Liquid Galaxy Bootcamp

- Submitted by Aju Tamang
- Tribhuwan University, Bsc.CSIT (Freshman)

Abstract

I presented all of my Liquid Galaxy Bootcamp tasks of Bash, Linux, Git, and networking problems. You will be knowns mostly basic Linux commands and automating repeated tasks using Bash. You will also learn about different networking protocols and performing few tasks through automation.

Table of contents

MS1 UNIX Basic commands

Shutdown

Make a script to shutdown the machine, but before the action, the user must be asked for confirmation and only perform it if the user input is yes. Feel free to change messages and outputs.

```
bash shutdown.sh

Are you sure that you want to shutdown the computer? y
Shutting down...
```

Solution

```
#!/bin/bash
echo "Are you sure that you want to shutdown the computer ? [Y,n]"
read input
if [[ $input == "Y" || $input == "y" ]]; then
        sudo poweroff
        echo "Shutting down..."
elif [[ $input == "N" || $input == "n" ]]; then
        echo "Your computer is not shutdown yet"
else
        echo "Please answer with y or n"
fi
```

Using out the root access and poweroff command, it will shutdown your linux system once you input the command as Y.

Reboot

Make a script to perform a reboot command on the machine, before the action a count down of 5 seconds should be prompted. Feel free to change messages and outputs.

```
bash restart.sh

The computer will reboot...
5
4
3
2
1
Rebooting...
```

Solution

```
#!/bin/bash
#START
echo "The computer will reboot..."
for i in {5..1}
do
    echo $i
done
echo "Rebooting..."
sleep 5; reboot
```

reboot will reboot the linux system and sleep will set out the time interval for rebooting the linux system.

Turn screen

Make a script to turn the screen to left or right, the direction must be passed by arguments. Feel free to change messages and outputs. Input example:

```
bash turn-screen.sh left
```

Solution

```
#!/bin/bash
for var in "$@"
do
    if [[ "$var" == "left" || "$var" == "l" ]]; then
        xrandr -o left
    elif [[ "$var" == "right" || "$var" == "r" ]]; then
        xrandr -o right
    else
         xrandr -o normal
    fi
done
```

The best way to turn your screen left , right or normal by using xrandr and you can set out left, right and normal one where I used it in if condition statement.

Change keyboard language

Make a script to change the keyboard language of the keyboard, the language must be passed via argument. Usage example:

```
bash keyboard-language.sh us
```

Solution

```
#!/bin/bash
sudo locale-gen && cat /etc/locale.gen | grep "$1" > /etc/locale.conf
```

Menu

Now it's time to make an interface to let the user choose between all previous commands, all possible operations should be prompt and the script must wait for user operations input. Also, operations with more than one possible usage (like turn screen and change keyboard language) should ask for the specific operation. Feel free to change messages and outputs. Flow example: Prompt all possible operations and wait for user input:

```
(1) Relaunch
(2) Restart
...
(4) Change keyboard language
```

Enter the operation:

