OS with linux lab test 2022 - 2023

1. Write a script that takes exactly one argument, a directory name. If the number of arguments is more or less than one, print a usage message. If the argument is not a directory, print another message. For the given directory, print the five biggest files and the five files that were most re
cently modified.

Sol:

```
argument=$1 # $1 indicate first argument

# check at list one argument is given
if [ ! $# == 1 ]; then
  echo "Usage: You can send only one directory as an argument."
  exit
fi

# check the argument is an directory
if [ -d "${argument}" ]; then # it's a directory.
        echo "Five biggest files are listed below"
        du -ah "${argument}" | sort -rh | head -n 5
        echo "Five files that were most recently modified are listed below"
        ls -ltr | tail -n 5
else # it's not a directory.
        echo "Usage: your argument is not a directory."
fi
```

2. Write a script that does the following:
(a) Display the name of the script being executed.
(b) Display the first, third and tenth argument given to the script.
(c) Display the total number of arguments passed to the script.
(d) If there were more than three positional parameters, use shift to move all the values 3 places to the left.
(e) Print all the values of the remaining arguments.
(f) Print the number of arguments.
Test with zero, one, three and over ten arguments.

```
i = 4

echo “the script name is $0”

        echo “first argument is:  $1, third argument is : $3, tenth argument is : ${10}”
```

```
    echo “total number of arguments passed are : $#”
count=0
while (($# > 3)
do
echo “$1”
shift 3
echo “$1”
    done
    echo “there are $# parameters with $@ values”
    else
    echo “enter 3 parameters”
    exit
    fi
```

3. Write a shell script that locates all the hard links of the argument file from the home directory. The filename provided as argument must exist in the current directory.

For creating hARD lINKS

```
ln file hardlink (Outside vi editor)

For finding hardlinks of a specific file

find . -lname "filename" (Inside vi editor )
```

4. Write a shell script that takes an ordinary file as an argument and removes the file

if its size is zero. Otherwise, the script displays file's name, size, number of hard links, owner, and modify date. Your script must do appropriate error checking.

Sol:-

```
#!/bin/bash

if [ $# = 0 ]
 then
       echo "One Argument expected"
       echo "Usage: $0 ordinary_file"
       exit 1
elif [ $# != 1 ]
then
       echo "only one argument expected"
       echo "Usage : $0 ordinary_file"
      exit 1
fi

filename=$1

if [ -d $filename ]
then
echo "File is directory"
echo "Usage: $0 ordinary_file"
exit 1
fi

if [ ! -f $filename ]
then
echo "File is not exists "
echo "Usage: $0 ordinary_file"
exit 1
fi

set -- `ls -al $filename`


if [ $5 -eq 0 ]
then
echo "Its empty file. file deleted"
rm -f $filename
exit 0
fi
echo "File is not empty. File detail are: "
```

```
#file name
#size
#no.of hard links
#owner
#modify date
echo -e "File Name:                    $9"
echo -e "File size:              $5 bytes"
echo -e "Hard Links:             $2"
echo -e "file Owner:             $3"
echo -e "File modification time :     $6 $7 $8"
```

5. Write a script that would recognize if a word entered from the keyboard started with an upper or lower case character or a digit. The script would then output the word, followed by "upper case", "lower case", "digit", or "not upper, lower, or Digit".

Sol:

```
echo -e " enter a word or number : \ c " ; read ANS
case $ANS in
[a-z ]*) echo " lower case"
;;
 [A - Z] *) echo " upper case"
;;
 [ 0-9 ] *) echo " number."
;;
          *//default case
          esac //for closing
```

6. Write a shell script whose single command line argument is a file. If you run the program with an ordinary file, the program displays the owner's name and last update time for the file. If the program is run with more than one argument, it generates meaningful error messages.

Sol:

```
//If the number of arguments are1 and that argument is a ordinary file
 if [ "$#" -eq 1 -a -f "$1" ]
then
      ls -l $1 //dispalying details of file
else
      echo "$0: specify exactly one argument"
```

```
        echo "It must be an ordinary file"
        exit 1
fi
exit 0
```

7. Write a sed script/command, to perform the following task. Refer to the below mentioned database, db1, for performing the task.

Name of Friend	DOB	Hobby	Phone #
V.K. Rajopadhey 5/22,Stree 4, A'bad,MH, INDIA.	5/12/73	Food, Music	98220-5678
A.G. Gite 22, MIDC, Mumbai,MH, INDIA.	15/6/72	Computers, Book Reading	98220-3333
M.M. Kale 6/21,Silver Estate, A'bad,MH, INDIA.	2/1/71	Food, Drinks, Lifestyle	98220-6823
R.K. Joshi Flat No.9, Pushpa Towers, Pune,MH, INDIA.	9/10/70	Colletion of Old coins	98220-6877
N.K. Kulkarni Sector 20, Padmavti, Pune,MH, INDIA.	1/2/74	Computer Games	98220-9888

The task is as follows for db1 database file:

1) Find all occurrence of "A'bad" word replace it with "Aurangabad" word
2) Expand MH state value to Maharastra
3) Find all blank line and replace with actual line (i.e. ========)
4) Insert e-mail address of each persons at the end of persons postal address. For each person e-mail ID is different

Sol:

```
cat > tempfile

s/A.bad/Aurangabad/g //s id for replacement and g is for every occurrence repalcement

s/MH/Maharastra/g

s/^$/==========/g

//^$ used for blank lines
```

```
/V.K. /{
N
N                                       //N => next line
a\                                      //a => append
email:vk@fackmail.co.in
}

/M.M. /{
N
N
a\
email:mm@fackmail.co.in
}

/R.K. /{
N
N
a\
email:rk@fackmail.co.in
}

/A.G. / {
N
N
a\
```

```
email:ag@fackmail.co.in

}


/N.K. / {

N

N

a\

email:nk@fackmail.co.in

}

cat > tempfile

s/A.bad/Aurangabad/g

s/MH/Maharastra/g

s/^$/=========================================================================/g


/V.K. /{

N

N

a\

email:vk@fackmail.co.in

}


/M.M. /{

N

N

a\

email:mm@fackmail.co.in
```

```
}

/R.K. /{
N
N
a\
email:rk@fackmail.co.in
}

/A.G. / {
N
N
a\
email:ag@fackmail.co.in
}

/N.K. / {
N
N
a\
email:nk@fackmail.co.in
}
```

Run it as follows:

sed -f tempfile > updated_file // -f is for file

`cat updated_file`

*8. Write a shell script that accepts two directory names bar1 and bar2, and delete those files in bar2 whose contents are identical to their namesakes in bar1.

9. Write a script that reads a "password" from a user (it will not show when typed, but will get displayed after the carriage return is entered).

Sol:

```
echo  -n "Enter username: "
read username
echo  -n "Enter password: "
stty -echo
read  passwd
stty echo
echo "$username, the password entered is $passwd"
```

10. Write a shell script called whichdaemon.sh that checks if the httpd and init daemons are running on your system. If an httpd is running, the script should print a message like, "This machine is running a web server."

Sol:

```
echo "Check for init"
ps -e | grep 'init'   //ps -e  To see every process on the system
echo "check for httpd"  //httpd is a daemon process
ps -e | grep 'httpd'
if [ $? -eq 0 ]
//$? - checks for the last given command executed correctly or not if it is
//executed correctly then the output is 0 else 1
then
      echo “The machine is running a web server”
else
      echo “The Machine is not running web server”
fi
```

11. Write a script that simulates the ls –l command but prints only three columns of your choice.

ls -l | awk ‘{print $1 “ ” $3 “ ” $5 }’

12. Write a shell script that, given a filename as the argument, deletes all even lines (lines 2,4,6…n) in the file.

```
file=$1
counter=0

out="oddfile.$$" # odd file name

if [ $# -eq 0 ]
then
        echo "$(basename $0) file"
        exit 1
fi

if [ ! -f $file ]
then
        echo "$file not a file!"
        exit
fi

while read line
do
        # find out odd or even line number
        isEvenNo=$( expr $counter % 2 )

        if [ $isEvenNo -eq 0 ]
        then
                # odd match; copy all odd lines $out file
                echo $line >> $out
        fi
        # increase counter by 1
        (( $counter = $counter + 1 ))
done < $file
# remove input file
/bin/rm -f $file
```

13. Write a shell script that, given a filename as the argument, combines odd and even lines together. In other words, lines 1 and 2 become line 1, lines 3 and 4 become line 2 and so on.

