Université d'Oran 1 Ahmed Ben Bella
Faculté des Sciences Exactes & Appliquées
Département d'Informatique

Module Systèmes d'Exploitation I (L2/S4)

Fiche TP n°5 **Scripts Shell** **Année 2015/2016**

Script avec un IF :

```
echo -n "Voulez-vous voir la liste des fichiers Y/N : "
read ouinon
if [ "$ouinon" = "y" ] || [ "$ouinon" = "Y" ]; then
   echo "Liste des fichiers :"
   ls -la
elif [ "$ouinon" = "n" ] || [ "$ouinon" = "N" ]; then
   echo "Ok, bye! "
else
   echo "Il faut taper Y ou N!! Pas $ouinon"
fi
```

Script avec un while.

```
cmpt=1
cm=3
echo -n "Mot de passe : "
read mdp

while [ "$mdp" != "ubuntu" ] && [ "$cmpt" != 4 ]
do
    echo -n "Mauvais mot de passe, plus que "$cm" chance(s): "
    read mdp
    cmpt=$(($cmpt+1))
    cm=$(($cm-1))
done
echo "Non mais, le brute-force est interdit en France !!"
exit 0
```

Script avec case :

```
echo -n "Etes-vous fatigué ? "
read on

case "$on" in
   oui | o | O | Oui | OUI ) echo "Allez faire du café !";;
   non | n | N | Non | NON ) echo "Programmez !";;
   * ) echo "Ah bon ?";;
esac
exit 0
```

Script complet :

```
clear
echo
echo "#################### Script ##############################"
echo
echo "##############################"
echo -n "LOGIN: "
read login
echo -n "Hôte: "
read hote
echo "##############################"
echo
echo "### Pour l'aide tapez help ###"
echo
while [ 1 ]; do                          # permet une boucle infinie
echo -n ""$login"@"$hote"$ "                # qui s'arrête avec break
read reps

case $reps in
  help | hlp )
    echo "A propos de TS --> about"
    echo "ls --> liste les fichiers"
    echo "rm --> détruit un fichier (guidé)"
    echo "rmd --> efface un dossier (guidé)"
    echo "noyau --> version du noyau Linux"
    echo "connect --> savoir qui s'est connecté dernièrement";;
  ls )
    ls -la;;
  rm )
    echo -n "Quel fichier voulez-vous effacer : "
    read eff
    rm -f $eff;;
  rmd | rmdir )
    echo -n "Quel répertoire voulez-vous effacer : "
    read eff
    rm -r $eff;;
  noyau | "uname -r" )
    uname -r;;
  connect )
    last;;
  about | --v | vers )
    echo "Script simple pour l'initiation aux scripts shell";;
  quit | "exit" )
    echo Au revoir!!
    break;;
  * )
   echo "Commande inconnue";;
esac
done
exit 0
```

Script for :

```
for var in 1 2 3 4 5 6 7 8 9; do
   echo $var
done
exit 0

tab=("John Smith" "Jane Doe")
len=${#tab[*]}
echo $ len
echo ${tab[1]}
echo ${tab[@]}
for (( i=0; i < ${#tab[@]}; i++ )); do echo ${tab[i]}; done
```

Fonctions :

```
# Je définis ma fonction effacer_fichier

effacer_fichier () {
    # J'affiche son nom et demande confirmation pour l'effacer
    echo "$1"
    echo "Voulez-vous vraiment l'effacer ? (o/n)"

    # Je lis la réponse de l'utilisateur
    read reponse

    # Et s'il dit oui, j'efface
    if [[ $reponse == "o" ]]
      then rm -f $1
    fi
}

# Je prends chaque fichier. aux du répertoire courant
for fichier in *.aux
  do
    # J'appelle la fonction effacer_fichier pour chaque fichier
    effacer_fichier $fichier
done

# Je prends chaque fichier .log du répertoire courant
for fichier in *.log
  do
    # J'appelle la fonction effacer_fichier pour chaque fichier
    effacer_fichier $fichier
done
```

```
# Je définis une première fonction

ecrire_sur_une_ligne () {
  echo -n $*
}

# Je définis une deuxième fonction qui appelle la première

saluer_utilisateur () {
  ecrire_sur_une_ligne "Bonjour "
  echo $USER
}


# J'appelle la deuxième fonction
saluer_utilisateur
```