Solution

```
#!/bin/bash

#START

##
# BASH menu script that perform:
#   - Shutdown
#   - Restart
#   - Turn screen
#   - Change keyboard language
##


function shutdown() {
    echo ""
    echo "Are you sure that you want to shutdown the computer ? [Y,n]"
    read input
    if [[ $input == "Y" || $input == "y" ]]; then
            sudo poweroff
            echo "Shutting down..."
    elif [[ $input == "N" || $input == "n" ]]; then
            echo "Your computer is not shutdown yet"
    else
            echo "Please answer with y or n"
    fi
    echo ""
}

function restart() {
    echo ""
    echo "The computer will reboot..."
    for i in {5..1}
    do
        echo $i
    done
    echo "Rebooting..."
    sleep 5; reboot
    echo ""
}

function turn_screen() {
```

```
    echo ""
    read input
    for var in "$input"
    do
        if [[ "$var" == "left" || "$var" == "l" ]]; then
            xrandr -o left
        elif [[ "$var" == "right" || "$var" == "r" ]]; then
            xrandr -o right
        else
            xrandr -o normal
        fi
    done
    echo ""
}

function change_keyboard_lang() {
    echo ""
    sudo locale-gen && cat /etc/locale.gen | grep "$1" > /etc/locale.conf
    echo ""
}

function all_checks() {
    shutdown
    restart
    turn_screen
    change_keyboard_lang
}

##
# Color  Variables
##
green='\e[32m'
blue='\e[34m'
clear='\e[0m'

##
# Color Functions
##

ColorGreen(){
    echo -ne $green$1$clear
}
ColorBlue(){
    echo -ne $blue$1$clear
}
```

```
menu(){
echo -ne "
My First Menu
$(ColorGreen '1)') Shut down the system
$(ColorGreen '2)') Restart the system
$(ColorGreen '3)') Turn the screen
$(ColorGreen '4)') Change keyboard language
$(ColorGreen '0)') Exit
$(ColorBlue 'Choose an option:') "
        read a
        case $a in
            1) shutdown ; menu ;;
            2) restart ; menu ;;
            3) turn_screen ; menu ;;
            4) change_keyboard_lang ; menu ;;
        0) exit 0 ;;
        *) echo -e $red"Wrong option."$clear; WrongCommand;;
        esac
}

# Call the menu function
menu
```

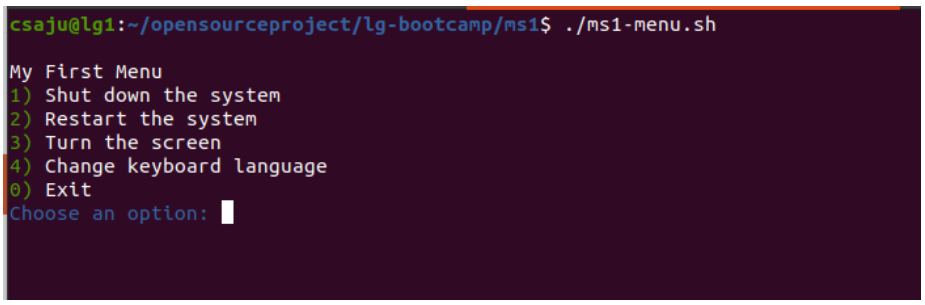

Figure 1: IMG

Here, all those above tasks are being combined and made it in a functional way to perform those tasks. User can select options and can peform those tasks easily.

MS2 SSH Communication Setup

SSH Basic connection and known hosts

Objectives

The objective of this task is to establish a connection between machines A and B using SSH protocol and perform some operations like:

- Ping one machine to another one.
- Use tcpdump to capture packets that are coming from the computer
- Check location of known hosts
- Check the history of machines A and B terminals and save the outputs in txt files

Solution

```
sudo apt-get install openssh-server
sudo service ssh start
```

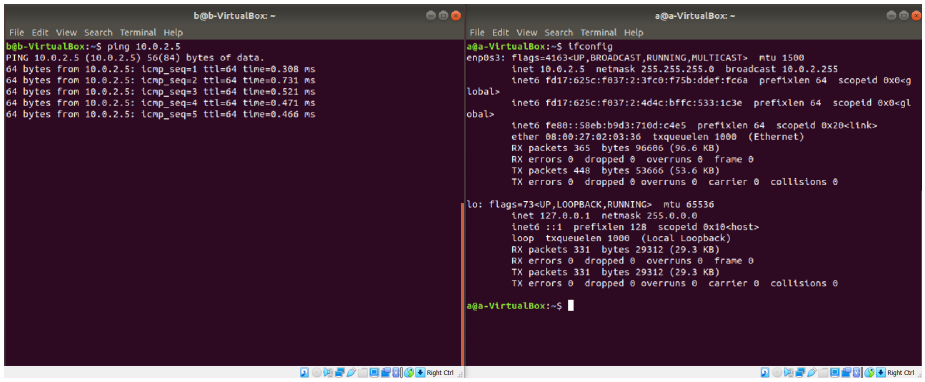

Figure 2: img

Here, I ping ip address of machine A from machine B. In order to lookup the ip address of machine, you go type

```
ifconfig
```

> Note: Make sure you created NAT network on the virtual machine so you ping to connected machine IPs.

On the machine A, tcpdump is used in order to capture request coming from the port 22 and machine B is trying to connect SSH connection between A and B.

Once the connection is successfuly, you can type any linux commands.

I had used history commands to save recently typed commands and to save it you can type.

```
history>machine-a-history.txt
```

Same goes for machine B too.

SSH and SCP

Objectives

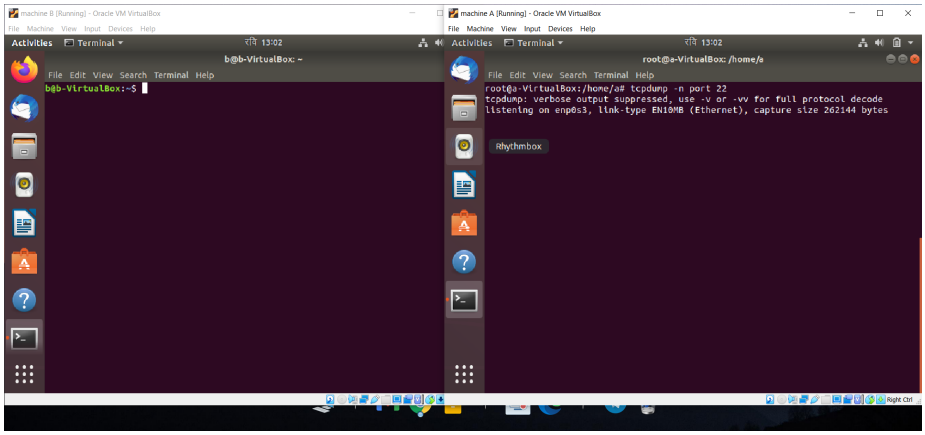

Figure 3: img

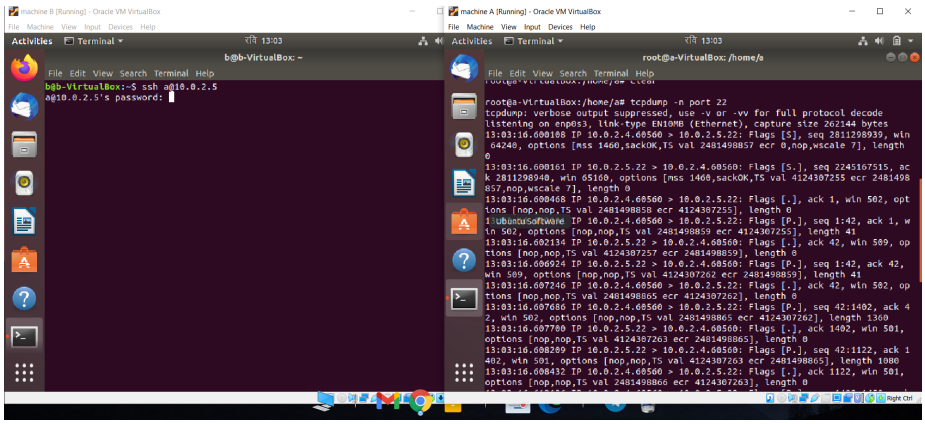

Figure 4: img

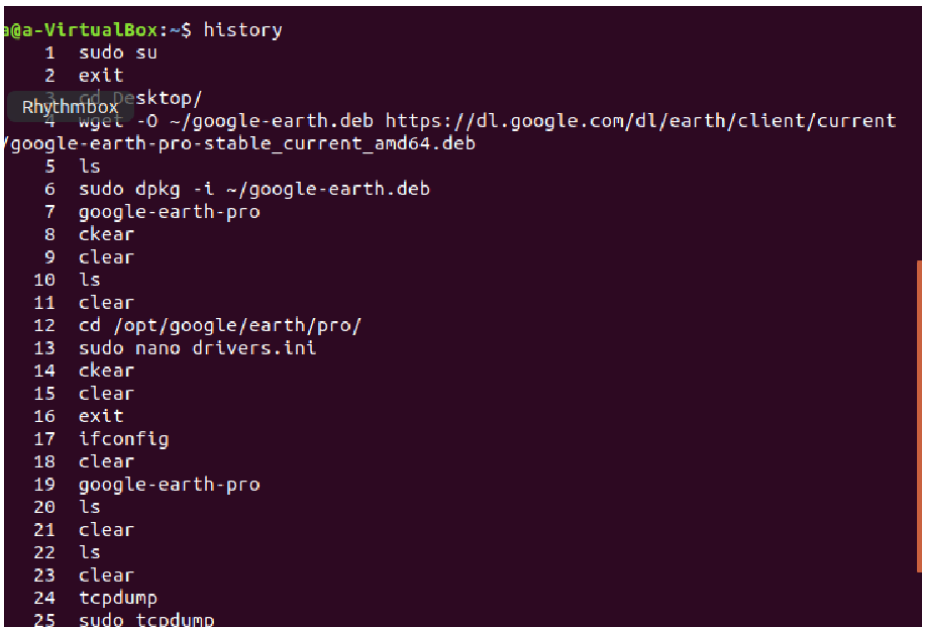

Figure 5: img

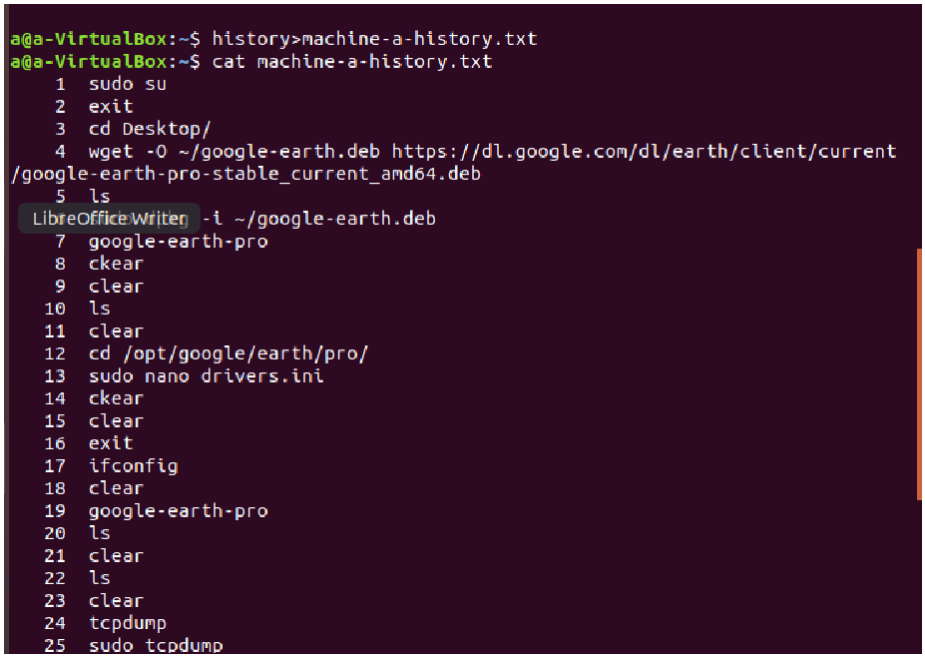

Figure 6: img

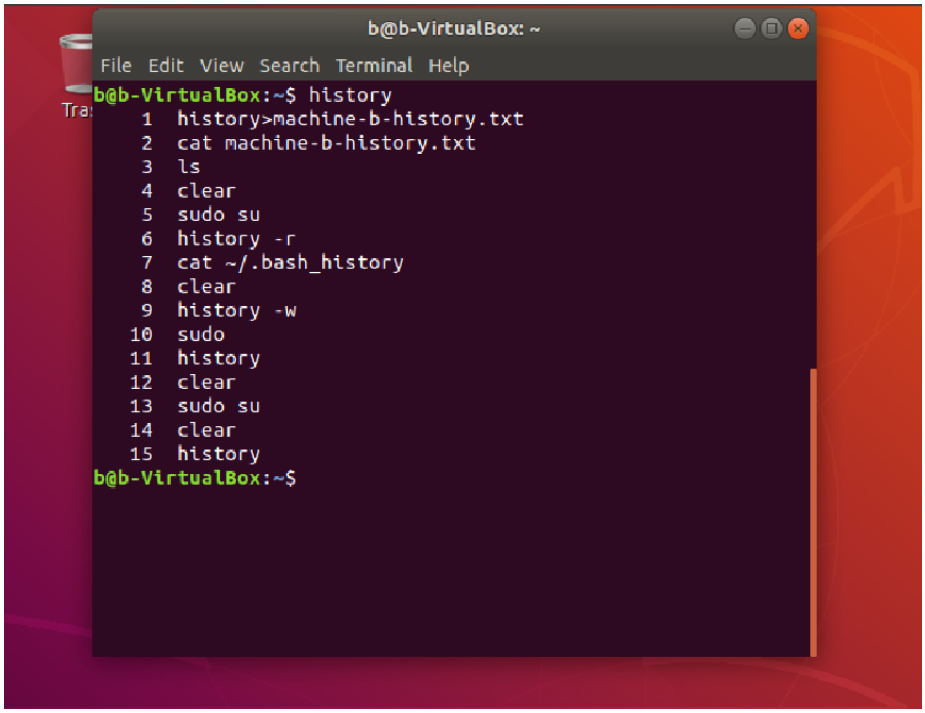
b@b-VirtualBox: ~
File Edit View Search Terminal Help
b@b-VirtualBox:~$ history
1 history>machine-b-history.txt
2 cat machine-b-history.txt
3 ls
4 clear
5 sudo su
6 history -r
7 cat ~/.bash_history
8 clear
9 history -w
10 sudo
11 history
12 clear
13 sudo su
14 clear
15 history
b@b-VirtualBox:~$

Figure 7: img

The objective of this task is to create a script that establishes a connection between a machine A and B and perform some operations that:

- establishes a connection with another computer in the same network using the SSH protocol. We'll be referring client as computer A and server as computer B
- Ping machine B from machine A
- Use tcpdump to capture packets that are coming from the computer
- Save all the packets transmitted in the first 10 seconds to a file
- Send the created file to machine A

Solution

```
#!/bin/bash
#START
usage(){
    echo "a == hostname";
    echo "b == ip";
    echo "p == pass"
    exit 1
}


scp_file(){
    echo "scp running...."
    sshpass -p $password scp  $hostname@$ip:ssh-ping-packets.pcap /tmp
    echo "success"
}

connect(){
    sshpass -p $password ssh $hostname@$ip "ping -c 2 $ip;\
    sudo -S tcpdump -G 2 -W 1 -i any -w ssh-ping-packets.pcap;"
    scp_file
}

while getopts a:b:c:d flag
do
    case "${flag}" in
        a) hostname=${OPTARG} ;;
        b) ip=${OPTARG} ;;
        c) password=${OPTARG} ;;
        *) usage;;
    esac
done
```

```
if [ $OPTIND -eq 1 ]; then
    echo "No options were passed"
    usage
else
    connect
fi

shift $((OPTIND-1))
```