commander@Terminal: ~/Documents/internal

File Edit Tabs Help

GNU nano 2.2.6 File: 13.sh

```
if [ ! $# == 1 ]
then
        echo "usage: $0 filename"
        exit 1
fi
file=$1
lines=$(cat $file | wc -l)
i=1
while (( $i <= $(($lines-1)) ))
do
        line1=$(awk 'NR=='$i'{print}' $file)
        line2=$(awk 'NR=='$i+1'{print}' $file)
        echo $line1 $line2 >> newfile
        i=$(($i+2))
done
cat > $file < newfile
rm newfile
```

[Read 17 lines]

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos

^X Exit ^J Justify ^W Where is

Ethernet network connection 'Wired connection 1' active

Output

commander@Terminal: ~/Documents/internal

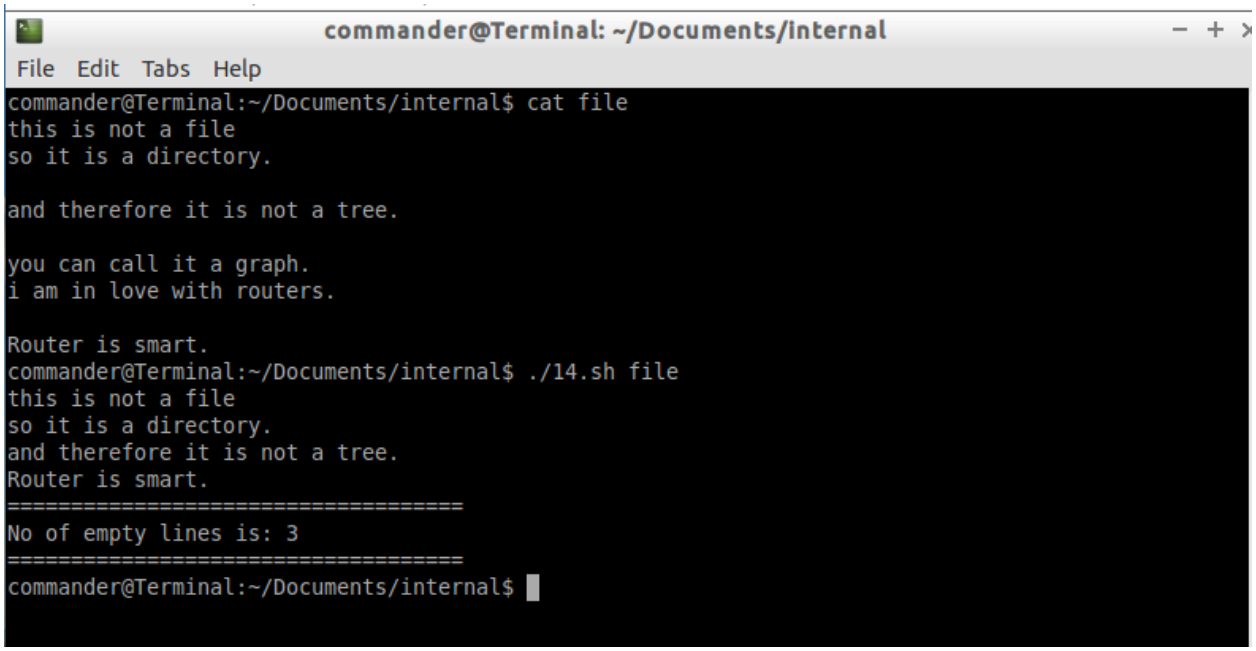
File Edit Tabs Help

```
commander@Terminal:~/Documents/internal$ ls
11.sh  13.sh  15.sh  1.sh  a1  a2  a3  lol  null
commander@Terminal:~/Documents/internal$ cat lol
line1
line2
line3
line4
line5
line6
line7
line8
commander@Terminal:~/Documents/internal$ ./13.sh
usage: ./13.sh filename
commander@Terminal:~/Documents/internal$ ./13.sh lol
commander@Terminal:~/Documents/internal$ cat lol
line1 line2
line3 line4
line5 line6
line7 line8
commander@Terminal:~/Documents/internal$
```

14. Write a shell script named count, which takes a file name as its parameter and prints number of blank lines in it. Also display the lines containing ‘is’ as a whole word in it (or the only word).