Exercice : tri d'un vecteur.

```
declare nos[5]=(4 -1 2 66 10)

#
# Prints the number befor sorting
#
echo "Original Numbers in array:"
for (( i = 0; i <= 4; i++ ))
do
  echo ${nos[$i]}
done

#
# Now do the Sorting of numbers
#

for (( i = 0; i <= 4 ; i++ ))
do
   for (( j = $i; j <= 4; j++ ))
   do
      if [ ${nos[$i]} -gt ${nos[$j]}  ]; then
           t=${nos[$i]}
           nos[$i]=${nos[$j]}
           nos[$j]=$t
      fi
   done
done

#
# Print the sorted number
#
echo -e "\nSorted Numbers in Ascending Order:"
for (( i=0; i <= 4; i++ ))
do
  echo ${nos[$i]}
done
```

Q.1. How to write shell script that will add two nos, which are supplied as command line argument, and if this two nos are not given show error and its usage

Answer:

```
if [ $# -ne 2 ]
then
    echo "Usage - $0   x    y"
    echo "        Where x and y are two nos for which I will print sum"
    exit 1
fi
    echo "Sum of $1 and $2 is `expr $1 + $2`"
#
# ./ch.sh: vivek-tech.com to nixcraft.com referance converted using this
tool
# See the tool at http://www.nixcraft.com/uniqlinuxfeatures/tools/
#
```

Q.2.Write Script to find out biggest number from given three nos. Nos are supplies as command line argument. Print error if sufficient arguments are not supplied.
Answer:
```
    if [ $# -ne 3 ]
    then
        echo "$0: number1 number2 number3 are not given" >&2
        exit 1
    fi
    n1=$1
    n2=$2
    n3=$3
    if [ $n1 -gt $n2 ] && [ $n1 -gt $n3 ]
    then
        echo "$n1 is Bigest number"
    elif [ $n2 -gt $n1 ] && [ $n2 -gt $n3 ]
    then
        echo "$n2 is Bigest number"
    elif [ $n3 -gt $n1 ] && [ $n3 -gt $n2 ]
    then
        echo "$n3 is Bigest number"
    elif [ $1 -eq $2 ] && [ $1 -eq $3 ] && [ $2 -eq $3 ]
    then
        echo "All the three numbers are equal"
    else
        echo "I can not figure out which number is biger"
    fi
```

Q.3.Write script to print nos as 5,4,3,2,1 using while loop.
Answer:
```
i=5
while test $i != 0
do
        echo "$i
"
        i=`expr $i - 1`
done
```

Q.4. Write Script, using case statement to perform basic math operation as follows
+ addition
- subtraction
x multiplication
/ division
The name of script must be 'q4' which works as follows
$ ./q4 20 / 3, Also check for sufficient command line arguments
Answer:
```
if test $# = 3
then
        case $2 in
         +) let z=$1+$3;;
         -) let z=$1-$3;;
         /) let z=$1/$3;;
         x|X) let z=$1*$3;;
         *) echo Warning - $2 invalied operator, only +,-,x,/ operator
allowed
            exit;;
        esac
        echo Answer is $z
```

```
else
        echo "Usage - $0   value1  operator value2"
        echo "        Where, value1 and value2 are numeric values"
        echo "               operator can be +,-,/,x (For Multiplication)"
fi
```

Q.5.Write Script to see current date, time, username, and current directory
Answer:

```
echo "Hello, $LOGNAME"
echo "Current date is `date`"
echo "User is `who i am`"
echo "Current direcotry `pwd`"
```

Q.6.Write script to print given number in reverse order, for eg. If no is 123 it must print as 321.
Answer:

```
if [ $# -ne 1 ]
then
    echo "Usage: $0   number"
    echo "       I will find reverse of given number"
    echo "       For eg. $0 123, I will print 321"
    exit 1
fi

n=$1
rev=0
sd=0

while [ $n -gt 0 ]
do
    sd=`expr $n % 10`
    rev=`expr $rev \* 10  + $sd`
    n=`expr $n / 10`
done
    echo  "Reverse number is $rev"
```

Q.7.Write script to print given numbers sum of all digit, For eg. If no is 123 it's sum of all digit will be 1+2+3 = 6.
Answer:

```
if [ $# -ne 1 ]
then
    echo "Usage: $0   number"
    echo "       I will find sum of all digit for given number"
    echo "       For eg. $0 123, I will print 6 as sum of all digit
(1+2+3)"
    exit 1
fi

n=$1
sum=0
sd=0
while [ $n -gt 0 ]
do
    sd=`expr $n % 10`
    sum=`expr $sum + $sd`
    n=`expr $n / 10`
done
    echo  "Sum of digit for numner is $sum"
```

