structure for et do...until

```
#Première partie (avec for)

# Traitement de la fonction x=y
# pour x allant de -10 à 10 avec un incrément de 1
# On stockera dans un fichier inc toutes les valeurs
# comprises entre -10 et 10

for i in `cat inc`; do
        x=$i
        y=$x
        echo "Pour x valant $x, y vaut $y"
done

# Traitement de la fonction y= x puiss 2

for i in `cat inc`; do
        x=$i
        y=`expr $x \* $x`
        echo "Pour x valant $x, y vaut $y"
done
#Deuxième partie (avec répéter)
# fonction x = y
echo -------- utilisation de until...
x=-10

until [ $x -eq 10 ]; do # on regarde si x est égal = 10
        y=$x
        echo "Pour x valant $x, y vaut $y"
        x=`expr $x + 1` # on incrémente
done

# fonction y = x puiss 2
echo -------- utilisation de until...
x=-10
abs=10
y=`expr $x \* $x`               # On calcule la première occurence

until [ $x -gt 10 ]; do         # on regarde si x est égal = 10
        echo "Pour x valant $x, y vaut $y"
        x=`expr $x + 1`         #on incrémente
        y=`expr $x \* $x`       #on recalcule y
done
#La suite du TP n'est plus que la reprise de cette dernière partie
# on traite $1 $2 $3 $4... borne moins, borne plus, par
# Exemple: monSCRIPT -100 100 4 (pas de 4)

x=$1
y=`expr $x \* $x`               # On calcule la première occurence

until [ $x -gt $2 ]; do         # on regarde si x est égal = 10
        echo "Pour x valant $x, y vaut $y"
        x=`expr $x + $3`        #on incrémente selon le pas souhaité
        y=`expr $x \* $x`       #on recalcule y
done
```

TP 4 - utilisation de la structure si

```
#-----------------------------------------------------------
# Programmation shell
# Utilisation de la structure si
# utiliser -a pour "et", -o pour "ou", ! pour non
# Donner les éléments sur la commande set -- pour éclater
# les mots d'une ligne
# Donner les éléments pour réaliser la lecture d'un fichier
#-----------------------------------------------------------
clear
echo
"-------------------------------------------------------------------------"
echo "<1>               Comptabilité"
echo "<2>               Gestion commerciale"
echo "<3>               Paie"
echo ""
echo ""
echo ""
echo "Taper un chiffre une option du menu, <q> pour quitter"

read saisie

if [ $saisie == 1 ]; then
        echo "Vous avez choisi l'option 1 Comptabilité"
elif [ $saisie == 2 ] ; then
        echo "Vous avez choisi l'option 2 Gestion Commerciale"
elif [ $saisie == 3 ] ; then
        echo "Vous avez choisi l'option 3 Paie"
elif [ $saisie == q  -o $saisie == Q ] ; then
        echo "Vous avez choisi de quitter l'application"
else
        echo "Vous avez tapé n'importe quoi !!!"
        echo $'\a'      # on beep
fi
#Deuxième partie traitement d'un fichier

cat donnees | while true; do    # on dump le fichier
read ligne
        if [ "$ligne" == "" ] ;then
         exit 0                 #On a atteint la fin de fichier
                                #Attention il ne faut pas de lignes vides
                                #dans le fichier
        fi
set -- $ligne                   #On split les valeurs de la ligne lue
                                #dans des variables $1, $2...
                                #voir man bash pour la commande set
                                #il ne reste plus qu'à afficher
        if [ $2 -ge 10 ]; then  #si supérieur ou egal on affiche
           echo -e "$1 \t $2"   #-e pour avoir le caractère de tabulation
        fi
done
```

TP 5 - Utilisation de la structure répéter jusqu'à

```
clear
saisie=xxx
set -- $saisie
until [ $1 == q -o $1 == Q ]; do
        echo "Entrez une commande"
        read -r saisie  #récupérer toute la ligne avec les paramètres
#       $saisie         #Première solution : Exécution de la commande
        eval $saisie    #Deuxième solution (préférable)
#       exec $saisie    #Troisième solution mais quitte le shell
#       echo $saisie    #Pour debogguer
        set -- $saisie  #on split la ligne de commande
                        #pour éviter les msg d'erreurs sur le test
done
```

TP 6 - Utilisation la structure tantque...

```
reponse=xxx
while [ $reponse != q -a $reponse != Q ]; do
        clear
        echo "-------- Menu général ---------"
        echo -e "\n"
        echo "<1>       Comptabilité"
        echo "<2>       Gestion commerciale"
        echo "<3>       Paie"
        echo -e "\n"
        echo "-------------------------------"
        echo -n "Choisissez une option, <q> pour terminer "
        read reponse

        case $reponse in
        1)      echo "Vous avez choisi la Compta" ;;
        2)      echo "Vous avez choisi la Gestion commerciale" ;;
        3)      echo "Vous avez choisi la Paie";;
        q|Q)    echo "Vous allez quitter l'application";;
        *)      echo "Vous avez tapé n'importe quoi !";;
        esac
        echo -n "Tapez [ENTER] pour continuer"
        read
done
```

