User Input

I was asked to write a script which takes input from user and run forensics-tools command on it according to file type.

First – the bash script checking user input's syntax, if there are any problems- user notified and asked to correct it.

Then – the script checks what target file's type user chooses.

If 'mem' – the script runs memory file's commands.

If 'hdd' – the script runs hdd file's command.

Code:

```
#If user did not enter target filetype / path, he will be asked to enter it
if [ -z $2 ]; then
        echo -e "\e[1;31m[!] You must specify file's type and filename!\e[1;0m"
        exit
fi

#to prevent problems with file's location- getting full path of file into variable
filename=$(readlink -f $2)

#If user enter invalid file's path, he will be notified and quitting the script
if !( test -f "$filename" ); then
        echo -e "\e[1;31m[!] The file not existed!\e[1;0m"
        exit
fi

#checking first argument- file type, and run appropiate commands. if neither the argument hdd or mem- user notified and quitting
if [ "$1" == mem ]; then
        foldername='mem' #defining output foldername
        VALID_INPUT  #executing the function 'VALID_INPUT' to make sure user enter valid input (filname, option etc.)
        vol_path=$(readlink -f $vol_path)
        INSTALL #executing the function 'Install' which checks (and install) required tools for script
        MEM  #executing the function 'MEM' which incloude all relevent commands to perform on target file
        LOG  #executing the function 'LOG' which will display general analysis to the user
elif [ "$1" == hdd ]; then
        foldername='hdd'  #defining output foldername
        VALID_INPUT #executing the function 'VALID_INPUT' to make sure user enter valid input (filname, option etc.)
        INSTALL #executing the function 'Install' which checks (and install) required tools for script
        HDD  #executing the function 'MEM' which incloude all relevent commands to perform on target file
        LOG  #executing the function 'LOG' which will display general analysis to the user
else
        echo -e "\e[1;31m[x] Please try again with valid option [mem/hdd]\e[1;0m"; exit
fi
```

Realtime script:

```
┌──(kali㉿kali)-[~/Desktop]
└─$ ./forensics_project.sh me memdump.mem
[x] Please try again with valid option [mem/hdd]

┌──(kali㉿kali)-[~/Desktop]
└─$ ./forensics_project.sh mem memdump.me
[!] The file not existed!

┌──(kali㉿kali)-[~/Desktop]
└─$ ./forensics_project.sh memdump.mem
[!] You must specify file's type and filename!

┌──(kali㉿kali)-[~/Desktop]
└─$ ./forensics_project.sh mem
[!] You must specify file's type and filename!
```

Functions

First function – **VALID_INPUT**

this function designed to make sure destination folder is not exist. if yes, user can delete it through the script.

(There is another part of this function which relevant only to memory files. I will add it below)

Code:

```
#If the directory name (mem/hdd) is already existed, user asked to delete it to prevent overwrite
if test -e "$foldername"; then
        echo -e "\e[1;33m[!] Directory\e[1;0m \e[33m'$foldername' \e[1;33mis already existed! Do you want to delete it? [enter 'yes' to delete | anything
        read del_choise
        if [ "$del_choise" == "yes" ]; then
                rm -rf "$foldername"
                echo -e "\e[1;33m[-] Folder\e[1;0m \e[33m'$foldername'\e[1;0m \e[1;33mdeleted!\e[1;0m"
                sleep 1
                clear
        elif [ "$del_choise" != "yes" ]; then
                exit
        fi
fi
}
```

Realtime-script:

```
┌──(kali㉿kali)-[~/Desktop]
└─$ ./forensics_project.sh hdd finder.dd
[!] Directory 'hdd' is already existed! Do you want to delete it? [enter 'yes' to delete | anything else to exit]:

┌──(kali㉿kali)-[~/Desktop]
└─$ ./forensics_project.sh hdd finder.dd
[!] Directory 'hdd' is already existed! Do you want to delete it? [enter 'yes' to delete | anything else to exit]:
yes
[-] Folder 'hdd' deleted!
```

The second (actually first) part of this function, designed to get volatility's binary file path.

If in entered path, the script can't find any file- he asked to correct it.

If he wants to download it through the script, he can do it by type 'install'.