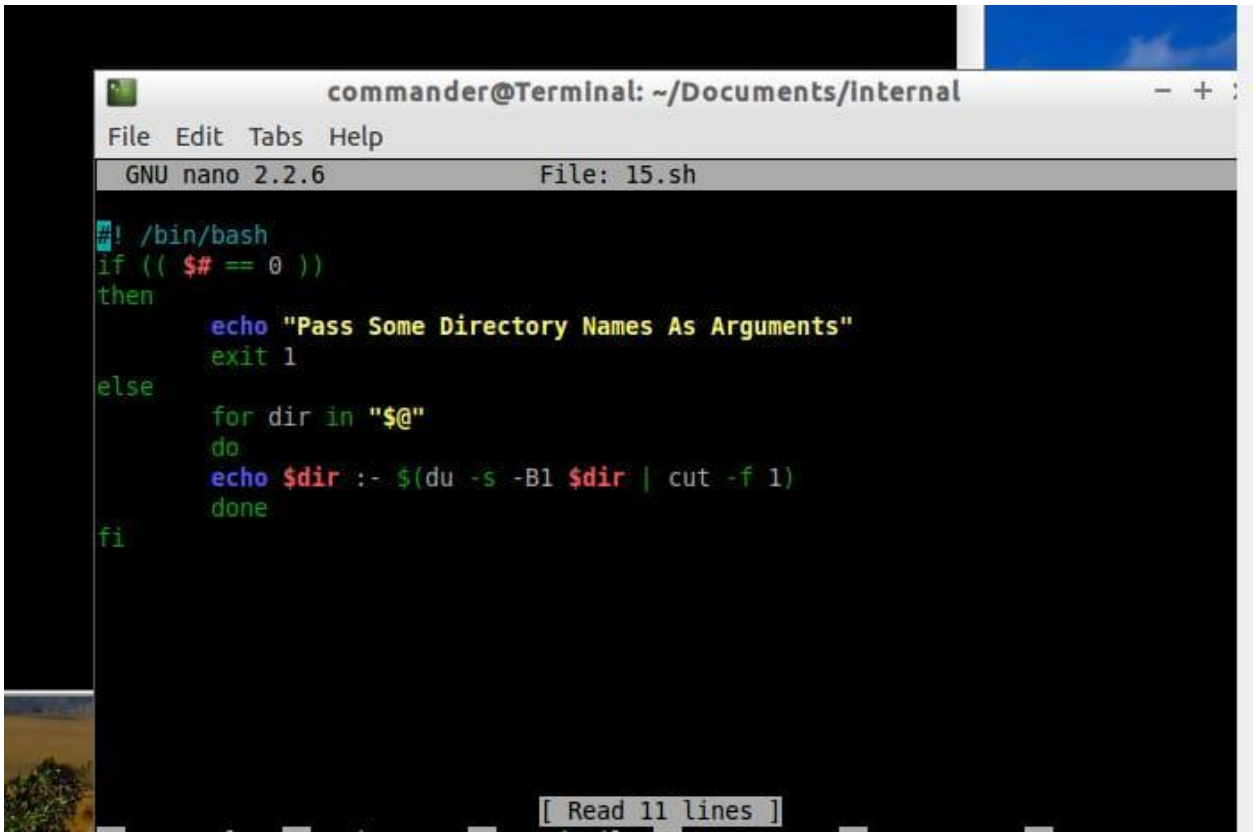
Code

Output

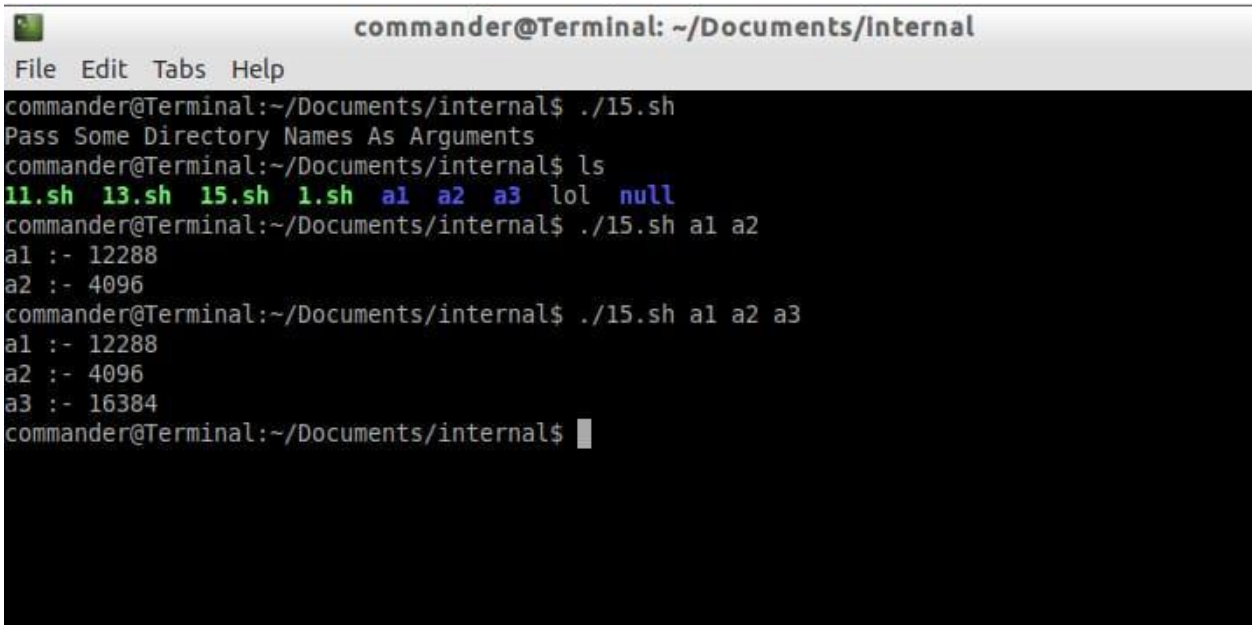


15. Write a shell script that takes a variable number of directory names as arguments. If no arguments are entered the script should abort with an appropriate error message.
For each directory in the given list, the script should display the filename and the size of the directory (including hidden files) in the following format:
<Directory Name>: - bytes occupied

SOL :



Output



*16. You are making a batch of beer. The beer has to stand in a warm place for 7 days and after that it has to stand in a cooler place for 2 weeks. You have a tendency to loose track of such mundane things so you want to write a small script that sends you a letter after 7 days telling you to move the beer and another letter 2 weeks after that that tells you that the beer is finished.

*17. Write a shell script to convert file names from UPPERCASE to lowercase file names or vice versa.

Sol:-
https://www.cyberciti.biz/faq/linux-unix-shell-programming-converting-lowercase-uppercase/

```
#!/bin/bash
# get filename
echo -n "Enter File Name : "
read fileName
If [$# eq 0]
  Echo “Give atleast one filename as argument”
  Exit


# make sure file exits for reading
for i in $@
do
if [  -f $i ]; then
  if [[ "$i" =~ [[:upper:]] ]]; then
```

```
mv $i  $i | tr [:upper:] [:lower:]
 fi

   ech linuxhint | tr [:lower:] [:upper:]

  exit 1
fi

# convert uppercase to lowercase using tr command
tr '[A-Z]' '[a-z]' < $fileName
```

Comments

To check with a regular expression (regex) if string contains at least one upper case character:

```
if [[ "$str" =~ [[:upper:]] ]]; then echo "uppercase character found" fi
```

To check with a regex if string contains at least one lower case character:

```
if [[ "$str" =~ [[:lower:]] ]]; then echo "lowercase character found" fi
```

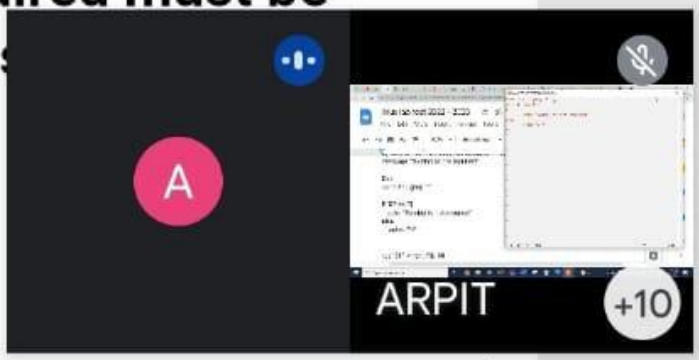
The `tr` command can be used in the following way to convert any string from uppercase to lowercase.

```
tr [:upper:] [:lower:]
```

18. 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".

Sol:

$ 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
$ Q12 /bin
$ Q12 /bin/*
Answer:




```
#
# Q12
# Script to check whether "/*" is included,
in $1 or not
#

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
```

19. Write a script to calculate factorial of a given number.

Algorithm

1. Get a number

2. Use for loop or while loop to compute the factorial by using the below formula

3. fact(n) = n * n-1 * n-2 * .. 1

4. Display the result.

Factorial of a number using while loop - Shell Script

```
#shell script for factorial of a number
#factorial using while loop
echo "Enter a number"
read num

fact=1

while [ $num -gt 1 ]
do
  fact=$((fact * num))  #fact = fact * num
  num=$((num - 1))        #num = num - 1
done
```

```
echo $fact
```

gt stands for greater than (>).

Output

```
Enter a number

3

6


Enter a number

4

24

Enter a number

5
```

120

Factorial of a number using for loop - Shell Script

```
#shell script for factorial of a number
#factorial using for loop

echo "Enter a number"
read num

fact=1

for((i=2;i<=num;i++))
{
  fact=$((fact * i))  #fact = fact * i
}

echo $fact
```

Output

```
Enter a number
```

3

6

Enter a number

4

2

Enter a number

5

120

(b) Write a script that will print message “Hello World” in bold and blink effect, and

in different colors like red, brown etc.

1-7 Text Effect
31-40//Colour Coding
Link
https://arkit.co.in/coloring-style-text-shell-scripting/

20. Write script to implement background process that will continually print current time (digital clock) in upper right corner of the screen, while user can do his/her normal job at $ prompt.

Link
http://www.freeos.com/guides/oldlsst/misc.htm

Sol:

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"` // %r for 12 Hr Format

echo -e -n "\033[7s" #save current screen postion & attributes // s for storing and u for restoring

#

 # 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 & attributes

#

#Delay for 1 second

#

sleep 1

Done

```
tput cup 0 0
     Send  the  sequence to move the cursor to row 0, column 0 (the
     upper left corner of the screen, usually known as  the  “home”
     cursor position).
```

*21. Write shell scripts which works similar to the following Linux commands:
head tail
Try to incorporate as many options as possible that are available with these Linux Commands.

Head options

Hhi

```
-c, --bytes=[-]NUM
       print the first NUM bytes of each file; with the leading '-', print all but the last NUM bytes of each file

-n, --lines=[-]NUM
       print the first NUM lines instead of the first 10; with the leading '-', print all but the last NUM lines of each file

-q, --quiet, --silent
       never print headers giving file names

-v, --verbose
       always print headers giving file names

-z, --zero-terminated
       line delimiter is NUL, not newline
```

Tail options

```
DESCRIPTION
      Print the last 10 lines of each FILE to standard output.  With more than one FILE, precede each with a header giving the file name.

      With no FILE, or when FILE is -, read standard input.

      Mandatory arguments to long options are mandatory for short options too.

      -c, --bytes=[+]NUM
             output the last NUM bytes; or use -c +NUM to output starting with byte NUM of each file

      -f, --follow[={name|descriptor}]
             output appended data as the file grows;

             an absent option argument means 'descriptor'

      -F     same as --follow=name --retry

      -n, --lines=[+]NUM
             output the last NUM lines, instead of the last 10; or use -n +NUM to output starting with line NUM

      --max-unchanged-stats=N
             with --follow=name, reopen a FILE which has not

             changed  size  after  N  (default  5)  iterations  to see if it has been unlinked or renamed (this is the usual case of rotated log
             files); with inotify, this option is rarely useful

      --pid=PID
             with -f, terminate after process ID, PID dies
```

```
   -q, --quiet, --silent
          never output headers giving file names

   --retry
          keep trying to open a file if it is inaccessible

   -s, --sleep-interval=N
          with -f, sleep for approximately N seconds (default 1.0) between iterations; with inotify and --pid=P, check  process  P  at  least
          once every N seconds

   -v, --verbose
          always output headers giving file names

   -z, --zero-terminated
          line delimiter is NUL, not newline
```

22. Write a script that asks the user to input a number and displays the squares of all numbers from 1 upto that number as follows:
1 Square=_____
2 Square=_____
:
n Square=_____
(n is the number entered by the user)

Sol:

```
echo "Enter a number"
read number
n=1
while [ $n -le $number ]
do
        echo " $n Square= `expr $n \* $n`"
        n=`expr $n + 1`
done
```

23,47. Write a shell script, which takes a argument either R, W or X and displays the names of all the ordinary files for which the user has read, write and execute permissions, respectively. If no arguments are entered, then the script should display the message:
Enter Either R, W or X:And accept the argument. If an invalid argument is given, the script should sound
the system bell, display the following message, and exit.
INVALID ARGUMENT! CANNOT EXECUTE! Follow the logic of Q 30

47.sh

File Edit Search Options Help

```
if [ $1 == r ]
then
for file in *
do
    if [ -r $file ]
    then
    echo $file
    fi
done
elif [ $1 == w ]
then
for file in *
do
    if [ -w $file ]
    then
    echo $file
    fi
done
elif [ $1 == x ]
then
for file in *
do
    if [ -x $file ]
    then
    echo $file
    fi
done
else
echo "INVALID ARGUMENT! CANNOT EXECUTE!"
```

Output

```
commander@Terminal:~/Documents/internal$ ./47.sh r
11.sh
13.sh
14.sh
15.sh
1.sh
28.sh
47and23Are470
47.sh
8.sh
a1
a2
a3
file
lol
null
commander@Terminal:~/Documents/internal$ ./47.sh x
11.sh
13.sh
14.sh
15.sh
1.sh
28.sh
47.sh
8.sh
a1
a2
a3
null
commander@Terminal:~/Documents/internal$
```

24 ,. While executing a shell script either the LOGNAME or the UID is supplied at the command prompt. Write a shell script to find out at how many terminals has this user logged in.