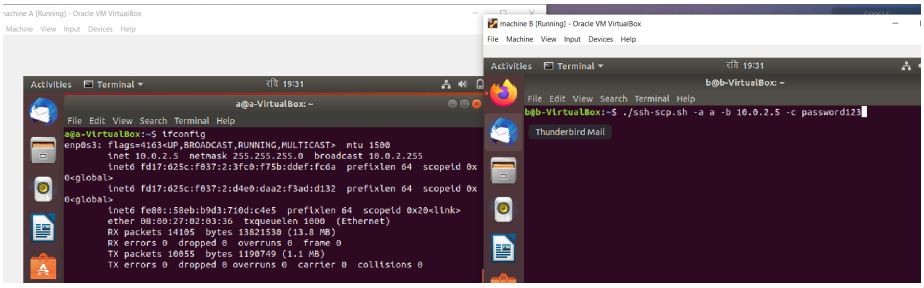

Figure 8: img

I pass the machine hostname argument as “a” and ipaddress as “10.0.2.5” and password as “password123”.

All the tasks will be performed which we had written on the bash script file. It create ssh connection and connect to the machine. It pings IP upto 10 seconds and tcpdump will the incoming packet request upto 10 seconds and saves the packets in the tmp folder.

I read the file using `cat`. You can also read and get the packets info using wireshark too.

Curl

Objectives

The objective of this task is to learn how to use the command CURL to:

- Check if a domain is up
- Make a http request
- Download files

To do that, you have to create a script that allows users to select a domain and a different action to be performed. The script needs to:

Have a menu where the user can choose between the options: - (1) Make GET request and receive a JSON - (2) Download a file from a remote server - Ask for the remote server domain - Check if domain is up - If domain is up, perform the action