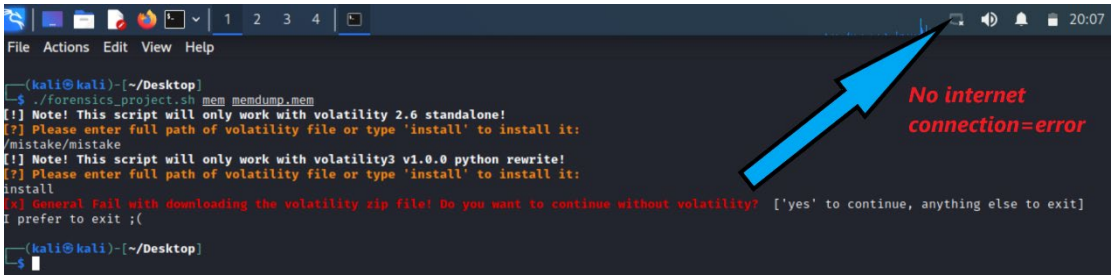
He also can continue without volatility.

Code:

```
if [ $totdirnum == "mem" ]; then
        #User asked to enter voltility's binary file path in order to use it in script
        echo -e "\e[1;37m[!] Note! This script will only work with volatility 2.6 standalone!\e[1;0m"
        echo -e "\e[1;93m[?] Please enter full path of volatility file or type 'install' to install it:\e[1;0m"
        read vol_path
        if [ "$vol_path" == 'install' ]; then
                wget http://downloads.volatilityfoundation.org/releases/2.6/volatility_2.6_lin64_standalone.zip 1>/dev/null 2>/dev/null
                wgetexitcode=$?
                if [[ $wgetexitcode -ne 0 ]]; then
                        echo -e "\e[1;31m[x] General Fail with downloading the volatility zip file! Do you want to continue without volatility? \e[1;0m[yes to cuntinue, anything else to exit]:"
                        read volafailchoise
                        if [ "$volafailchoise" == "yes" ]; then
                        echo -e "\e[1;33m[!] Continue without volatility\e[1;0m"
                        else
                                exit
                        fi
                fi
                unzip $vol_ver.zip
                rm $vol_ver.zip
                vol_path=./$vol_ver/$vol_ver
                clear
        else
                while !( test -e $vol_path ) || [ -z $vol_path ] || [ $vol_path == 'install' ]; do
                        echo -e "\e[1;37m[!] Note! This script will only work with volatility3 v1.0.0 python rewrite!\e[1;0m"
                        echo -e "\e[1;93m[?] Please enter full path of volatility file or type 'install' to install it:\e[1;0m"
                        read vol_path
                        if [ "$vol_path" == 'install' ]; then
                        wget http://downloads.volatilityfoundation.org/releases/2.6/volatility_2.6_lin64_standalone.zip 1>/dev/null 2>/dev/null
                        wgetexitcode=$?
                        if [[ $wgetexitcode -ne 0 ]]; then
                                echo -e "\e[1;31m[x] General Fail with downloading the volatility zip file! Do you want to continue without volatility? \e[1;0m ['yes' to continue, anything else to exit]"
                                read volafailchoise
                                if [ "$volafailchoise" == "yes" ]; then
                                echo -e "\e[1;33mContinue without volatility\e[1;0m"
                                else
                                        exit
                                fi
                        fi
                        unzip $vol_ver.zip
                        rm $vol_ver.zip
                        vol_path=./$vol_ver/$vol_ver
                        clear
                        break
                        fi
                done
        echo ""
        fi
volfullpath=$(readlink -f $vol_path)
fi
```

Realtime-script:

With error (from user/downloading volatility):



Without errors or problems with downloading:

```
┌──(kali㉿kali)-[~/Desktop]
└─$ ./forensics_project.sh mem memdump.mem
[!] Note! This script will only work with volatility 2.6 standalone!
[?] Please enter full path of volatility file or type 'install' to install it:
./volatility/volatility

[!] Directory 'mem' is already existed! Do you want to delete it? [enter 'yes' to delete | anything else to exit]:
^C

┌──(kali㉿kali)-[~/Desktop]
└─$ ./forensics_project.sh mem memdump.mem
[!] Note! This script will only work with volatility 2.6 standalone!
[?] Please enter full path of volatility file or type 'install' to install it:
install
^C
```

Second function – **'INSTALL'**

This function designed to make sure all necessary tools is installed on the machine.

If all tools installed, it is doing nothing.

If any tool is missing- user notified and can choose to install it.