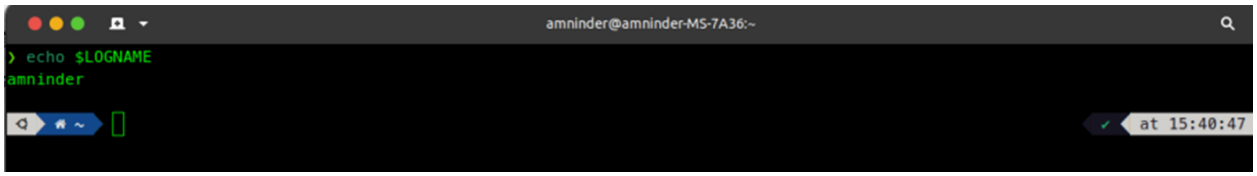
Sol:

Here we are going to see Find How Many Terminals Has User Logged In. Using who command to fetch the user list and then using grep command we can find the number of Terminal that the user has logged in the Linux. We can use LOGNAME or UID to identify the user. UID is a unique User ID assigned to every user that logged in the system, it is an integer value. The LOGNAME is the unique username of the user it can be alphanumeric.

We can use the following command to know the username of the current and its user ID:

For Username/LOGNAME

echo $LOGNAME



For UID(User ID):

id -u

Approach :

- **Taking input from Terminal**
- **Check if the input is UID or LOGNAME**
- **From the user list find all the numbers of Terminal that are opened via input UID.**
- **Then read the *passwd* file from *etc* directory that contains all the information about users.**

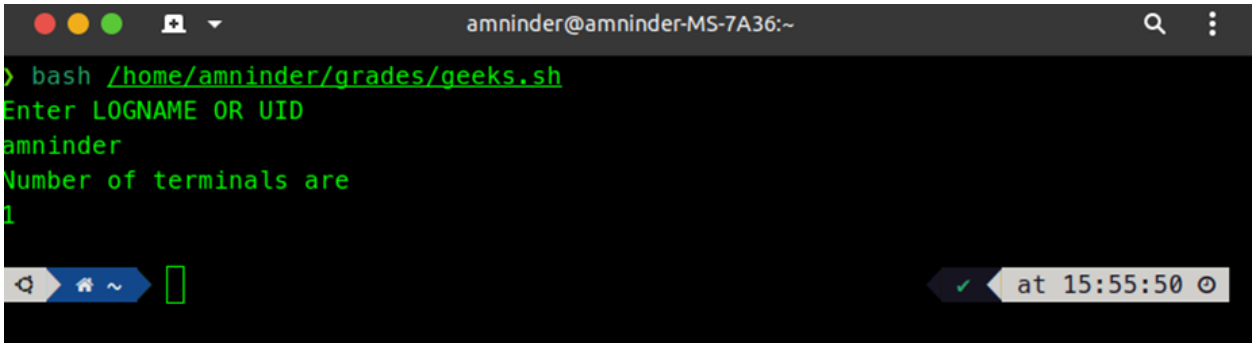
Below is the implementation:

#! /bin/bash

 # Taking input from user

echo "Enter LOGNAME OR UID"

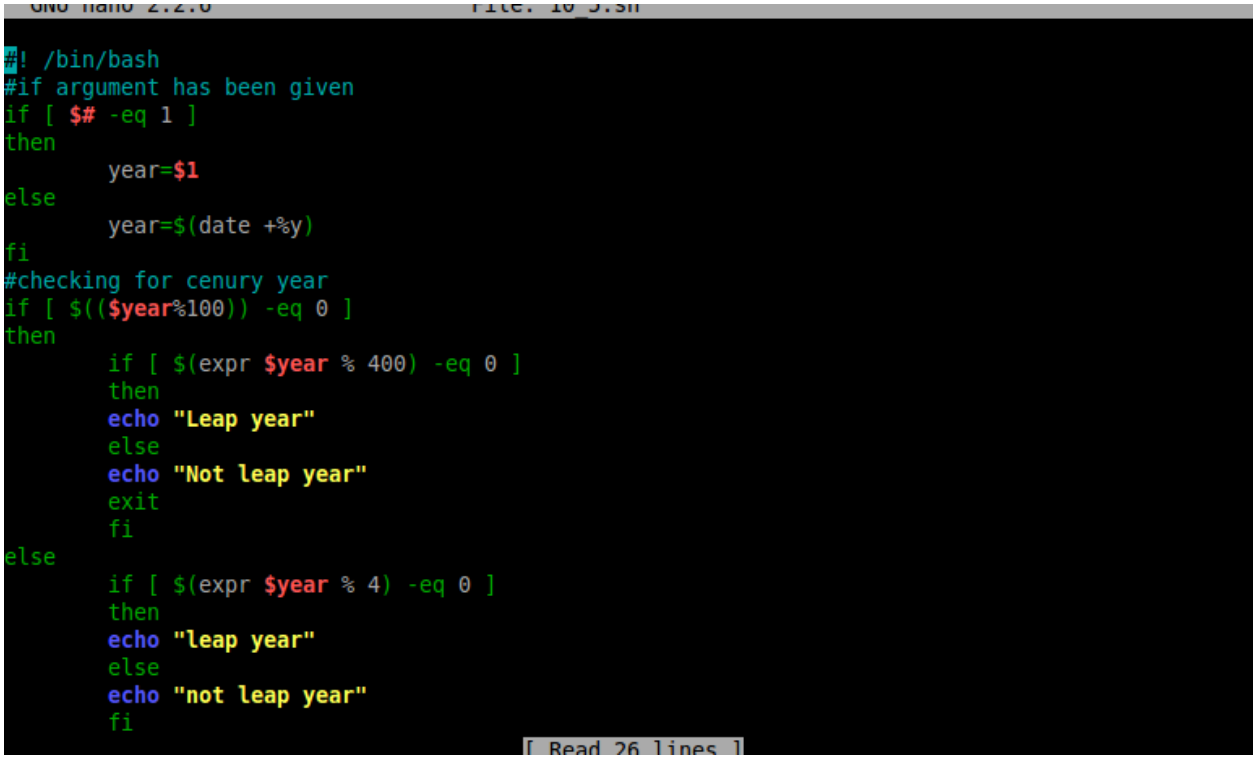
```
read input


# checking if input is a UID or LOGNAME
if [[ $input ]] && [ $input -eq $input 2>/dev/null ]


 # If input is UID
 then
        echo "Number of terminals are "
        cat /etc/passwd | grep $input -c


 # If input is LOGNAME
 else
        cat /etc/passwd>userlist
        echo "Number of terminals are "
        grep -c $input userlist
fi
```



25,51. Write a shell script for renaming each file in the directory such that it will have the current shell PID as an extension. The shell script should ensure that the directories do not get renamed.

