Project: Linux Fundamental

Unit: CFC011023

Project Goal:
Understanding the usage of Bash scripting in Linux Fundamentals

Student No: S12
Student Name: Ivan Wong Sen Loong

Date of Completion: 27 October 2023

Table of Contents

Methodology

1.0.1 Displays the Linux version of your machine.

```
# 1. Display the Linux version of your machine
Version=$(cat /etc/os-release | grep VERSION_ID | awk -F= '{print$NF}' | tr -d '"')
echo "Initializing Script file in Linux "$Version" environment..."
read -n 1 -r -s -p $'Press enter to continue...\n'
echo "-----------------------------------------------"
```

Explanation

cat /etc/os-release | grep VERSION_ID | awk -F= '{print$NF}' | tr -d '"'

Using this website as a head start, it had directed me to go into the /etc directory first and locate the os-release file in which contains the version of my linux:

Sample outputs:

```
NAME="Ubuntu"
VERSION="20.04.1 LTS (Focal Fossa)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 20.04.1 LTS"
VERSION_ID="20.04"
```

To extract the the “20.04”, I had used the command of *grep VERSION_ID | awk -F= '{print$NF}' | tr -d '"'*. The breakdown of it is as follows:

grep VERSION_ID	To filter out the other info and directly read into the row of “VERSION_ID”
awk -F= '{print$NF}'	Starting with the awk command, use “=” as the field separator in the VERSION_ID row and with the combination of print + NF, prints out the 1st value in that row, counting from the back.
tr -d '"'	This is a translation command in which to delete the double quotation mark that exist in the extracted value.

1.0.2 Displays the Private IP, Public IP, and Default Gateway

```
# 2. Extract and display the Private IP, Public IP, and Default Gateway
private_ip=$(ifconfig | grep -A 1 'inet ' | grep -v '127.0.0.1' | awk '{print $2}' | cut -d':' -f2)
public_ip=$(curl -s ifconfig.me)
default_gateway=$(route -n | awk '$1 == "0.0.0.0" {print $2}')
echo "Displaying IP infos..."
echo "Private IP = $private_ip"
echo "Public IP = $public_ip"
echo "Default Gateway = $default_gateway"
read -n 1 -r -s -p $'Press enter to continue...\n'
echo "-------------------------------------------"
```

Explanation

1.) *private_ip=$(ifconfig | grep -A 1 'inet ' | grep -v '127.0.0.1' | awk '{print $2}' | cut -d':' -f2)*
2.) *public_ip=$(curl -s ifconfig.me)*
3.) *default_gateway=$(route -n | awk '$1 == "0.0.0.0" {print $2}')*

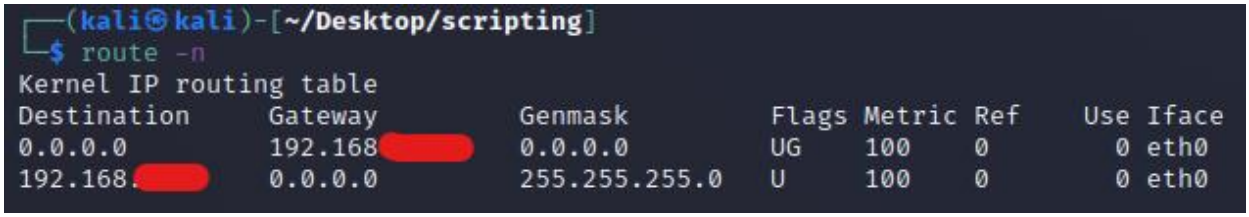
1.) ifconfig prints this out:

```
┌──(kali㉿kali)-[~/Desktop/scripting]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.[redacted]  netmask 255.255.255.0  broadcast 192.168.[redacted]
        inet6 fe80::20c:29ff:[redacted]  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:[redacted]  txqueuelen 1000  (Ethernet)
        RX packets 3439  bytes 221522 (216.3 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 502  bytes 41967 (40.9 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

grep -A 1 'inet '	*This part of the command uses grep to search for lines containing the word "inet" in the output of ifconfig. The "-A 1" option, which tells grep to include one line of context after the matching line. This is used to capture the line that contains the IP address information.*
grep -v '127.0.0.1'	*This line is included as it is used to exclude lines that contain "127.0.0.1," which is the loopback address and not a relevant IP address in this context.*
awk '{print $2}'	*The awk command is used to extract the second field (column) from each line of the remaining output. In the context of the output, the second field typically contains the IP address.*
cut -d':' -f2	*Finally, the cut command is used to split the line using the colon (":") as a delimiter and extract the second field, which is the IP address itself. This is necessary because ifconfig output can contain multiple pieces of information separated by colons.*

2.) curl -s ifconfig.me

3.) route -n | awk '$1 == "0.0.0.0" {print $2}



route -n	*The route command is used to display the kernel's routing table, which contains information about how network packets are routed. The -n option is used to display the numeric format (IP addresses and not hostnames) for better parsing.*
awk '$1 == "0.0.0.0" {print $2}	*In this part of the command, awk is used to process the output of the route -n command. Here's how it works:* *$1 refers to the first field (column) of each line in the route -n output. In the context of the routing table, the first field typically represents the destination IP address or network.* *== is the equality operator in awk. It checks if the first field is equal to "0.0.0.0," which represents the default route (the route for all outgoing traffic).* *{print $2} is the action to be performed when the condition is met. It instructs awk to print the second field (the gateway IP address) when the condition is true.*

1.0.3 Display the Hard Disk size, and free and used space