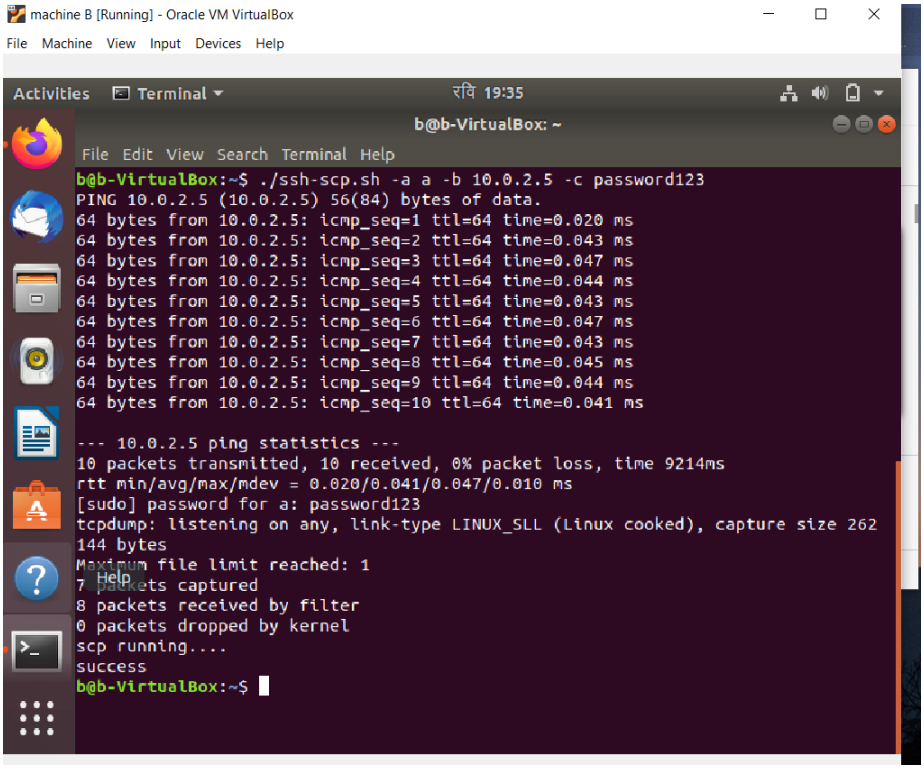

Figure 9: img

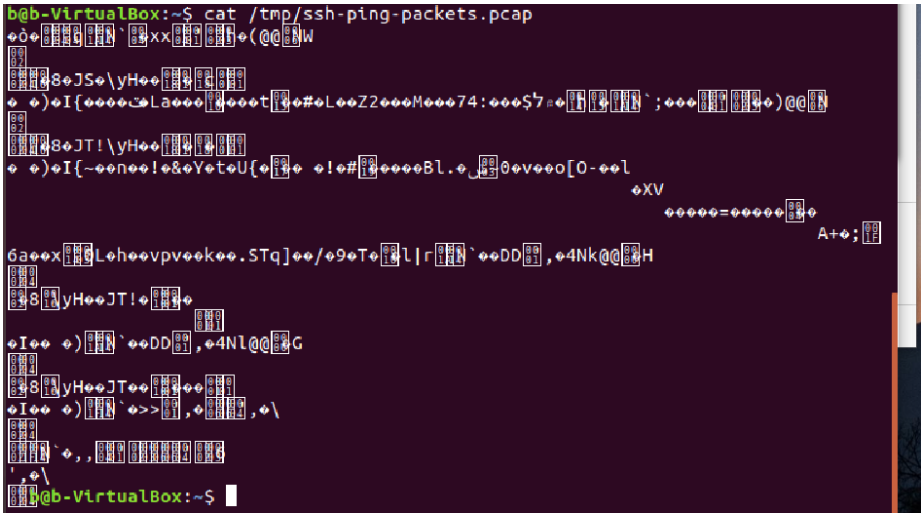

Figure 10: img

Solution

```
#!/bin/bash

checkDomainStatus(){
    echo "Enter a domain";
    read domain
    if curl -I $domain 2>&1 | grep -w "200\|301\|302"; then
        echo $domain "is up"
    else
        echo $domain "is down"
        exit 1
    fi
}

getResponse(){
    curl $domain -s -o get-json-response.json
    echo "Saved info to get-json-reponse.json"
}

makeHTTPReq(){
    checkDomainStatus
    getResponse
}

downloadFile(){
    checkDomainStatus
    if [[${domain:(-3)} -eq "zip"]]; then
        echo "It's a ZIP file"
        curl $domain -s -o file.zip
        unzip file.zip >/dev/null
        exit 1
    else
        echo "It's not a zip file"
        exit 1
    fi
}

##
# Color  Variables
##
green='\e[32m'
blue='\e[34m'
clear='\e[0m'

##
```

```
# Color Functions
##

ColorGreen(){
    echo -ne $green$1$clear
}
ColorBlue(){
    echo -ne $blue$1$clear
}

menu(){
echo -ne "
$(ColorGreen '1)') Make a http request
$(ColorGreen '2)') Download a file
$(ColorGreen '0)') Exit
$(ColorBlue 'Choose an option:') "
        read a
        case $a in
            1) makeHTTPReq ; menu ;;
            2) downloadFile ; menu ;;
        0) exit 0 ;;
        *) echo -e $red"Wrong option."$clear; WrongCommand;;
        esac
}

# Call the menu function
menu
```

```
1) Make a http request
2) Download a file
0) Exit
Choose an option: 1
Enter a domain
google.com
HTTP/1.1 301 Moved Permanently
google.com is up
Saved info to get-json-reponse.json

1) Make a http request
2) Download a file
0) Exit
Choose an option:
```

Figure 11: img

Here, I choose option 1 and hit the valid domain url and it gives the info about

valid domain name google.com is up(server is live). Once it shows the info, it saved info of request into new created file called get-json-response.json. You can read the content using `cat` command or any text editors.

```
csaju@lg1:~/opensourceproject/lg-bootcamp/ms2/T3$ cat get-json-response.json
<HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">
<TITLE>301 Moved</TITLE></HEAD><BODY>
<H1>301 Moved</H1>
The document has moved
<A HREF="http://www.google.com/">here</A>.
</BODY></HTML>
```

