Programmation en SHELL BASH sous Linux

Alix MASCRET

Séquences de TP sur la programmation en SHELL BASH sous Linux.

Éléments de corrigé pour les exercices.

1. BASH - Programmation en SHELL

1.1. TP 1 - Utilisation de test

```
#!/bin/bash
#-------------------------------------------------------
# TP 1 première partie
# 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."
```

```
#!/bin/bash
#-------------------------------------------------------
# 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"
exit 0
```

1.2. TP 2 - Utilisation de `case`

```
#!/bin/bash
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
exit 0
```

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

```
#!/bin/bash
#
#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
echo "Traitement de la fonction y= x"
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
echo "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
exit 0


#!/bin/bash
#
# Deuxième partie (avec répéter)
#
# fonction x = y
echo -------- utilisation de until x = y...

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 y = x^2...

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
exit 0
```

```
#!/bin/bash
#
# 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)
#
prog=$0
x=$1
z=$2
p=$3
shift;shift;shift
OTHERS=$*

usage () {
        echo "usage: $prog borne_min borne_max pas ($prog -100 100 4)"
        exit 1
}

if [ x$x == "x" ]; then
        usage
fi
if [ x$z == "x" ]; then
        usage
fi
if [ x$p == "x" ]; then
        usage
fi

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

until [ $x -gt $z ]; do         # on regarde si x est égal = 10
        echo "Pour x valant $x, y vaut $y"
        x=`expr $x + $p`        #on incrémente selon le pas souhaité
        y=`expr $x \* $x`       #on recalcule y
done
exit 0
```

1.4. TP 4 - utilisation de la structure si

```
#!/bin/bash
#
#-------------------------------------------------------------
# Alix MASCRET éléments de corrigé
# 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
#Préférer l'utilisation du code retour du script
#pour tester la fin de fichier, ça permet d'avoir des lignes
#vides dans le 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
```

1.5. 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
```

1.6. TP 6 - Utilisation la structure tant que...

```
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
```

1.7. 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
```

1.8. 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 ............"
```

1.9. TP appels et contrôles de scripts

```
################################################
# Ici commence le programme principal
################################################

# On crée une fonction qui récupère le PID
# du script enfant

_pid() {
 echo -n "On kill le processus N° `cat $1` "
 echo "avec la commande  \"kill -9 `cat $1`\"" # On tue le processus
 kill -9 `cat $1`
}


#On se fait une petite fonction pour le wait
#Pour l'appeler wait x ou wait est un entier
_wait() {
i=0
while [ $i -lt $1 ]; do
        i=`expr $i + 1`
done
}
```

```
while true; do
        echo "On lance ping"
        ./ping &
        _wait 100
        _pid enf1       #On passe en paramètre le nom du fichier
                        #dans lequel est stocké le PID
                        #On aurait pu utiliser la table des processus

                        #On lance pong et on recommance
        echo "On lance pong"
        ./pong &
        _wait 100
        _pid enf2
done


#Appel et gestion inter processus
#On va utiliser le n° de processus retourné par $$
#Ce n° de processus sera stocké dans un fichier
#enf1 pour le script enfant 1
#enf2 pour le script enfant 2


### Code du processus ping
#!/bin/bash
#script enfant 1 pour le TP sur les inter processus
#Lancer en utilisant ./ping sinon gare avec la commande réseau ping

echo $$ > enf1                  #on garde le pid
echo $$
echo "On est dans ping"
sleep 500                       #Pour garder en mémoire le processus


# pour le lancer en tâche de fond  ./ping &

### Code du processus pong
#!/bin/bash
#script enfant 2 pour le TP sur les inter processus
#Lancer en utilisant ./pong
echo $$ > enf2                  #on garde le pid
echo $$
echo "On est dans pong"
sleep 500                       #Pour garder le processus en mémoire


# pour le lancer en tâche de fond  taper ./pong &
```