Q.8.How to perform real number (number with decimal point) calculation in Linux
Answer: Use Linux's bc command
Q.9.How to calculate 5.12 + 2.5 real number calculation at $ prompt in Shell ?
Answer: Use command as , $ echo 5.12 + 2.5 | bc , here we are giving echo commands output to bc to calculate the 5.12 + 2.5
Q.10.How to perform real number calculation in shell script and store result to third variable , lets say a=5.66, b=8.67, c=a+b?
Answer:

```
a=5.66
b=8.67
c=`echo $a + $b | bc`
echo "$a + $b = $c"
```

Q.11.Write script to determine whether given file exist or not, file name is supplied as command line argument, also check for sufficient number of command line argument
Answer:

```
if [ $# -ne 1 ]
then
    echo "Usage - $0  file-name"
    exit 1
fi

if [ -f $1 ]
then
    echo "$1 file exist"
else
    echo "Sorry, $1 file does not exist"
fi
```

Q.12.Write script to determine whether given command line argument ($1) contains "*" symbol or not, if $1 does not contains "*" symbol add it to $1, otherwise show message "Symbol is not required". For e.g. If we called this script Q12 then after giving ,
$ Q12 /bin
Here $1 is /bin, it should check whether "*" symbol is present or not if not it should print Required i.e. /bin/*, and if symbol present then Symbol is not required must be printed. Test your script as
$ Q12 /bin
$ Q12 /bin/*
Answer:

```
cat "$1" > /tmp/file.$$   2>/tmp/file0.$$

grep "*"  /tmp/file.$$    >/tmp/file0.$$

if [ $? -eq 1 ]
then
    echo "Required i.e. $1/*"
else
    echo "Symbol is Not required"
fi

rm -f /tmp/file.$$
rm -f /tmp/file0.$$
```

Q.13. Write script to print contains of file from given line number to next given number of lines. For e.g. If we called this script as Q13 and run as
$ Q13 5 5 myf , Here print contains of 'myf' file from line number 5 to next 5 line of that file.
Answer:

```
if [ $# -eq 0 ]
then
```

```
    echo "$0:Error command arguments missing!"
    echo "Usage: $0 start_line   uptoline   filename"
    echo "Where start_line is line number from which you would like to
print file"
    echo "uptoline is line number upto which would like to print"
    echo "For eg. $0 5 5 myfile"
    echo "Here from myfile total 5 lines printed starting from line no. 5
to"
    echo "line no 10."
    exit 1
fi

#
# Look for sufficent arg's
#

    if [ $# -eq 3 ]; then
        if [ -e $3 ]; then
            tail +$1 $3 | head -n$2
         else
            echo "$0: Error opening file $3"
            exit 2
        fi
    else
        echo "Missing arguments!"
    fi
```

Q.14. Write script to implement getopts statement, your script should understand following command line argument called this script Q14,
Q14 -c -d -m -e
Where options work as
-c clear the screen
-d show list of files in current working directory
-m start mc (midnight commander shell) , if installed
-e { editor } start this { editor } if installed
Answer:

```
#
# Function to clear the screen
#
cls()
{
    clear
    echo "Clear screen, press a key . . ."
    read
    return
}

#
# Function to show files in current directory
#
show_ls()
{
    ls
    echo "list files, press a key . . ."
    read
    return
}

#
```

```
# Function to start mc
#
start_mc()
{
    if which mc > /dev/null ; then
        mc
        echo "Midnight commander, Press a key . . ."
        read
    else
        echo "Error: Midnight commander not installed, Press a key . . ."
        read
    fi
    return
}

#
# Function to start editor
#
start_ed()
{
    ced=$1
    if which $ced > /dev/null ; then
        $ced
        echo "$ced, Press a key . . ."
        read
    else
        echo "Error: $ced is not installed or no such editor exist, Press a
key . . ."
        read
    fi
    return
}

#
# Function to print help
#
print_help_uu()
{
           echo "Usage: $0 -c -d -m -v {editor name}";
           echo "Where -c clear the screen";
           echo "      -d show dir";
           echo "      -m start midnight commander shell";
           echo "      -e {editor}, start {editor} of your choice";
           return
}

#
# Main procedure start here
#
# Check for sufficent args
#

if [ $# -eq 0 ] ; then
    print_help_uu
    exit 1
fi

#
# Now parse command line arguments
#
while getopts cdme: opt
```

```
do
    case "$opt" in
       c) cls;;
       d) show_ls;;
       m) start_mc;;
       e) thised="$OPTARG"; start_ed $thised ;;
       \?) print_help_uu; exit 1;;
    esac
done
```