Figure 12: img

```
csaju@lg1:~/opensourceproject/lg-bootcamp/ms2/T3$ ./curl.sh

1) Make a http request
2) Download a file
0) Exit
Choose an option: 2
Enter a domain
http://stash.compciv.org/ssa_baby_names/names.zip
HTTP/1.1 200 OK
http://stash.compciv.org/ssa_baby_names/names.zip is up
It's a ZIP file
csaju@lg1:~/opensourceproject/lg-bootcamp/ms2/T3$
```

Figure 13: img

I choose here option 2 where I can download url having .zip file. I hit the url with .zip file and make sure the website is up. Once, it confirms the file extension is .zip, it will download the file and unzip it. In order to look up the extraction file, you can list the file using `ls`

```
csaju@lg1:~/opensourceproject/lg-bootcamp/ms2/T3$ ls
curl.sh                yob1892.txt  yob1908.txt  yob1924.txt  yob1940.txt  yob1956.txt  yob1972.txt  yob1988.txt  yob2004.txt
file.zip               yob1893.txt  yob1909.txt  yob1925.txt  yob1941.txt  yob1957.txt  yob1973.txt  yob1989.txt  yob2005.txt
get-json-response.json  yob1894.txt  yob1910.txt  yob1926.txt  yob1942.txt  yob1958.txt  yob1974.txt  yob1990.txt  yob2006.txt
NationalReadMe.pdf     yob1895.txt  yob1911.txt  yob1927.txt  yob1943.txt  yob1959.txt  yob1975.txt  yob1991.txt  yob2007.txt
yob1880.txt            yob1896.txt  yob1912.txt  yob1928.txt  yob1944.txt  yob1960.txt  yob1976.txt  yob1992.txt  yob2008.txt
yob1881.txt            yob1897.txt  yob1913.txt  yob1929.txt  yob1945.txt  yob1961.txt  yob1977.txt  yob1993.txt  yob2009.txt
yob1882.txt            yob1898.txt  yob1914.txt  yob1930.txt  yob1946.txt  yob1962.txt  yob1978.txt  yob1994.txt  yob2010.txt
yob1883.txt            yob1899.txt  yob1915.txt  yob1931.txt  yob1947.txt  yob1963.txt  yob1979.txt  yob1995.txt  yob2011.txt
yob1884.txt            yob1900.txt  yob1916.txt  yob1932.txt  yob1948.txt  yob1964.txt  yob1980.txt  yob1996.txt  yob2012.txt
yob1885.txt            yob1901.txt  yob1917.txt  yob1933.txt  yob1949.txt  yob1965.txt  yob1981.txt  yob1997.txt  yob2013.txt
yob1886.txt            yob1902.txt  yob1918.txt  yob1934.txt  yob1950.txt  yob1966.txt  yob1982.txt  yob1998.txt
yob1887.txt            yob1903.txt  yob1919.txt  yob1935.txt  yob1951.txt  yob1967.txt  yob1983.txt  yob1999.txt
yob1888.txt            yob1904.txt  yob1920.txt  yob1936.txt  yob1952.txt  yob1968.txt  yob1984.txt  yob2000.txt
yob1889.txt            yob1905.txt  yob1921.txt  yob1937.txt  yob1953.txt  yob1969.txt  yob1985.txt  yob2001.txt
yob1890.txt            yob1906.txt  yob1922.txt  yob1938.txt  yob1954.txt  yob1970.txt  yob1986.txt  yob2002.txt
yob1891.txt            yob1907.txt  yob1923.txt  yob1939.txt  yob1955.txt  yob1971.txt  yob1987.txt  yob2003.txt
csaju@lg1:~/opensourceproject/lg-bootcamp/ms2/T3$
```

What I learned

I learned about how the bash, Linux works and their importance on automating day-to-day workload. I learned debugging and troubleshooting skills when I got into the Linux environment. Apart from this Bootcamp, I also learned about the architecture of the Liquid Galaxy Core system and how it helps millions of people through this project.

Improvements

I think, having multiple problems for the tasks is required as there are limited problems that we have to solve and it will be much more beneficial if there is an IRC channel that needs little help for breaking down their problem and its solution.