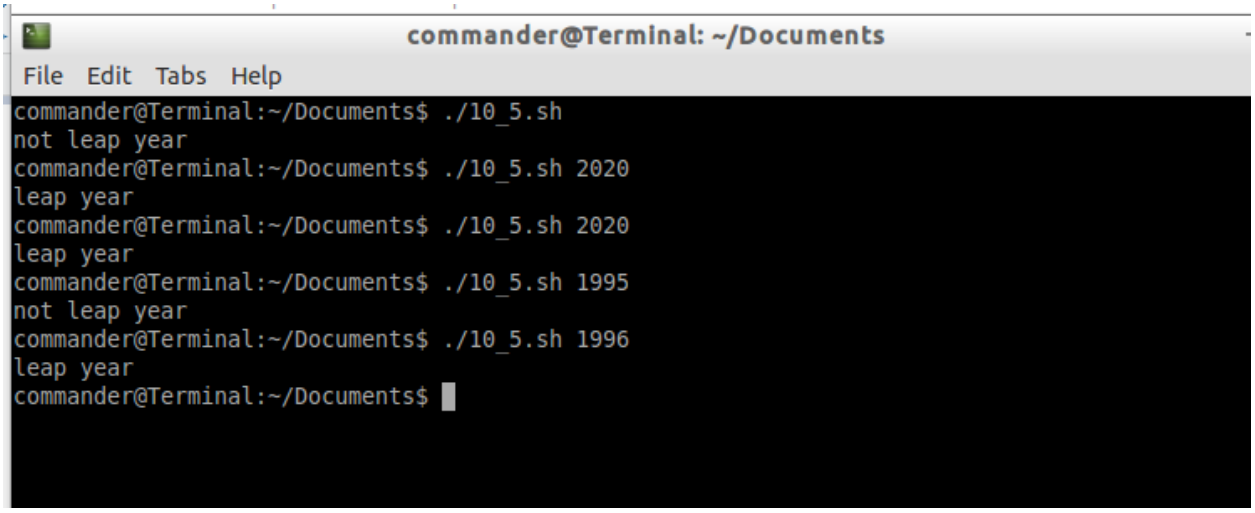
```
for i in `ls -a` //for all ordinary as well as  hidden files
do
   If [ -f $i ] //If argument is a file
   then
      mv $i $i.$$ // $$ for displaying the current shell pid
  fi
done
```

26. Write a shell script, which receives any year from the Keyboard, and determine whether the year is a leap year or not. If no argument is supplied the current year should be assumed.



```
#! /bin/bash
#if argument has been given
if [ $# -eq 1 ]
then
        year=$1
else
        year=$(date +%y)
fi
#checking for cenury year
if [ $(($year%100)) -eq 0 ]
then
        if [ $(expr $year % 400) -eq 0 ]
        then
        echo "Leap year"
        else
        echo "Not leap year"
        exit
        fi
else
        if [ $(expr $year % 4) -eq 0 ]
        then
        echo "leap year"
        else
        echo "not leap year"
        fi
```

[Read 26 lines]



Lab-12 question no -5
*27. Write a shell script, which will automatically get executed, on logging in. This shell script should display the present working directory and report whether your friend whose logname is aa10 has currently logged in or not. If he has logged in then the shell script should send a message to his terminal suggesting a dinner tonight. If you do not have write permission to his terminal or if he hasn't logged in then such a message should be mailed to him with a request to send confirmation about your dinner proposal.

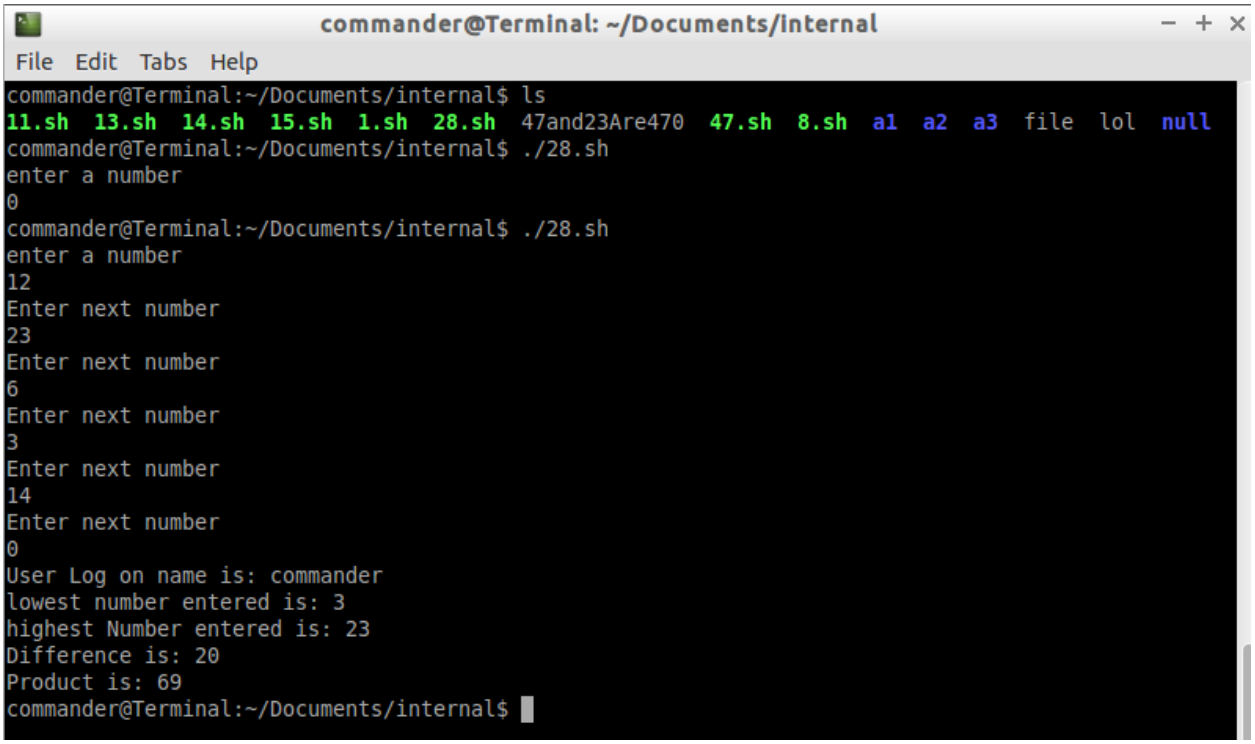
Sol:
Run chmod +x file_name.sh to make it executable, and place it in the /etc/profile.d/ directory to make it run once the user logs in.

/etc/profile file sets the environment variables at startup of the Bash shell. The /etc/profile.d directory contains other scripts that contain application-specific startup files, which are also executed at startup time by the shell.

Script: write this script with - sudo vi file_name.sh

28. Write a script asking the user to input some numbers. The script should stop asking for numbers when the number 0 is entered. The output should look like:
i. user: logon_name
ii. Lowest number entered:
iii. Highest number entered:
iv. Difference between the two:
v. Product of the two:


File Machine View Input Devices Help
28.sh
File Edit Search Options Help
echo "enter a number"
read current
if [$current -eq 0]
then
exit 1
else
low=$current
high=$current
while [$current -ne 0]
do
echo "Enter next number"
read next
if [$next -ne 0]
then
if (($next < $low))
then
low=$next
elif (($next > $high))
then
high=$next
fi
fi
current=$next
done
fi
echo "User Log on name is: $LOGNAME"
echo "lowest number entered is: $low"
echo "highest Number entered is: $high"
echo "Difference is: $(expr $high - $low)"
echo "Product is: $(expr $high * $low)"



29. A shell script receives even number of filenames. Suppose four filenames are supplied then the first file should get copied into second file, the third file should get copied into fourth file, and so on. If odd number of filenames are supplied then no copying should take place and an error message should be displayed.

```
if [ `expr $# % 2` -ne 0 ]
then
echo Enter even number of parameters
else
i=0
for k in $*
do
i=`expr $i + 1`
if [ $i -eq 1 ]
then
temp1=$k
fi
if [ $i -eq 2 ]
then
temp2=$k
i=0
cp $temp1 $temp2
fi
done
fi
```

30. Write a shell script, which displays a list of all files in the current directory to which you have read, write and execute permissions.
Sol:-

```
# Shell script to display list of file names
# having read, Write and Execute permission
echo "The name of all files having all permissions :"

# loop through all files in current directory
for file in *
do

# check if it is a file
if [ -f $file ]
then

# check if it has all permissions
if [ -r $file -a -w $file -a -x $file ] //-a is for AND -o is for OR ! is for NOT
then

# print the complete file name with -l option
ls -l $file

# closing second if statement
fi

# closing first if statement
fi

done
```

31. Write a script that would first verify if file "myfile" exists. If it does not, create it, then ask the user for confirmation to erase it…

```
SOL:  FILE=myfile
if [ -f "$FILE" ];
then
      echo "$FILE exists."
else
      touch $FILE
      echo "Press Y to delete file else N : "
```

```
        read option
        case "$option" in
                "Y")
                        rm $FILE
                        echo "File deleted"
                        ;;
                "N") exit 1
                        ;;
        esac
fi
```

32. Write a shell script, which receives two filenames as arguments. It should check whether the two file's contents are same or not. If they are same then second file should be deleted.

```
echo "Enter first file"
read file1
echo "Enter second file"
read file2
#fl1="`wc -c $file1 |cut -d '' -f 1`"
#f12="`wc -c $file2 | cut -d '' -f 1`"

if [ cmp -s -- "$file1" "$file2" ]
then
        echo "Both File Content are same"
        rm -r $file2
else
        echo "Difference"
fi
```

33. Write a shell script that adds an extension ".new" to all the files in a directory.

Sol:

//Refer to Q25

33. Write a shell script to print only those lines of file that do not contain word example and store the output in the file.

```
if [ $# -eq 0 ]
then
echo "$0: you must enter a file name"
exit 1
fi

echo `grep -v example $file > updateFile`
~
~
```

```
-v, --invert-match
       Invert the sense of matching, to select non-matching lines.
```

34. Write a shell script to print a number in reverse order. It should support the following requirements.
• The script should accept the input from the command line.
• If you don't input any data, then display an error message to execute the script Correctly.