Q.15. Write script called sayHello, put this script into your startup file called .bash_profile, the script should run as soon as you logon to system, and it print any one of the following message in infobox using dialog utility, if installed in your system, If dialog utility is not installed then use echo statement to print message : -
Good Morning
Good Afternoon
Good Evening , according to system time.
Answer:

```
temph=`date | cut -c12-13`
dat=`date +"%A %d in %B of %Y (%r)"`

if [ $temph -lt 12 ]
then
    mess="Good Morning $LOGNAME, Have nice day!"
fi

if [ $temph -gt 12 -a $temph -le 16 ]
then
    mess="Good Afternoon $LOGNAME"
fi

if [ $temph -gt 16 -a $temph -le 18 ]
then
    mess="Good Evening $LOGNAME"
fi

if which dialog > /dev/null
then
    dialog --backtitle "Linux Shell Script Tutorial"\
    --title "(-: Welcome to Linux :-)"\
    --infobox "\n$mess\nThis is $dat" 6 60
    echo -n "                           Press a key to continue. . .
"
    read
    clear
else
    echo -e "$mess\nThis is $dat"
fi
```

Q.16. How to write script, that will print, Message "Hello World" , in Bold and Blink effect, and in different colors like red, brown etc using echo command.
Answer:

```
# Syntax: echo -e "escape-code your message, var1, var2 etc"
# For eg. echo -e "\033[1m  Hello World"
#                    |         |
#                    |         |
#                Escape code   Message
#

clear
```

```
echo -e "\033[1m Hello World"
 # bold effect
echo -e "\033[5m Blink"
       # blink effect
echo -e "\033[0m Hello World"
 # back to noraml

echo -e "\033[31m Hello World"
 # Red color
echo -e "\033[32m Hello World"
 # Green color
echo -e "\033[33m Hello World"
 # See remaing on screen
echo -e "\033[34m Hello World"
echo -e "\033[35m Hello World"
echo -e "\033[36m Hello World"

echo -e -n "\033[0m "
  # back to noraml

echo -e "\033[41m Hello World"
echo -e "\033[42m Hello World"
echo -e "\033[43m Hello World"
echo -e "\033[44m Hello World"
echo -e "\033[45m Hello World"
echo -e "\033[46m Hello World"


echo -e "\033[0m Hello World"
  # back to noraml
```

Q.17. Write script to implement background process that will continually print current time in upper right corner of the screen , while user can do his/her normal job at $ prompt.
Answer:

```
# Q17
# To run type at $ promot as
# $ q17 &
#

echo
echo "Digital Clock for Linux"
echo "To stop this clock use command kill pid, see above for pid"
echo "Press a key to continue. . ."

while :
do
    ti=`date +"%r"`
    echo -e -n "\033[7s"    #save current screen postion & attributes
    # Show the clock
    tput cup 0 69          # row 0 and column 69 is used to show clock
    echo -n $ti            # put clock on screen
    echo -e -n "\033[8u"   #restore current screen postion & attributs
    #
    #Delay fro 1 second
    #
    sleep 1
done
```

Q.18. Write shell script to implement menus using dialog utility. Menu-items and action according to select menu-item is as follows:

Menu-Item	Purpose	Action for Menu-Item
Date/time	To see current date time	Date and time must be shown using infobox of dialog utility
Calendar	To see current calendar	Calendar must be shown using infobox of dialog utility
Delete	To delete selected file	First ask user name of directory where all files are present, if no name of directory given assumes current directory, then show all files only of that directory, Files must be shown on screen using menus of dialog utility, let the user select the file, then ask the confirmation to user whether he/she wants to delete selected file, if answer is yes then delete the file , report errors if any while deleting file to user.
Exit	To Exit this shell script	Exit/Stops the menu driven program i.e. this script

Note: Create function for all action for e.g. To show date/time on screen create function show_datetime().
Answer:

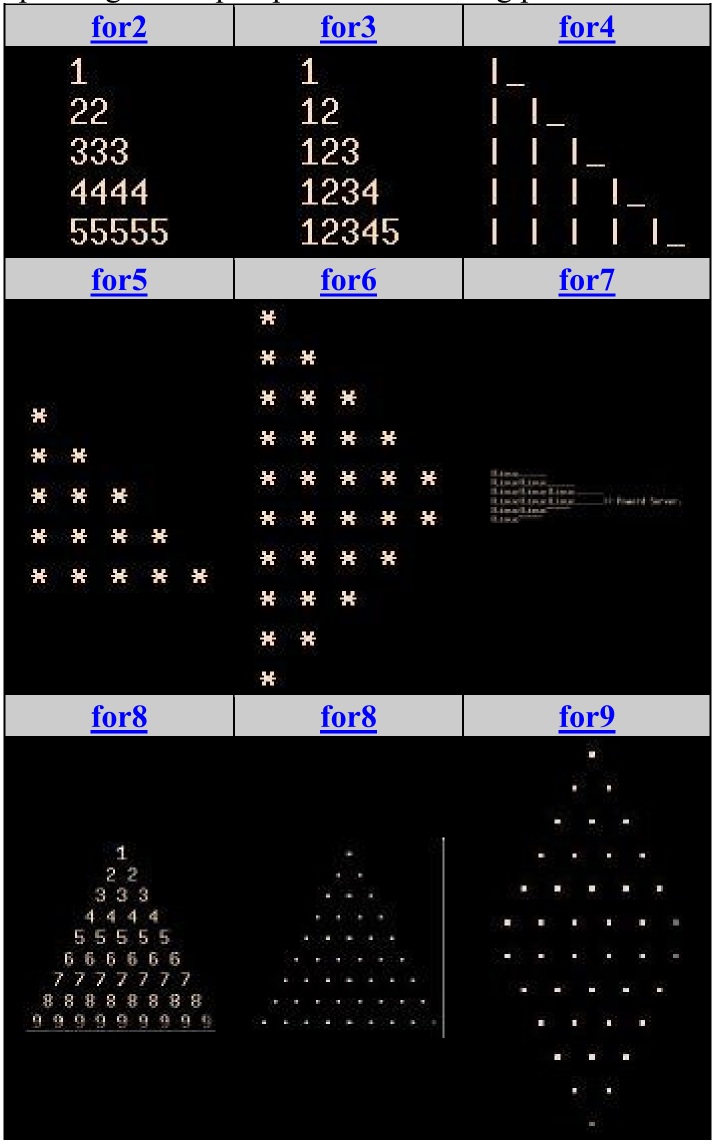
```
show_datetime()
{
   dialog --backtitle "Linux Shell Tutorial" --title "System date and Time"
--infobox "Date is `date`" 3 40
   read
   return
}

show_cal()
{
   cal > menuchoice.temp.$$
   dialog --backtitle "Linux Shell Tutorial" --title "Calender" --infobox
"`cat menuchoice.temp.$$`" 9 25
   read
   rm -f menuchoice.temp.$$
   return
}

delete_file()
{
 dialog --backtitle "Linux Shell Tutorial" --title "Delete file"\
 --inputbox "Enter directory path (Enter for Current Directory)"\
 10 40  2>/tmp/dirip.$$
 rtval=$?

 case $rtval in
     1) rm -f /tmp/dirip.$$ ; return ;;
     255) rm -f /tmp/dirip.$$ ; return ;;
 esac

 mfile=`cat /tmp/dirip.$$`

 if [ -z $mfile ]
 then
     mfile=`pwd`/*
 else
     grep "*" /tmp/dirip.$$
     if [ $? -eq 1 ]
     then
        mfile=$mfile/*
     fi
 fi

 for i in $mfile
 do
    if [ -f $i ]
    then
        echo "$i Delete?" >> /tmp/finallist.$$
    fi
 done

 dialog --backtitle "Linux Shell Tutorial" --title "Select File to Delete"\
 --menu "Use [Up][Down] to move, [Enter] to select file"\
 20 60 12 `cat /tmp/finallist.$$` 2>/tmp/file2delete.tmp.$$

 rtval=$?

 file2erase=`cat /tmp/file2delete.tmp.$$`

 case $rtval in
     0) dialog --backtitle "Linux Shell Tutorial" --title "Are you shur"\
      --yesno "\n\nDo you want to delete : $file2erase " 10 60

        if [ $? -eq 0 ] ; then
          rm -f  $file2erase
         if [ $? -eq 0 ] ; then
            dialog --backtitle "Linux Shell Tutorial"\
            --title "Information: Delete Command" --infobox "File:
$file2erase is Sucessfully deleted,Press a key" 5 60
            read
           else
            dialog --backtitle "Linux Shell Tutorial"\
            --title "Error: Delete Command" --infobox "Error deleting File:
$file2erase, Press a key" 5 60
            read
           fi
        else
          dialog --backtitle "Linux Shell Tutorial"\
          --title "Information: Delete Command" --infobox "File:
$file2erase is not deleted, Action is canceled, Press a key" 5 60
          read
        fi
     ;;
    1)  rm -f /tmp/dirip.$$ ; rm -f /tmp/finallist.$$ ;
        rm -f /tmp/file2delete.tmp.$$; return;;
    255) rm -f /tmp/dirip.$$ ;  rm -f /tmp/finallist.$$ ;
         rm -f /tmp/file2delete.tmp.$$; return;;
esac
 rm -f /tmp/dirip.$$
 rm -f /tmp/finallist.$$
 rm -f /tmp/file2delete.tmp.$$
 return
}



while true
do
dialog --clear --title "Main Menu" \
        --menu "To move [UP/DOWN] arrow keys \n\
[Enter] to Select\n\
        Choose the Service you like:" 20 51 4 \
        "Date/time"       "To see System Date & Time" \
        "Calender"        "To see Calaender"\
        "Delete"          "To remove file"\
        "Exit"            "To exit this Program" 2> menuchoice.temp.$$

retopt=$?

choice=`cat menuchoice.temp.$$`

rm -f menuchoice.temp.$$

case $retopt in
    0)
        case $choice in
            Date/time) show_datetime ;;
            Calender) show_cal ;;
            Delete) delete_file ;;
            Exit) exit 0;;
        esac
      ;;
     1) exit ;;
     255) exit ;;
 esac
done
clear
```

