Scenario 1

#!/bin/bash //read the location of log directory, extension & backup directory from user

read -e -p "log directory:" log\_directory

read -e -p "file extension: " extension

read -e -p "backup directory: " backup\_directory

// creates a tar archive file with name “archive123.tar.gz”

tar czf archive123.tar.gz $(find $log\_directory -name "\*.$extension")

//move the tar file to the backup directory

mv archive123.tar.gz $backup\_directory

//delete the log file after moving to backup

rm $(find $log\_directory -name "\*.$extension")

exit 0 //exit the process with success

Scenario 2

#!/bin/bash  
read -p "archivefiles\_directory: " archive\_directory  
find $(realpath "$archive\_directory") -name \*.tar -mtime +2 -delete  
if !( crontab -l | grep -Fq "\* 0 \* \* \* /home/ubuntu/scenario2.sh" );  
then  
crontab -l | { cat;  echo "\* 0 \* \* \* /home/ubuntu/scenario2.sh";} | crontab -  
fi  
echo "script added to cron"

Scenario 6

**#!/bin/bash  
i=0  
b=0  
while [ $i -le 2 ]  
do  
read A  
result= $A  
a=`echo $?`  
echo $a  
if [ $a -eq $b ]; then  
echo "Command is Valid"  
else  
echo "Command is Invalid"  
fi  
((i++))  
done**

Scenario 7

#!/bin/bash

read -p "Enter device name: " dname

read -p "Enter threshold limit in bytes: " dlimit

echo "/dev/$dname"

if [ $( ls /dev/ | grep $dname ) == $dname ]

then

echo "$dname device exists"

if [ ! $( blockdev --getsize64 /dev/$dname ) -ge $dlimit ]

then

echo "$( date ) : $dname threshold limit $dlimit is greater than size " >> device.log

fi

if [ $( blockdev --getsize64 /dev/$dname ) -ge $dlimit ]

then

if [ $( df | grep $dname | wc -l ) -eq 1 ]

then

if [ $dlimit -ge $( df --output=avail /dev/$dname | tail -n +2 ) ]

then

echo "$( date ) : $dname threshold limit $dlimit reached " >> device.log

echo "$dname threshold reached!!" >> /etc/motd

fi

else

echo "device not mounted !"

fi

fi

else

echo "$dname device does not exist"

echo "$( date ) : $dname device does not exist " >> device.log

fi

Scenario 8

#!/bin/bash

// set the location of the recycle bin directory and the location of copy and remove tools

recycle\_bin="$HOME/.recycle\_bin"

rm="/bin/rm -r "

copy="/bin/cp -r "

//throw an error if there is no path provided

if [ $# -eq 0 ] ; then

echo "ERROR: Please enter the file path to delete." >&2; exit 1;

fi

flags=""

// parse all options looking for ‘-f’

while getopts "dfiPRrvW" args; do

case $args in

//if ‘-f’ is passed as argument delete the file/folder permanently and other arguments are passed to rm command.

f ) exec $rm "$@”

;;

\* ) flags="$flags -$args"

;;

esac

done

// shift “$((optiond-1))” removes all options that has been parsed by getopts from the parameter list.

shift $(( $OPTIND - 1 ))

// if recycle\_bin does not exist then make directory recycle\_bin

if [ ! -d $recycle\_bin ] ; then

mkdir $recycle\_bin

fi

for arg; do

//copy the file and folders and add date and time stamp to everydeleted file

newname="$recycle\_bin/$(date "+%S.%M.%H.%d.%m").$

// provide all other arguments and file paths to remove the utility

(basename "$arg")"

if [ -f "$arg" ] ; then

$copy "$arg" "$newname"

elif [ -d "$arg" ] ; then

$copy "$arg" "$newname"

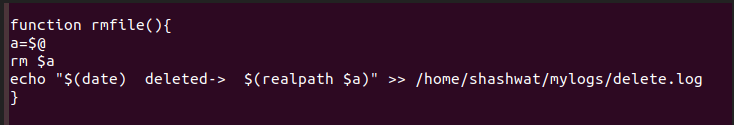
fi

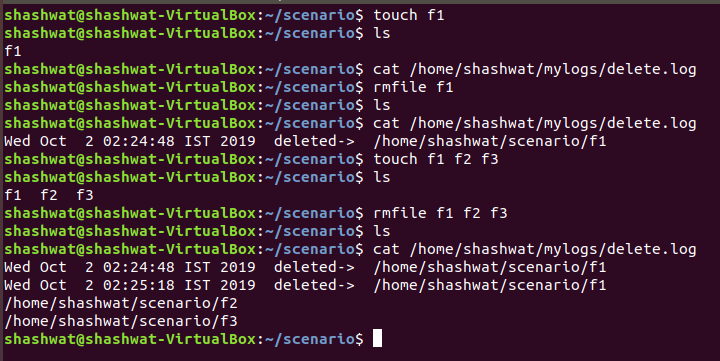
done

exec $rm $flags "$@"

exit 0

Scenario 10





Scenario 11

# taking bash as default bin

#!/bin/bash

# providing path for files to be traversed

FILES=/home/neha/myfile/DevAuto/\*

# start for loop

for f in $FILES

do

# take action on each file. $f store current file name

echo "Processing $f"

#wide spaced line

echo -en '\n'

# count of lines and the characters

wc $f

echo -en '\n'

#to get the owner of the file

#ls -l $f

stat -c "%U" $($f)

# counter to indicate the line numbers of the file

cat -n $f

echo -en '\n'

echo -en '\n'

echo -en '\n'

done

Scenario 12

#!/bin/bash

if [ -z "$1" ]; then #In this line we check that an argument is provided or not

echo "No file provided" >&2

exit 1

fi

read -p "Enter the password" pass #reading the password which would be used for encrypton

function func()

{

file\_name=$1

if [ -d `realpath $file\_name` ]; then #checking if the provided argument is a directory

cd `realpath $file\_name` #going to the location of the directory

array=(`ls | uniq `) #saving the output of the “ls | uniq” in a an array

len=${#array[\*]} #storing the length of the array in a variable

i=0

while [ $i -lt $len ]; do #looping through all the indices of the array

echo "${array[$i]}" #displaying the file in the index right now

func ${array[$i]} #recursively calling the function

let i++ #increasing the counter

done

fi

if [ -f `realpath $file\_name` ]; then #checking if the argument is a file

test= echo $pass | gpg -c --batch --yes --passphrase-fd 0 $file\_name #encrypting the file with the given password and storing the return value of the gpg in a variable

if [ "$test" == "1" ]; then #checking if the gpg executed properly or not

echo "Bad signature: gpg not executed properly" >&2 #writing to the standard error

exit 1

elif [ "$test" == "2" ]; then #checking if the gpg executed properly or not

echo "unexpected error: gpg not executed properly" >&2 #writing to the standard error