Code:

```
#This function makeing sure all necessary tool for script is installed. if no- user can install it trough the script
function INSTALL(){

tools=("wget" "unzip" "bulk-extractor" "foremost" "binwalk" "pv")
for tool in ${tools[@]}; do
        apt list --installed 2>/dev/null | cut -d / -f1 | grep -q ^$tool$
        if [ $? -eq 1 ]; then
                echo -e "\e[1;33m[!] it looks like \e[1;34m$tool\e[1;33m is missing on your pc, do you want to install it? [Y/n]\e[1;0m"
                read installchoise
                        if [ "$installchoise" ≠ "n" ]; then
                                sudo apt-get install $tool 1>/dev/null 2>/dev/null
                                if [ $? -eq 0 ]; then
                                        echo -e "\e[1;34m[√] $tool\e[1;32m have been successfully installed on your pc\e[1;0m"
                                        echo ""
                                elif [ $? -ne 0 ]; then
                                        echo -e "\e[1;31[x] Can't install \e[1;33m$tool\e[1;33m! It may be internet connection error.\e[1;0m"
                                fi
                        else
                                echo -e "\e[1;33mOK :(\e[1;0m"
                                continue
                        fi
        else
                continue
        fi
done
clear
}
```

Realtime-script:

```
[!] it looks like foremost is missing on your pc, do you want to install it? [Y/n]

[√] foremost have been successfully installed on your pc

[!] it looks like pv is missing on your pc, do you want to install it? [Y/n]
n
```

Third function – **'MEM'/'HDD' (which includes the function 'COMMONCOMMANDS')**

The function **'COMMONCOMMANDS'**

This function designed to concentrate common command for both hdd and memory files.

It also creating relevant folder based on input ('mem'/'hdd')

It has two main advantages:

1. **Changes for both functions ('MEM'/'HDD') becoming easier**
2. **It saves code lines and preventing mistakes in code**

This function cares to show the user what actually being done in script and outputting the results of commands into relevant files.

<u>Code:</u>

```
#This function incloude common commands for both HDD and MEMORY files
#It made to save coding lines and make changes become more easier and comfort
function COMMONCOMMANDS(){
mkdir $foldername
cd $foldername

echo -e "\e[1;32m[+] Folder\e[1;0m \e[32m'$foldername'\e[1;0m \e[1;32mcreated!\e[1;0m"

echo -e "\e[1;32m[!] Processing the file\e[1;0m \e[32m$filename\e[1;0m"

echo -e "\e[1;34m       [!] Searching for readable\e[1;0m \e[34mSTRINGS\e[1;0m \e[1;34min target file \e[1;0m"
        strings $filename > strings_output
echo -e "\e[1;34m       [!]\e[1;0m \e[34mBINWALK\e[1;0m \e[1;34mon target file \e[1;0m"
        binwalk $filename -q -f ./binwalk_output
echo -e "\e[1;34m       [!]\e[1;0m \e[34mFOREMOST\e[1;0m \e[1;34mon target file \e[1;0m"
        foremost -Q $filename -o ./foremost_output 1,2>/dev/null
echo -e "\e[1;34m       [!]\e[1;0m \e[34mBULK_EXTRACTOR\e[1;0m \e[1;34mon target file \e[1;0m"
        bulk_extractor $filename -o bulk_output 1,2>/dev/null

}
```

<u>Realtime-script:</u>

In Realtime-script it simply integrated in the relevant function ('MEM'/'HDD').

Pictures will be added below, in function's pictures

The function '**MEM**'

This function designed to run commands on memory files. It includes the function 'COMMONCOMMANDS' (which relevant for both hdd and memory files), And volatility command includes the pre-selected plugins(in "volcommands" list).

Code:

```
#This function will be used in case user choised target file as memory dump
function MEM(){
COMMONCOMMANDS #Executing the function 'COMMONCOMMAND' which incloude relevent commands for both functions [MEM\HDD]
volcommands=("imageinfo" "pslist" "sockets" "connscan") #Here, user can change values to change the plugins will be used in volatility command
if [ "$volafailchoise" == "yes" ]; then
        echo -e "\e[1;33m   [-] Skipping Volatility (user choised) \e[1;0m"
else
        echo -e "\e[1;34m       [!]\e[1;0m \e[34mVolatility\e[1;0m \e[1;34mon target file \e[1;0m"
        for command in ${volcommands[@]}; do
                echo -e "\e[1m                --$command\e[1;0m"
                echo "                        $command:                                   " >> ./volatility_output
                echo "_____________________________________________________________________" >> ./volatility_output
                echo "" >> ./volatility_output
                $volfullpath $command -f $filename >> ./volatility_output 2>/dev/null
                echo "" >> ./volatility_output
                echo "" >> ./volatility_output
                echo "_____________________________________________________________________" >> ./volatility_output
                #if exitcode is error, user notified
                volexitcode=$?
                [ $volexitcode -ne 0 ] && echo -e "\e[1;31m[x] Volatility Fail! \e[1;93mIt may because you didn't entered full path of volatility or you
        done
fi
}
```

Realtime-script:

```
[+] Folder 'mem' created!
[!] Processing the file /home/kali/Desktop/memdump.mem
        [!] Searching for readable STRINGS in target file
        [!] BINWALK on target file
        [!] FOREMOST on target file
        [!] BULK_EXTRACTOR on target file
        [!] Volatility on target file
                --imageinfo
                --pslist
                --sockets
                --connscan
```

The function '**HDD**'

This function designed to run commands on memory files. It includes the function 'COMMONCOMMANDS' (which relevant for both hdd and memory files).