Q.19. Write shell script to show various system configuration like
1) Currently logged user and his logname
2) Your current shell
3) Your home directory
4) Your operating system type
5) Your current path setting
6) Your current working directory
7) Show Currently logged number of users
8) About your os and version ,release number , kernel version
9) Show all available shells
10) Show mouse settings
11) Show computer cpu information like processor type, speed etc
12) Show memory information
13) Show hard disk information like size of hard-disk, cache memory, model etc
14) File system (Mounted)
Answer:

```
nouser=`who | wc -l`
echo -e "User name: $USER (Login name: $LOGNAME)" >> /tmp/info.tmp.01.$$$
echo -e "Current Shell: $SHELL"  >> /tmp/info.tmp.01.$$$
echo -e "Home Directory: $HOME" >> /tmp/info.tmp.01.$$$
echo -e "Your O/s Type: $OSTYPE" >> /tmp/info.tmp.01.$$$
echo -e "PATH: $PATH" >> /tmp/info.tmp.01.$$$
echo -e "Current directory: `pwd`" >> /tmp/info.tmp.01.$$$
echo -e "Currently Logged: $nouser user(s)" >> /tmp/info.tmp.01.$$$

if [ -f /etc/redhat-release ]
then
    echo -e "OS: `cat /etc/redhat-release`" >> /tmp/info.tmp.01.$$$
fi
```

```
if [ -f /etc/shells ]
then
    echo -e "Available Shells: " >> /tmp/info.tmp.01.$$$
    echo -e "`cat /etc/shells`"  >> /tmp/info.tmp.01.$$$
fi

if [ -f /etc/sysconfig/mouse ]
then
    echo -e "-----------------------------------------------------------------
------" >> /tmp/info.tmp.01.$$$
    echo -e "Computer Mouse Information: " >> /tmp/info.tmp.01.$$$
    echo -e "-----------------------------------------------------------------
------" >> /tmp/info.tmp.01.$$$
    echo -e "`cat /etc/sysconfig/mouse`" >> /tmp/info.tmp.01.$$$
fi
echo -e "---------------------------------------------------------------------
--" >> /tmp/info.tmp.01.$$$
echo -e "Computer CPU Information:" >> /tmp/info.tmp.01.$$$
echo -e "---------------------------------------------------------------------
--" >> /tmp/info.tmp.01.$$$
cat /proc/cpuinfo >> /tmp/info.tmp.01.$$$

echo -e "---------------------------------------------------------------------
--" >> /tmp/info.tmp.01.$$$
echo -e "Computer Memory Information:" >> /tmp/info.tmp.01.$$$
echo -e "---------------------------------------------------------------------
--" >> /tmp/info.tmp.01.$$$
cat /proc/meminfo >> /tmp/info.tmp.01.$$$

if [ -d /proc/ide/hda ]
then
    echo -e "-----------------------------------------------------------------
------" >> /tmp/info.tmp.01.$$$
    echo -e "Hard disk information:" >> /tmp/info.tmp.01.$$$
    echo -e "-----------------------------------------------------------------
------" >> /tmp/info.tmp.01.$$$
    echo -e "Model: `cat /proc/ide/hda/model` " >> /tmp/info.tmp.01.$$$
    echo -e "Driver: `cat /proc/ide/hda/driver` " >> /tmp/info.tmp.01.$$$
    echo -e "Cache size: `cat /proc/ide/hda/cache` " >>
/tmp/info.tmp.01.$$$
fi
echo -e "---------------------------------------------------------------------
--" >> /tmp/info.tmp.01.$$$
echo -e "File System (Mount):" >> /tmp/info.tmp.01.$$$
echo -e "---------------------------------------------------------------------
--" >> /tmp/info.tmp.01.$$$
cat /proc/mounts >> /tmp/info.tmp.01.$$$

if which dialog > /dev/null
then
    dialog  --backtitle "Linux Software Diagnostics (LSD) Shell Script
Ver.1.0" --title "Press Up/Down Keys to move" --textbox
/tmp/info.tmp.01.$$$ 21 70
else
    cat /tmp/info.tmp.01.$$$ |more
fi

rm -f /tmp/info.tmp.01.$$$
```

Q.20.Write shell script using for loop to print the following patterns on screen



for2

```
1
22
333
4444
55555
```

for3

```
1
12
123
1234
12345
```

for4

```
|_
| |_
| | |_
| | | |_
| | | | |_
```

for5

```
*
* *
* * *
* * * *
* * * * *
```

for6

```
*
* *
* * *
* * * *
* * * * *
* * * * *
* * * *
* * *
* *
*
```

for7

for8

```
        1
       2 2
      3 3 3
     4 4 4 4
    5 5 5 5 5
   6 6 6 6 6 6
  7 7 7 7 7 7 7
 8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9
```

for8

```
        .
       . .
      . . .
     . . . .
    . . . . .
   . . . . . .
  . . . . . . .
 . . . . . . . .
. . . . . . . . .
```

for9

```
     .
    . .
   . . .
  . . . .
 . . . . .
. . . . . .
. . . . . .
 . . . . .
  . . . .
   . . .
    . .
     .
```

Answer:

```
echo "Can you see the following:"

for (( i=1; i<=5; i++ ))
do
    for (( j=1; j<=i;  j++ ))
    do
     echo -n "$i"
    done
    echo ""
done
********************************************
echo "Can you see the following:"

for (( i=1; i<=5; i++ ))
do
    for (( j=1; j<=i;  j++ ))
    do
     echo -n "$j"
    done
    echo ""
```

```
done
**************************************************
echo "Climb the steps of success"

for (( i=1; i<=5; i++ ))
do
    for (( j=1; j<=i;  j++ ))
    do
     echo -n " |"
    done
    echo "_ "
done
***********************************************
echo "Stars"

for (( i=1; i<=5; i++ ))
do
    for (( j=1; j<=i;  j++ ))
    do
     echo -n " *"
    done
    echo ""
done
*********************************************************
echo "Stars"

for (( i=1; i<=5; i++ ))
do
    for (( j=1; j<=i;  j++ ))
    do
     echo -n " *"
    done
    echo ""
done

for (( i=5; i>=1; i-- ))
do
    for (( j=1; j<=i;  j++ ))
    do
     echo -n " *"
    done
    echo ""
done
**************************************************************
clear

for (( i=1; i<=3; i++ ))
do
    for (( j=1; j<=i;  j++ ))
    do
     echo -n "|Linux"
    done
    echo "______"
done

for (( i=3; i>=1; i-- ))
do
    for (( j=1; j<=i;  j++ ))
    do
     echo -n "|Linux"
    done
    if [ $i -eq 3 ]; then
       echo -n "______"
       echo -n -e ">> Powerd Server.\n"
    else
       echo "~~~~~"
    fi
done
******************************************************************
MAX_NO=0

echo -n "Enter Number between (5 to 9) : "
read MAX_NO

if ! [ $MAX_NO -ge 5 -a $MAX_NO -le 9 ] ; then
   echo "I ask to enter number between 5 and 9, Okay"
   exit 1
fi

clear

for (( i=1; i<=MAX_NO; i++ ))
do
    for (( s=MAX_NO; s>=i; s-- ))
    do
       echo -n " "
    done
    for (( j=1; j<=i;  j++ ))
    do
     echo -n " $i"
    done
    echo ""
done

for (( i=1; i<=MAX_NO; i++ ))
do
    for (( s=MAX_NO; s>=i; s-- ))
    do
       echo -n " "
    done
    for (( j=1; j<=i;  j++ ))
    do
     echo -n " ."
    done
    echo ""
done

echo -e "\n\n\t\t\tI hope you like it my stupidity (?)"
```

```
MAX_NO=0

echo -n "Enter Number between (5 to 9) : "
read MAX_NO

if ! [ $MAX_NO -ge 5 -a $MAX_NO -le 9 ] ; then
   echo "I ask to enter number between 5 and 9, Okay"
   exit 1
fi

clear
```

```
for (( i=1; i<=MAX_NO; i++ ))
do
    for (( s=MAX_NO; s>=i; s-- ))
    do
       echo -n " "
    done
    for (( j=1; j<=i;  j++ ))
    do
     echo -n " $i"
    done
    echo ""
done

for (( i=1; i<=MAX_NO; i++ ))
do
    for (( s=MAX_NO; s>=i; s-- ))
    do
       echo -n " "
    done
    for (( j=1; j<=i;  j++ ))
    do
     echo -n " ."
    done
    echo ""
done

echo -e "\n\n\t\t\tI hope you like it my stupidity (?)"
*****************************************************************

MAX_NO=0

echo -n "Enter Number between (5 to 9) : "
read MAX_NO

if ! [ $MAX_NO -ge 5 -a $MAX_NO -le 9 ] ; then
   echo "I ask to enter number between 5 and 9, Okay"
   exit 1
fi

clear

for (( i=1; i<=MAX_NO; i++ ))
do
    for (( s=MAX_NO; s>=i; s-- ))
    do
       echo -n " "
    done
    for (( j=1; j<=i;  j++ ))
    do
     echo -n " ."
    done
    echo ""
done
###### Second stage ######################
##
##
for (( i=MAX_NO; i>=1; i-- ))
do
    for (( s=i; s<=MAX_NO; s++ ))
    do
       echo -n " "
    done
    for (( j=1; j<=i;  j++ ))
    do
     echo -n " ."
    done
    echo ""
done


echo -e "\n\n\t\t\tI hope you like it my stupidity (?)"
```
****

Q.21.Write shell script to convert file names from UPPERCASE to lowercase file names or vice versa.

Answer:

See :

```
BEGIN{
}

#
# main logic is here
#
{
    isdir1 = "[ -d " $1 " ] "
    isdir2 = "[ -d " $2 " ] "

    scriptname = "up2low"
    awkscriptname = "rename.awk"

    sfile = $1
    dfile = $2
    #
    # we are not suppose to rename dirs in source or destination
    #

    #
    # make sure we are renaming our self if in same dir
    #
    if ( sfile == scriptname || sfile == awkscriptname )
      next
    else if( ( system(isdir1) ) == 0 || system((isdir2)) == 0 )
    {
      printf "%s or %s is directory can't rename it to lower
case\n",sfile,dfile
      next # continue with next recored
    }
    else if ( sfile == dfile )
    {
       printf "Skiping, \"%s\" is alrady in lowercase\n",sfile
       next
    }
    else # everythink is okay rename it to lowercase
    {
     mvcmd = "mv " sfile " " dfile
     printf "Renaming %s to %s\n",sfile,dfile
     system(mvcmd)
    }
}

#
# End action, if any, e.g. clean ups
```

```
#
END{
}
******************************************
```

and

```
AWK_SCRIPT="rename.awk"

#
# change your location here
#
awkspath=$HOME/bin/$AWK_SCRIPT

ls -1 > /tmp/file1.$$

tr "[A-Z]" "[a-z]" < /tmp/file1.$$ > /tmp/file2.$$

paste /tmp/file1.$$ /tmp/file2.$$ > /tmp/tmpdb.$$

rm -f /tmp/file1.$$
rm -f /tmp/file2.$$

#
# Make sure awk script exist
#

if [ -f $awkspath ]; then
   awk -f $awkspath /tmp/tmpdb.$$
else
   echo -e "\n$0: Fatal error - $awkspath not found"
   echo -e "\nMake sure \$awkspath is set correctly in $0 script\n"
fi

rm -f /tmp/tmpdb.$$
```