Sol:

```
if [ $# -eq 0 ]  //$# checks for number of arguments
then
      echo "Please enter the digit in the argument"
      exit
fi
n=$1
sd=0
rev=0
while [ $n -gt 0 ]
do
         sd=$(( $n % 10 ))
         rev=`expr $rev \* 10 + $sd`
         n=$(( $n / 10 ))
done
echo "Reverse number of entered digit is $rev"
```

*35/6M. How will you delete a file which has special characters in its file name? Write a shell Script.

https://www.linux.com/training-tutorials/linux-shell-tip-remove-files-names-contains-spaces-and-special-characters-such/

```
$ find . -name '*[~*]*'
```

In the above command, replace dot(.) with the folder location where you want to look for the files. Replace ~* inside square brackets [] with the special characters present in your file's name.

If your file contains newline character (\n) or tab character (\t) then include them in your find command as shown.

```
$ find . -name '*[\t \n]*'
```

36. How can we perform numeric comparisons in Linux?

Syntax of comparisons in shell script

```
if [ conditions/comparisons]

      then

      commands

fi
```

An example

```
if [2 -gt 3]

      then

      print "2 is greater"
```

```
      else

      print "2 is not greater"

fi
```

This was just a simple example of numeric comparison & we can use more complex statement or conditions in our scripts. Now let's learn numeric comparisons in bit more detail.

Compare Numbers in Linux Shell Script

This is one the most common evaluation method i.e. comparing two or more numbers. We will now create a script for doing numeric comparison, but before we do that we need to know the parameters that are used to compare numerical values . Below mentioned is the list of parameters used for numeric comparisons

- **num1 -eq num2** check if 1st number is equal to 2nd number
- **num1 -ge num2** checks if 1st number is greater than or equal to 2nd number
- **num1 -gt num2** checks if 1st number is greater than 2nd number
- **num1 -le num2** checks if 1st number is less than or equal to 2nd number
- **num1 -lt num2** checks if 1st number is less than 2nd number
- **num1 -ne num2** checks if 1st number is not equal to 2nd number

Now that we know all the parameters that are used for numeric comparisons, let's use these in a script,

```
#!/bin/bash

# Script to do numeric comparisons

var1=10

var2=20
```

```
if [ $var2 -gt $var1 ]

    then

       echo "$var2 is greater than $var1"

fi

# Second comparison

If [ $var1 -gt 30]

    then

       echo "$var is greater than 30"

    else

       echo "$var1 is less than 30"

fi
```

This is the process to do numeric comparison

37 How will you find the sum of all numbers in a file in Linux?

Sol:

```
SUM=0
for num in $(cat num.txt)
   do
      ((SUM+=num))
done
echo $SUM
```

*38/9M. Write a shell script to delete the lines containing a word <dd> if it appears between the 5th and 7th position?

https://eduladder.com/viewquestions/20645/Write-A-Shell-Script-To-Delete-The-Lines-Containing-A-Word-Dd-If-It-Appears-Between-The-5th-And-7th-Position

*39/40. Write a shell script to find out the unique words in a file and also count the occurrence of each of these words. We can say that the file under consideration contains many lines, and each line has multiple words.
SOL:
sed -e 's/ /\n/g' filename | grep -v '^$' | sort | uniq -c | sort -n

https://eduladder.com/viewquestions/20646/Write-A-Shell-Script-To-Find-Out-The-Unique-Words-In-A-File-And-Also-Count-The-Occurrence-Of-Each-Of-These-Words-We-Can-Say-That-The-File-Under-Consideration-Contains-Many-Lines-And-Each-Line-Has-Multiple-Words

40. Write a shell script to get the total count of the word “Linux” in all the “.txt” files and also across files present in subdirectories.

Sol:
find . -name "*.txt" | xargs -o grep -i "Linux" | wc -l

The find command in UNIX is a command line utility for walking a file hierarchy. It can be used to find files and directories and perform subsequent operations on them. It supports searching by file, folder, name, creation date, modification date, owner and permissions. By using the ‘-exec’ other UNIX commands can be executed on files or folders found.

Syntax :

```
$ find [where to start searching from]
 [expression determines what to find] [-options] [what to find]
```

```
-o, --open-tty
      Reopen stdin as /dev/tty in the child process before executing the command.  This is useful if you want xargs to run an interactive
      application.
```

41. Write a shell script to validate password strength. Here are a few assumptions for the password string.
• Length – minimum of 8 characters.
• Contain both alphabet and number.
• Include both the small and capital case letters.If the password doesn't comply with any of the above conditions, then the script
should report it as a <Weak Password>.

echo "Enter Password"

```
read password
len="${#password}"

if test $len -ge 8 ; then
    echo "$password" | grep -q [0-9]
     if test $? -eq 0 ; then
           echo "$password" | grep -q [A-Z]
                if test $? -eq 0 ; then
                    echo "$password" | grep -q [a-z]
                      if test $? -eq 0 ; then
                       echo "Strong Password"
                   else
                       echo "Weak Password -> Should include a lower case letter."
                   fi
            else
               echo "Weak Password -> Should include a capital case letter."
            fi
     else
       echo "Weak Password -> Should use numbers in your password."
     fi
else
    echo "Weak Password -> Password length should have at least 8 characters."
fi
```

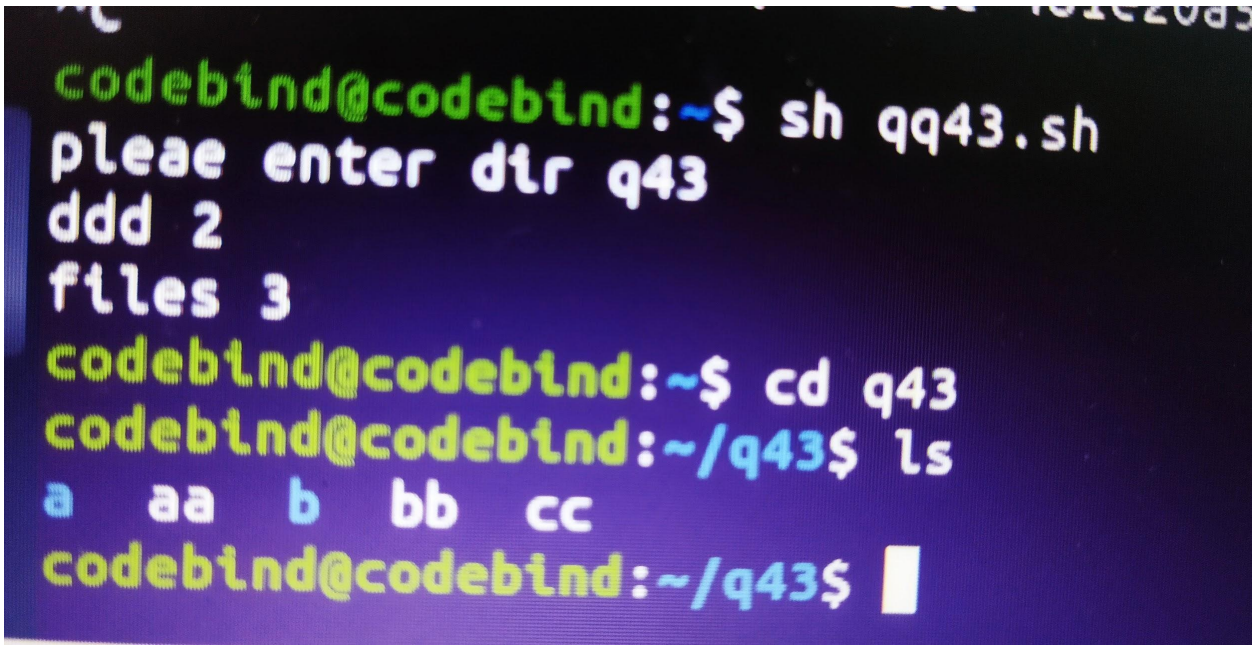
```
-o, --only-matching
       Print  only the matched (non-empty) parts of a matching line,
       with each such part on a separate output line.
```

42. Write a shell script to print the count of files and subdirectories in the specified directory.

Sol:-

```
//Give dir name you want to know the count of files and dir
read -p " Please enter the directory you want to count its files & subdirectories = " path

//for counting dir
echo "Number of directories in $path =  "$(find $path/* -type d | wc -l)

//for counting files
echo "Number of Files in $path = "$(find $path/* -type f | wc -l)
```

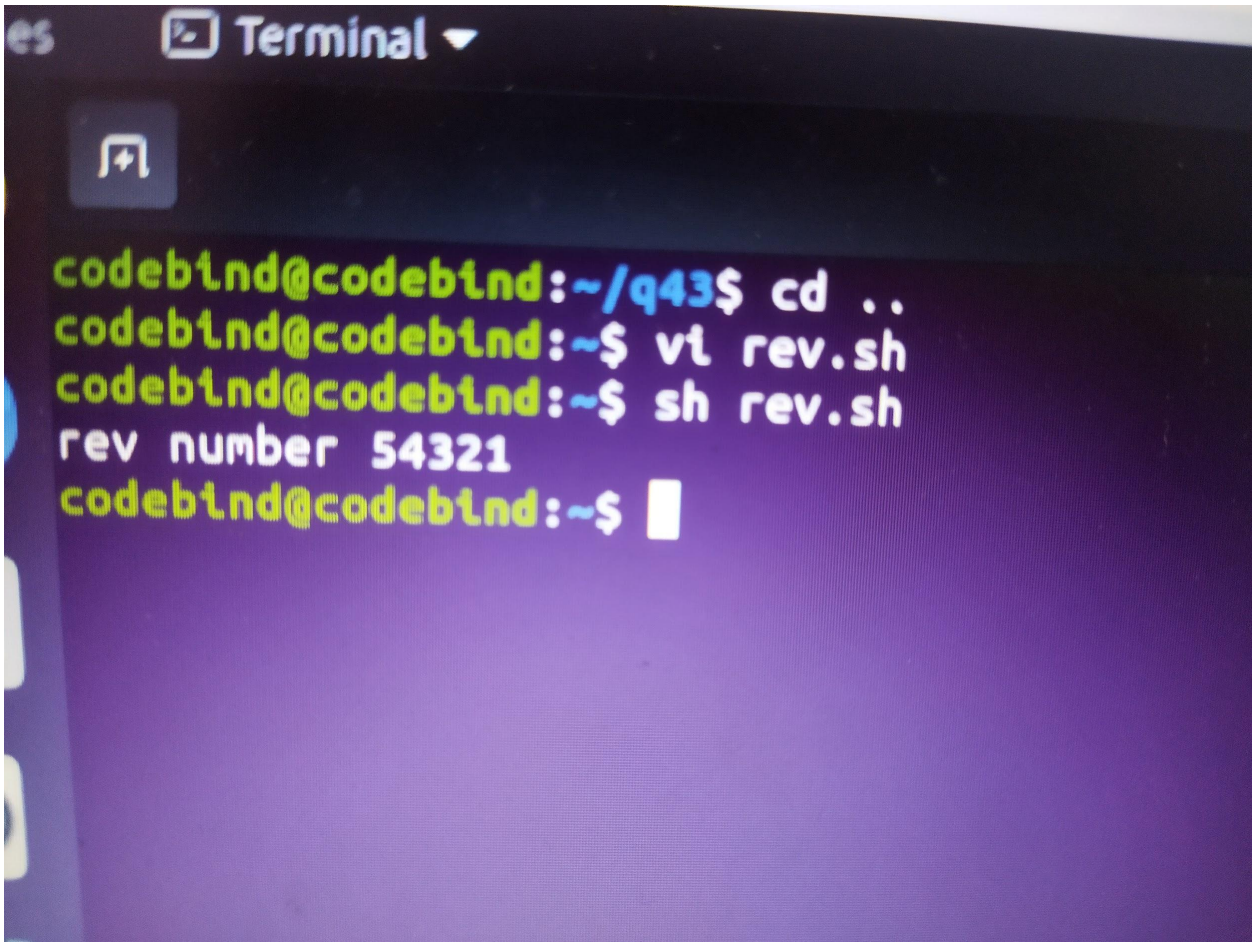
Output:-



43. Write a shell script to print the reverse of an input number.

```
n=123465
sd=0
rev=0
while [ $n -gt 0 ]
do
   sd=$(( $n % 10 ))
   rev=$(( $rev * 10 + $sd ))
   n=$(( $n / 10 ))
done
echo "Reverse number of entered digit is $rev"
```

Output:-



44. Write a shell script to reverse the list of strings and reverse each string further in the List.

```
#!/bin/bash

if [ $# != 0 ]; then
    len=`echo $@ | wc -c`
    len=`expr $len - 1`
    strrev=""
    while test $len -gt 0
    do
        strrev1=`echo $@ | cut -c$len`
        strrev=$strrev$strrev1
        len=`expr $len - 1`
    done
    echo $strrev
else
    echo "ERROR: Retry with a list of strings."
fi
```

45. Write a shell script to display the last updated file or the newest file in a directory?

```
echo "Last modified file-"
echo `ls -t | head -n 1`
~
```

```
-t      sort by modification time, newest first
```

46. Write a shell script, which takes a argument either R, W or X and displays the names of all the ordinary files for which the user has read, write and execute permissions, respectively. If no arguments are entered, then the script should display the message: Enter Either R, W or X:
And accept the argument. If an invalid argument is given, the script should sound the system bell, display the following message, and exit.

INVALID ARGUMENT! CANNOT EXECUTE!

Sol:

```
#!/bin/bash
# Shell script to find out whether file has read, write and execute
# permission
# -------------------------------------------------------------------
# Copyright (c) 2005 nixCraft project <http://cyberciti.biz/fb/>
# This script is licensed under GNU GPL version 2.0 or above
# -------------------------------------------------------------------
# This script is part of nixCraft shell script collection (NSSC)
# Visit http://bash.cyberciti.biz/ for more information.
# -------------------------------------------------------------------
```

47.sh

File Edit Search Options Help

```
if [ $1 == r ]
then
for file in *
do
    if [ -r $file ]
    then
    echo $file
    fi
done
elif [ $1 == w ]
then
for file in *
do
    if [ -w $file ]
    then
    echo $file
    fi
done
elif [ $1 == x ]
then
for file in *
do
    if [ -x $file ]
    then
    echo $file
    fi
done
else
echo "INVALID ARGUMENT! CANNOT EXECUTE!"
```

Output

```
commander@Terminal:~/Documents/internal$ ./47.sh r
11.sh
13.sh
14.sh
15.sh
1.sh
28.sh
47and23Are470
47.sh
8.sh
a1
a2
a3
file
lol
null
commander@Terminal:~/Documents/internal$ ./47.sh x
11.sh
13.sh
14.sh
15.sh
1.sh
28.sh
47.sh
8.sh
a1
a2
a3
null
commander@Terminal:~/Documents/internal$
```

Method 2

```
echo -n "Enter file name : "
read file

# find out if file has write permission or not
[ -w $file ] && W="Write = yes" || W="Write = No"

# find out if file has excute permission or not
[ -x $file ] && X="Execute = yes" || X="Execute = No"

# find out if file has read permission or not
[ -r $file ] && R="Read = yes" || R="Read = No"

echo "$file permissions"
echo "$W"
echo "$R"
echo "$X"
```

47. While executing a shell script either the LOGNAME or the UID is supplied at the command prompt. Write a shell script to find out at how many terminals has this user logged in.

Sol:

Here we are going to see Find How Many Terminals Has User Logged In. Using who command to fetch the user list and then using grep command we can find the number of Terminal that the user has logged in the Linux. We can use LOGNAME or UID to identify the user. UID is a unique User ID assigned to every user that logged in the system, it is an integer value. The LOGNAME is the unique username of the user it can be alphanumeric.

We can use the following command to know the username of the current and its user ID:

For Username/LOGNAME

echo $LOGNAME

For UID(User ID):

id -u

Approach :

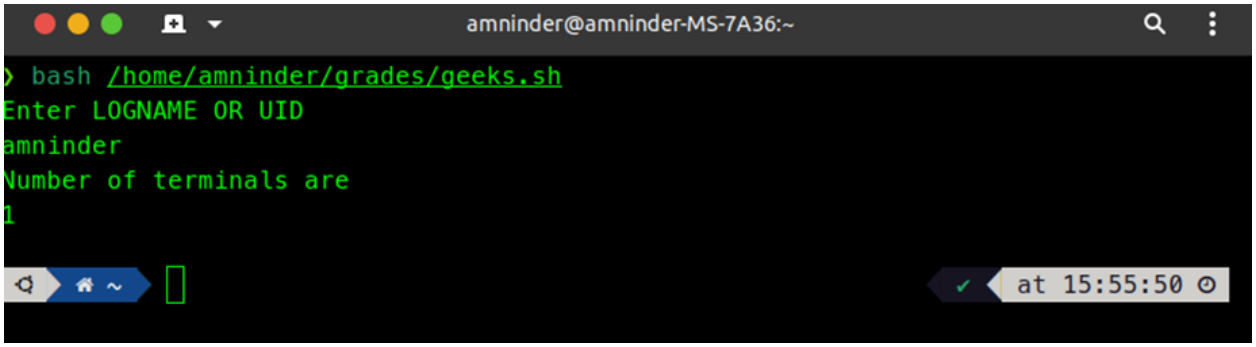
- **Taking input from Terminal**

- **Check if the input is UID or LOGNAME**
- **From the user list find all the numbers of Terminal that are opened via input UID.**
- **Then read the *passwd* file from *etc* directory that contains all the information about users.**

Below is the implementation:

```
#! /bin/bash


# Taking input from user

echo "Enter LOGNAME OR UID"

read input


# checking if input is a UID or LOGNAME

if [[ $input ]] && [ $input -eq $input 2>/dev/null ]


 # If input is UID

 then

        echo "Number of terminals are "

        cat /etc/passwd | grep $input -c


 # If input is LOGNAME

 else

        cat /etc/passwd>userlist

        echo "Number of terminals are "

        grep -c $input userlist
```

fi



48. Write a shell script for renaming each file in the directory such that it will have the current shell PID as an extension. The shell script should ensure that the directories do not get renamed.

```
for i in `ls -a` //for all ordinary as well as  hidden files
do
    If [ -f $i ] //If argument is a file
    then
       mv $i $i.$$ // $$ for displaying the current shell pid
   fi
done
```

49. Write a shell script to find out the unique words in a file and also count the occurrence of each of these words. We can say that the file under consideration contains many lines, and each line has multiple words.

Already done above

50. Write a shell script to get the total count of the word “Linux” in all the “.txt” files and also across files present in subdirectories.

Sol:
Already done

51. Write a shell script to validate password strength. Here are a few assumptions for the password string.

• Length – minimum of 8 characters.
• Contain both alphabet and number.• Include both the small and capital case letters.
If the password doesn't comply with any of the above conditions, then the script should report it as a <Weak Password>.

Sol;
Already done

52. Write a shell script to print the count of files and subdirectories in the specified Directory.
SOL: Already done

53. 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".

Sol:-
Already done

54. Write a shell script that, given a filename as the argument, combines odd and even lines together. In other words, lines 1 and 2 become line 1, lines 3 and 4 become line 2 and so on.
Sol **Already done**

55. Write a shell script named count, which takes a file name as its parameter and prints number of blank lines in it. Also display the lines containing 'is' as a whole word in it (or the only word).
Sol **Already done**

56. Write a shell script that takes a variable number of directory names as arguments. If no arguments are entered the script should abort with an appropriate error message.
For each directory in the given list, the script should display the filename and the size of the directory (including hidden files) in the following format:
<Directory Name>: - bytes occupied
Sol **Already done**

57. Write a shell script to validate password strength. Here are a few assumptions for the password string.
• Length – minimum of 8 characters.
• Contain both alphabet and number.
• Include both the small and capital case letters.

If the password doesn’t comply with any of the above conditions, then the script should report it as a <Weak Password>.

Sol **Already done**

58.. Write a shell script called whichdaemon.sh that checks if the httpd and init daemons are running on your system. If an httpd is running, the script should print a message like, "This machine is running a web server."

Sol **Already done**

59. Write a shell script that takes an ordinary file as an argument and removes the file if its size is zero. Otherwise, the script displays file’s name, size, number of hard links, owner, and modify date. Your script must do appropriate error checking. Sol **Already done**

60. Write a script that would recognize if a word entered from the keyboard started with an upper or lower case character or a digit. The script would then output the word, followed by "upper case", "lower case", "digit", or "not upper, lower, or digit".

Sol **Already done**

61. Write a shell script whose single command line argument is a file. If you run the program with an ordinary file, the program displays the owner’s name and last update time for the file. If the program is run with more than one argument, it generates meaningful error messages.

Sol **Already done**

19. Write a script to calculate factorial of a given number.

(b) Write a script that will print message “Hello World” in bold and blink effect, and

in different colors like red, brown etc.

Algorithm

1. Get a number

2. Use for loop or while loop to compute the factorial by using the below formula

3. fact(n) = n * n-1 * n-2 * .. 1

4. Display the result.

Factorial of a number using while loop - Shell Script

```
#shell script for factorial of a number
#factorial using while loop
echo "Enter a number"
read num

fact=1

while [ $num -gt 1 ]
do
  fact=$((fact * num))  #fact = fact * num
  num=$((num - 1))        #num = num - 1
done

echo $fact
```

gt stands for greater than (>).

Output

Enter a number

3

6

Enter a number

4

24

Enter a number

5

120

Factorial of a number using for loop - Shell Script

```
#shell script for factorial of a number

#factorial using for loop


echo "Enter a number"

read num


fact=1


for((i=2;i<=num;i++))

{

  fact=$((fact * i))  #fact = fact * i

}


echo $fact
```

Output

```
Enter a number


3


6
```

Enter a number

4

24

Enter a number

5

120

(b)#!/bin/bash

#

Linux Shell Scripting Tutorial 1.05r3, Summer-2002

#

Written by Vivek G. Gite <vivek@nixcraft.com>

#

Latest version can be found at http://www.nixcraft.com/

#

Q16

echo command with escape sequance to give differnt effects

#

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

#

# ./ch.sh: vivek-tech.com to nixcraft.com referance converted using this tool

# See the tool at http://www.nixcraft.com/uniqlinuxfeatures/tools/

#
```

32.

Write a shell script, which receives two filenames as arguments. It should check whether the two file's contents are same or not. If they are same then second file should be deleted.

$ bash lab8f.sh

Script:-

#Program to check whether the two file's content are same or not

echo "Enter first file name "

```
read file1

echo "Enter second file name "

read file2

fl1=` wc -c $file1 |cut -f 1 -d “ ”`

fl2=“` wc -c $file2 |cut -f 1 -d “ ”`

if [ $fl1 = $fl2 ]

then

echo "Both file's contents are same"

echo "Remove the second file "

else

echo " file's contents are not same"

fi
```

45.

. Write a program that will take an undetermined list of parameters, and reverse them.

$ bash lab12c.sh

Script:-

```
#Program that will take an undetermined list of paramters, and reverse them.


if [ $# -eq 0 ]
```

```
then
echo "There were no arguments on the command line!"
exit
fi
args=""
while [ ! $# -eq 0 ]
do
args="$1 $args"
shift
done
echo "The reversed arguments are: $args"
```

OR

```
// reverse a string using shell script
// reverse a string is in linux and unix

#!/ bin / bash
// reading a string
// using via user input
read - p "Enter string:" string
        // getting the length of given string
```

```
        len
        = $
{
        #string
}
// looping for reversing a string
// initialize i=len-1 for reversing a string and run till i=0
// printing in the reverse order of the given string
for ((i = $len - 1; i >= 0; i--))
        do
        // "${string:$i:1}"extract single single character from string.
        reverse = "$reverse${string:$i:1}" done
        echo "$reverse"
```

58. Write a shell script named count which takes a file name as its parameter and prints number of blank lines in it. Also display the lines containing “is” as a word.

$ bash lab10a.sh

Script:-

#Program which takes a file name as its parameter and prints number of blank lines in it.

```
if test $# -eq 0
then
echo "Filename not entered"
exit 1
fi
echo "count: "
grep -c "^$" $*
grep "is" $*
```

60. Write a shell script to validate password strength. Here are a few assumptions for the password string

Length – minimum of 8 characters.

Contain both alphabet and number

Include both the small and capital case letters.

if the password doesn't comply with any of the above conditions, then the script should report it as a

http://www.geekinterview.com/question_details/91266

```
1. echo "enter the password"
2. read password
3. len="${#password}"
```

```
4.
5. if test $len -ge 8 ; then
6.
7.     echo "$password" | grep -q [0-9]
8.      if test $? -eq 0 ; then
9.             echo "$password" | grep -q [A-Z]
10.                 if test $? -eq 0 ; then
11.                     echo "$password" | grep -q [a-z]
12.                       if test $? -eq 0 ; then
13.                         echo "Strong password"
14.                     else
15.                         echo "weak password include lower case char"
16.                     fi
17.             else
18.                echo "weak password include capital char"
19.             fi
```

```
20.     else
21.       echo "please include the numbers in password it is weak password"
22.     fi
23.else
24.     echo "password lenght should be greater than or equal 8 hence weak
   password"
25.fi
```