exit 1

else

rm $file\_name #deleting the original file

echo " $file\_name.gpg " #displaying the gpg created

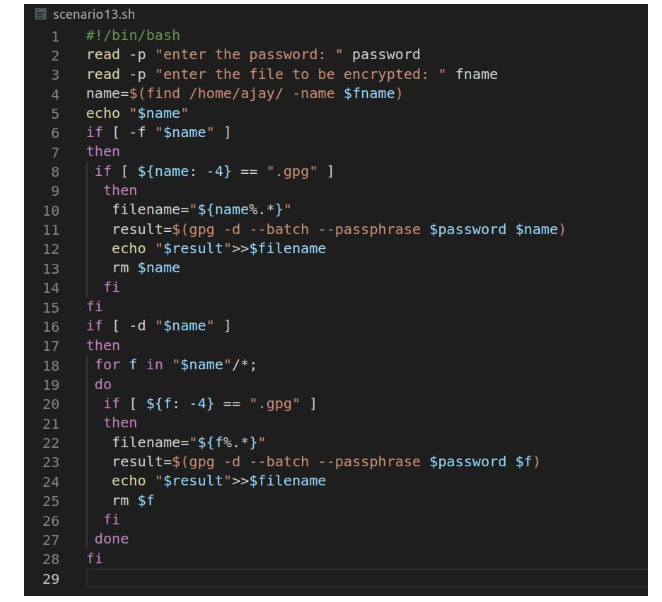
fi

fi

}

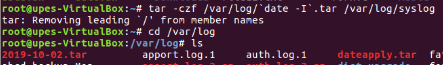
func $1

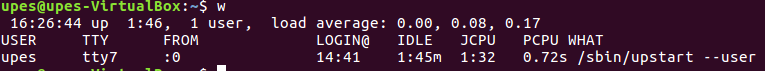
Scenario 13

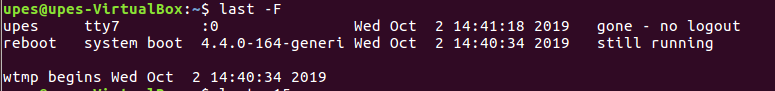


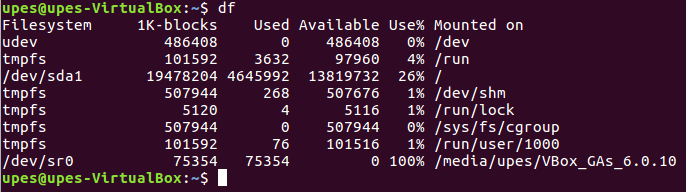
Scenario 14

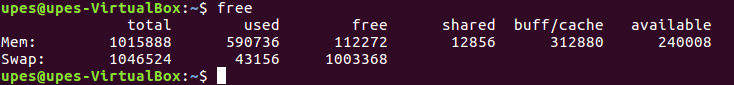
date.PNG

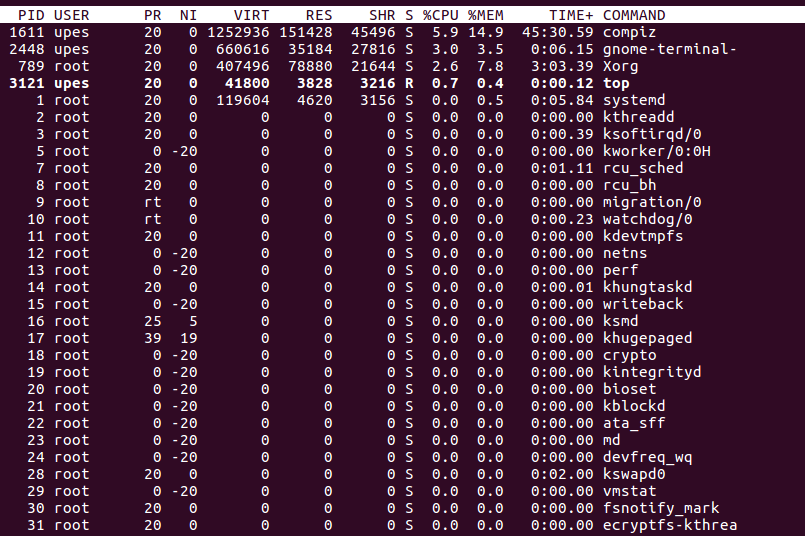


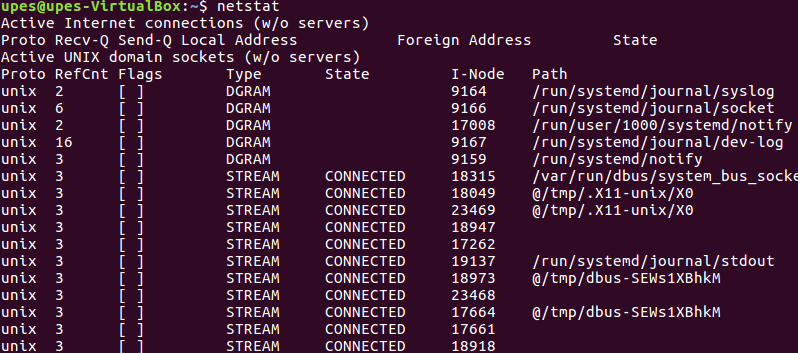


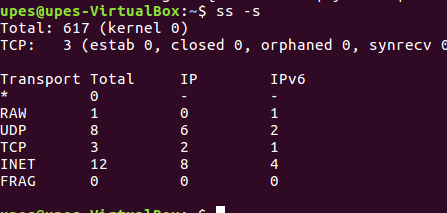












Scenario 15