TP 7 - Utilisation de la fonction select ...

```
select choix in  \
        "Affiche la listes des utilisateurs connectés" \
        "Affiche la liste des processus"\
        "Affiche les informations vous concernant"\
        "QUITTER"
         do
                case $REPLY  in
                1)      who ;;
                2)      ps ax ;;
                3)      echo -e "Bonjour $USER , voici les informations \n
`id`";;
                4)      echo "Vous allez quitter l'application"
                        exit 0 ;;
                *)      echo "Vous avez tapé n'importe quoi !";;
                esac
        done
```

TP 8 - Utilisation de fonctions

```
#Utilisation de fonctions

#Première partie : traitement d'une table de multiplication

affTABLE0 () {
i=1
while [ $i -le 10 ] ; do
        echo -e "$1 * $i \t = `expr $1 \* $i`"
        i=`expr $i + 1`
done
}


#Ici le premier programme principal

affTABLE0  $1   # On passe le paramètre à la fonction
#Deuxième partie on modifie la fonction
affTABLE1 () {
i=$2
while [ $i -le $3 ] ; do
        echo -e "$1 * $i \t = `expr $1 \* $i`"
        i=`expr $i + 1`
done
}

#Ici le deuxième programme principal
affTABLE1 $1 $2 $3
```

```
#Troisième partie la calculatrice
#Troisième partie on modifie la fontion
calcule () {
echo "---------------> $1 --- $2 --- $3"
case $2 in
        "\+")   nombre1=`expr $nombre1 + $nombre2`;;
        "\-")   nombre1=`expr $nombre1 - $nombre2`;;
        "\/")   nombre1=`expr $nombre1 / $nombre2`;; #Attention à la
division par 0
        "\*")   nombre1=`expr $nombre1 \* $nombre2`;;
        *)      echo "Erreur de saisie"
                exit 0;;
esac
}


#Ici le programme principal
clear
echo "C'est parti .............saisissez un nombre"
read nombre1    #On lit le premier nombre
echo -n "Entrez vos opérations (exemple + 3) "
masque=\\                               #Utilisé pour déspécialiser les
caractères
read op nombre2
while [ $op != '=' ] ; do
                                        #On lit l'opérateur et le second
nombre
                                        #Il y a un pb avec le caractère *
                                        #qu'à cela ne tienne on le masque
                                        #en les déspécialisant
        calcule $nombre1 $masque$op $nombre2
                                        #l'Opérateur est masqué
                                        #Cela aurait également posé un pb
pour le case
                                        # * correspond à tous les cas non
traités (par défaut)
        echo $nombre1                   # On affiche le résultat
        read op nombre2
done
echo "= $nombre1"
echo "Terminé ............."
```

TP 9 - Utilisation de getopts et optind

```
#Utilisation de getopts et optind
#On ne traite pas les incompatibilités ou erreurs
# de saisie, ce n'est pas l'opjet du TP

if [ -z $1 ]; then
        echo "tar/compress $0 -t -c fichier "
        echo "untar/décompresse $0 -u -d fichier "
        exit 0
fi

#On regarde si le fichier existe, il est en deuxième position
#s'il n'y a qu'une option de passée, sinon il est en troisième
#position sur la ligne de commande.
if [ $# -eq 2 ]; then
         if [ ! -f $2 ]; then
                echo "Le fichier $2 ne semble pas exister"
        fi
elif [ $# -eq 3 ] ; then
        if [ ! -f $3 ]; then
                echo "Le fichier $2 ne semble pas exister"
        fi
else    #il y a un problème sur la ligne de commande
        clear
        echo "Vous avez commis une erreur sur la ligne de saisie"
        exit 0
fi




#On récupère le nombre d'options passées
for i in $@ ; do
        getopts tc option # On récupère la première option t ou c

         if [ $option == 't' ] ;  # Ici c'est pour archiverr
           then
                for i in $@ ; do
                 getopts c option # On regarde s'il faut décompresser
                done
                 if [ $option == 'c' ] ; then #il faut également compresser
                 else
                  echo "ici compléter"
                fi

        elif [ $option == 'u' ] ; then
                for i in $@ ; do
                 getopts d option # On regarde s'il faut décompresser
                done
                 if [ $option == 'd' ] ; then
                  echo "ici compléter"
                 else
                  echo "ici compléter"
                fi
        fi

done
```