```
# 3. Display the Hard Disk size, and free and used space
echo "Analysizing your disk space..."
df -H /
read -n 1 -r -s -p $'Press enter to continue...\n'
echo "-----------------------------------------------"
```

Explanation of the commands used

df -H /

I've found that this command is best used to display all information about the root filesystem, which includes the hard disk size, used and avail(able) space. It even displays out in an invisible table format which was unbeknown to me.

```
Analysizing your disk space...
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1        83G   11G   68G  14% /
Press enter to continue...
```

I believe the terms for Size, Used, Avail, Use% is self-explanatory, so I had focus more on why this command is used. From google searching, this source had mentioned that *df* = basic reporting command and the -H directly means it to be "Human" Readable format. I also went to seek the info of the command's manual by inputting *man df* and sure enough it had given me a more accurate answer for the use case of -H:

```
-H, --si
       print sizes in powers of 1000 (e.g., 1.1G)
```

Which explains the difference of the output when typing *df* and *df -H:*

```
┌──(kali㉿kali)-[~/Desktop/scripting]
└─$ df
Filesystem     1K-blocks     Used Available Use% Mounted on
udev              966820        0    966820   0% /dev
tmpfs             202076     1192    200884   1% /run
/dev/sda1       81000912 10671588  66168712  14% /
tmpfs            1010376        0   1010376   0% /dev/shm
tmpfs               5120        0      5120   0% /run/lock
tmpfs             202072       68    202004   1% /run/user/1000

┌──(kali㉿kali)-[~/Desktop/scripting]
└─$ df -H
Filesystem      Size  Used Avail Use% Mounted on
udev            991M     0  991M   0% /dev
tmpfs           207M  1.3M  206M   1% /run
/dev/sda1        83G   11G   68G  14% /
tmpfs           1.1G     0  1.1G   0% /dev/shm
tmpfs           5.3M     0  5.3M   0% /run/lock
tmpfs           207M   70k  207M   1% /run/user/1000
```

1.0.4 Display the top 5 largest directories (according to size) in the virtual machine

```
# 4. Display the top 5 largest directories (according to size) in the virtual machine
echo "Displaying your Top 5 largest directory in descending order: "
du -a / 2>/dev/null | sort -n -r | head -n 5
read -n 1 -r -s -p $'Press enter to continue...\n'
echo "-----------------------------------------------"
```

Explanation

du -a / 2>/dev/null | sort -n -r | head -n 5

I have gotten this command from this source and a simple breakdown of the command would be:

```
$ sudo du -a / 2>/dev/null | sort -n -r | head -n 20
```

du -a /	du= disk usage -a=tells du to include all file and directories into its output / = to specify the starting point for calculating disk usage
2>/dev/null	This command redirects error messages (standard error) to /dev/null, which discards them. It's used to hide any error messages that might occur when trying to access certain directories for which the user running the command does not have permission.
sort -n -r	This command sorts the output of du numerically (-n) and in reverse order (-r), which means it will list the largest directories first.
head -n 5	Finally this command is put at the most back so that it only takes only the first five of the already sorted output.

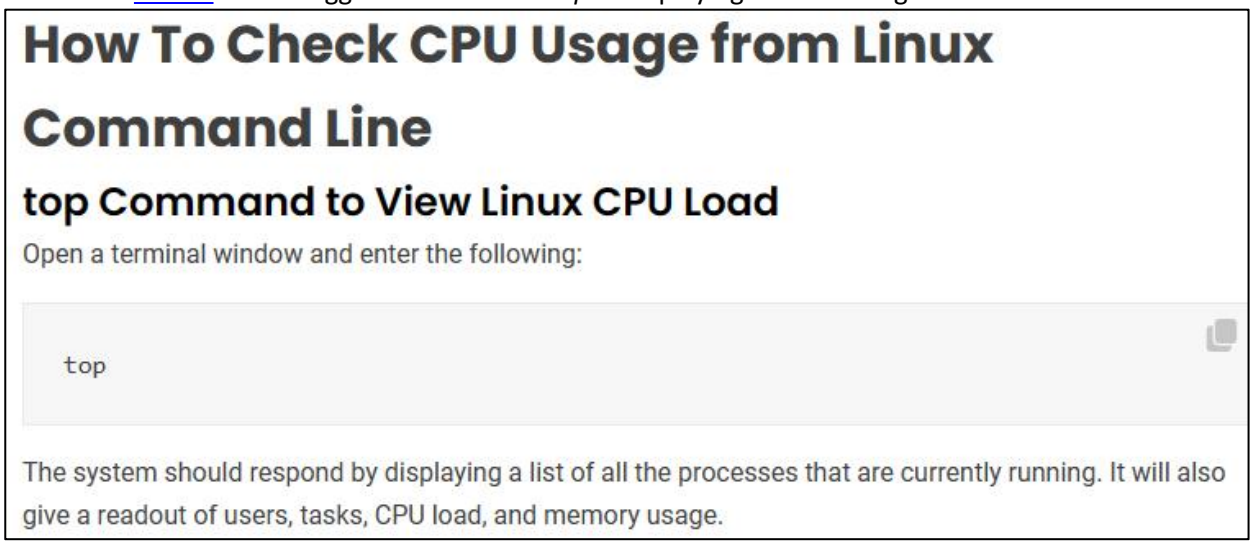
1.0.5 Display the CPU usage - refresh every 10 seconds.

```
# 5. Display the CPU usage - refresh every 10 seconds
echo "Displaying your CPU usage:"
top -d 10
```

Explanation

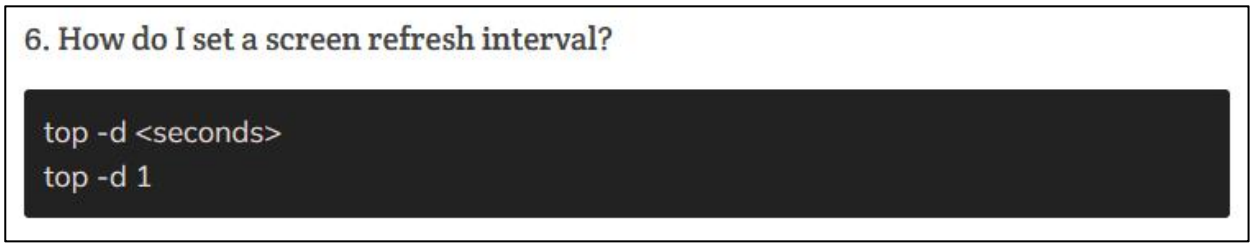
top -d 10

Based on my input of "what is the command to display CPU usage in Linux" into google, I have come across this source which suggested me to use *top* in displaying the CPU usage of Linux.



As for the "refresh every 10 seconds" part of the question, I first google the input with "how to reinput a command for an x amount of time" as I thought I would have to use a loop function which would set an interval of 10 seconds in executing the *top* command.

Upon further googling, this website has enlightened me that the top command has its own innate refresh command which is d follow by an interval second.

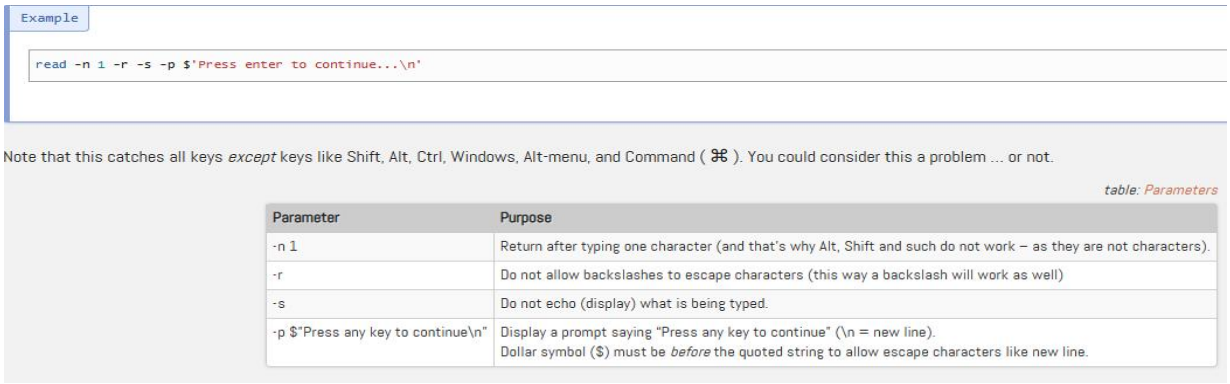


With this knowledge in mind, it gave birth to the command *top -d 10* which is used in my bash script to display the CPU usage and refreshes it every 10 seconds.

Extra commands used in the bash script

1.) *read -n 1 -r -s -p $'Press enter to continue...\n'*

[*Source*]



Example

```
read -n 1 -r -s -p $'Press enter to continue...\n'
```

Note that this catches all keys *except* keys like Shift, Alt, Ctrl, Windows, Alt-menu, and Command (⌘). You could consider this a problem ... or not.

table: Parameters

Parameter	Purpose
-n 1	Return after typing one character (and that's why Alt, Shift and such do not work – as they are not characters).
-r	Do not allow backslashes to escape characters (this way a backslash will work as well)
-s	Do not echo (display) what is being typed.
-p $"Press any key to continue\n"	Display a prompt saying "Press any key to continue" (\n = new line). Dollar symbol ($) must be *before* the quoted string to allow escape characters like new line.

Purpose of using: To systematically run the bash script by waiting for an ENTER input from the user when a process had finish executing.

2.) *echo "----------"*

Purpose of using: For better viewing layout of each process.