This function has not unique command except of common commands, but it can be edited and changed.

Code:

```
#This function will be used in case user choised target file as HDD dump
function HDD(){
COMMONCOMMANDS
}
```

Realtime-script:

```
[+] Folder 'hdd' created!
[!] Processing the file /home/kali/Desktop/finder.dd
        [!] Searching for readable STRINGS in target file
        [!] BINWALK on target file
        [!] FOREMOST on target file
        [!] BULK_EXTRACTOR on target file
```

Fourth function **'LOG'**

This function designed to display general analysis to the user.

It shows the user concise report about the file.

The user notified about destination folder's path and he can also open destination folder in GUI view.

<u>Code:</u>

```
function LOG(){
clear
for second in {5..1..-1}
do
        echo -e "\e[1;32m[✓] Action done! showing results in $second seconds \e[1;0m"
        sleep 1
        clear
done
echo -e "\e[1;35m $filename analysis: \e[1;0m"
echo "_____________________________________________________"
echo ""
echo -e "\e[1;36m./$foldername/strings_output: \e[1;0m"
        echo -e "\e[32m$(cat strings_output | wc -l) \e[1;0m readable strings found"
        printf "\n \n"
echo -e "\e[1;36m./$foldername/binwalk_output: \e[1;0m"
        echo -e "\e[32m$(cat binwalk_output | wc -l) files \e[1;0m found with binwalk"
        printf "\n \n"
echo -e "\e[1;36m./$foldername/foremost_output: \e[1;0m"
        echo "$(cat foremost_output/audit.txt | grep 'FILES EXTRACTED' -A 5000)"
        printf "\n \n"
echo -e "\e[1;36m./$foldername/bulk_output: \e[1;0m"
        echo -e "\e[32mFiles can be found in the folder\e[1;0m"
        printf "\n \n"
if [ "$foldername" == "mem" ]; then
        if [ -z $volexitcode ] || [ $volexitcode -ne 0 ]; then
                echo -e "\e[1;36m./$foldername/volatility_output: \e[1;0m"
                echo -e "\e[33mNo output from Volatility (SKIPPED) \e[1;0m"
        elif  [[ $volexitcode -eq 0 ]]; then
                echo -e "\e[1;36m./$foldername/volatility_output: \e[1;0m"
                cat volatility_output
        fi
fi
printf "_____________________________________________________________ \n \n"
apt list --installed 2>/dev/null | cut -d / -f1 | grep -q ^pv$
if [ $? -eq 0 ]; then
        echo -e "\e[1;32m[!] You can find all files in\e[1;0m \e[32m$(pwd)\e[1;0m" | pv -qL 15
else
        echo -e "\e[1;32m[!] You can find all files in\e[1;0m \e[32m$(pwd)\e[1;0m"
fi
printf "_____________________________________________________________ \n \n"
echo -e "\e[1;32mDo you want to open directory with FileExplorer? ['yes' to open | anything else to finish]: \e[1;0m"
read openchoise
if [ "$openchoise" == "yes" ]; then
        xdg-open ../$foldername
        if [ $? -ne 0 ]; then
                echo -e "\e[1;31mError! Can't open $(pwd) in GUI. 'xdg-open' may not installed\e[1;0m"
        fi
fi
printf "_____________________________________________________________\n"
}
```

Realtime-script:

MEM(cropped screenshot) :

```
/home/kali/Desktop/memdump.mem analysis:
_____________________________________________

./mem/strings_output:
657154  readable strings found


./mem/binwalk_output:
10 files  found with binwalk


./mem/foremost_output:
185 FILES EXTRACTED

jpg:= 1
gif:= 4
bmp:= 5
wmv:= 1
mov:= 1
rif:= 4
htm:= 1
exe:= 168
_____________________________________________

Foremost finished at Wed Aug 10 21:41:57 2022


./mem/bulk_output:
Files can be found in the folder


./mem/volatility_output:
                              imageinfo:
_____________________________________________

          Suggested Profile(s) : WinXPSP2x86, WinXPSP3x86 (Instantiated with WinXPSP2x86)
```

HDD: