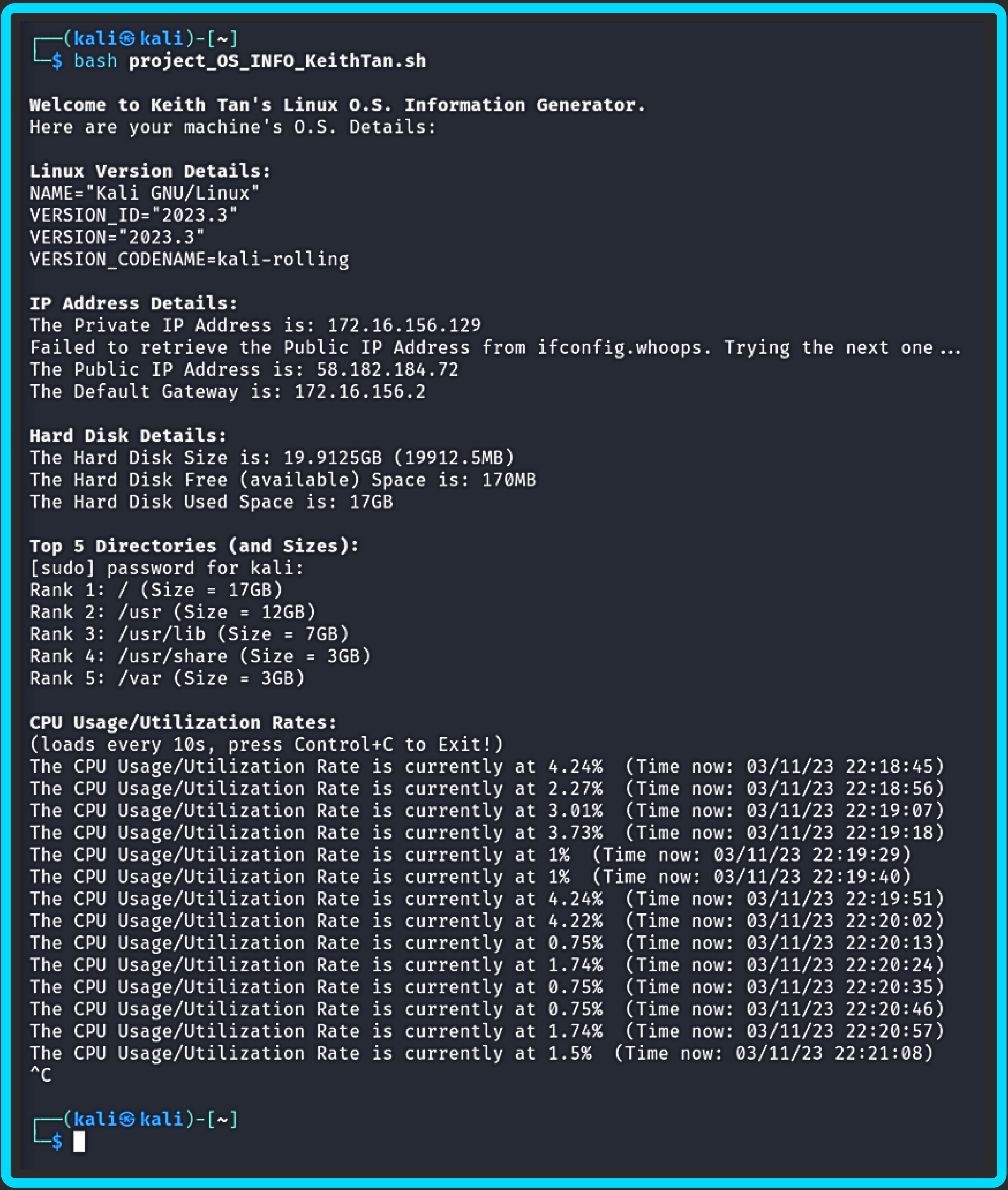
PROJECT:
O.S. INFO

LINUX FUNDAMENTALS

TAN KE HAN KEITH
[S9]

PROJECT_OS_INFO_KEITHTAN.SH

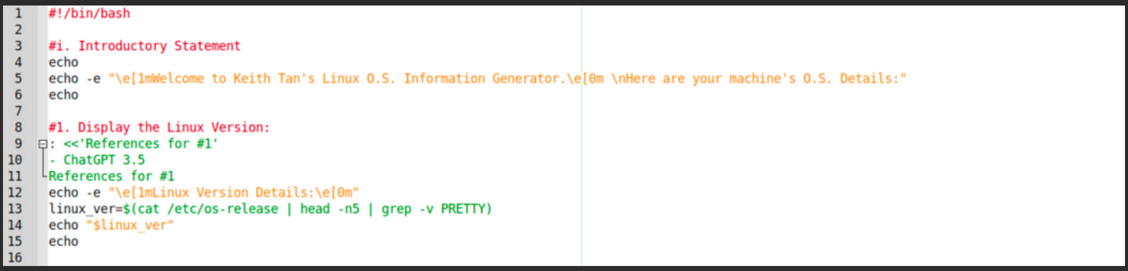
Bashed! by Keith Tan



```
┌──(kali㉿kali)-[~]
└─$ bash project_OS_INFO_KeithTan.sh

Welcome to Keith Tan's Linux O.S. Information Generator.
Here are your machine's O.S. Details:

Linux Version Details:
NAME="Kali GNU/Linux"
VERSION_ID="2023.3"
VERSION="2023.3"
VERSION_CODENAME=kali-rolling

IP Address Details:
The Private IP Address is: 172.16.156.129
Failed to retrieve the Public IP Address from ifconfig.whoops. Trying the next one...
The Public IP Address is: 58.182.184.72
The Default Gateway is: 172.16.156.2

Hard Disk Details:
The Hard Disk Size is: 19.9125GB (19912.5MB)
The Hard Disk Free (available) Space is: 170MB
The Hard Disk Used Space is: 17GB

Top 5 Directories (and Sizes):
[sudo] password for kali:
Rank 1: / (Size = 17GB)
Rank 2: /usr (Size = 12GB)
Rank 3: /usr/lib (Size = 7GB)
Rank 4: /usr/share (Size = 3GB)
Rank 5: /var (Size = 3GB)

CPU Usage/Utilization Rates:
(loads every 10s, press Control+C to Exit!)
The CPU Usage/Utilization Rate is currently at 4.24%  (Time now: 03/11/23 22:18:45)
The CPU Usage/Utilization Rate is currently at 2.27%  (Time now: 03/11/23 22:18:56)
The CPU Usage/Utilization Rate is currently at 3.01%  (Time now: 03/11/23 22:19:07)
The CPU Usage/Utilization Rate is currently at 3.73%  (Time now: 03/11/23 22:19:18)
The CPU Usage/Utilization Rate is currently at 1%  (Time now: 03/11/23 22:19:29)
The CPU Usage/Utilization Rate is currently at 1%  (Time now: 03/11/23 22:19:40)
The CPU Usage/Utilization Rate is currently at 4.24%  (Time now: 03/11/23 22:19:51)
The CPU Usage/Utilization Rate is currently at 4.22%  (Time now: 03/11/23 22:20:02)
The CPU Usage/Utilization Rate is currently at 0.75%  (Time now: 03/11/23 22:20:13)
The CPU Usage/Utilization Rate is currently at 1.74%  (Time now: 03/11/23 22:20:24)
The CPU Usage/Utilization Rate is currently at 0.75%  (Time now: 03/11/23 22:20:35)
The CPU Usage/Utilization Rate is currently at 0.75%  (Time now: 03/11/23 22:20:46)
The CPU Usage/Utilization Rate is currently at 1.74%  (Time now: 03/11/23 22:20:57)
The CPU Usage/Utilization Rate is currently at 1.5%  (Time now: 03/11/23 22:21:08)
^C

┌──(kali㉿kali)-[~]
└─$ █
```

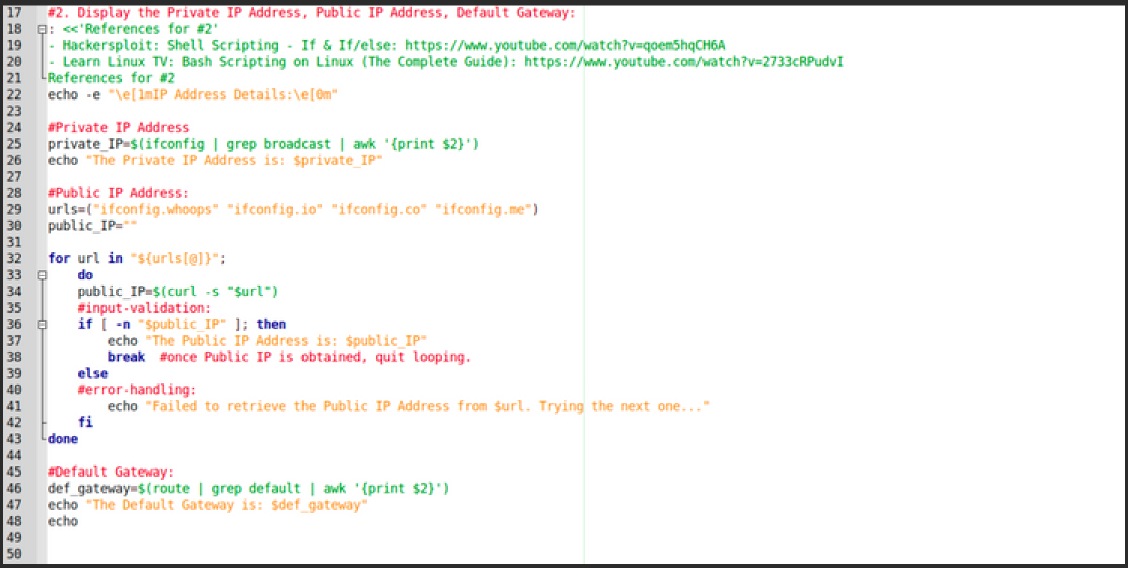
SCRIPT BREAKDOWN

1. PREFACE + DISPLAY LINUX VERSION



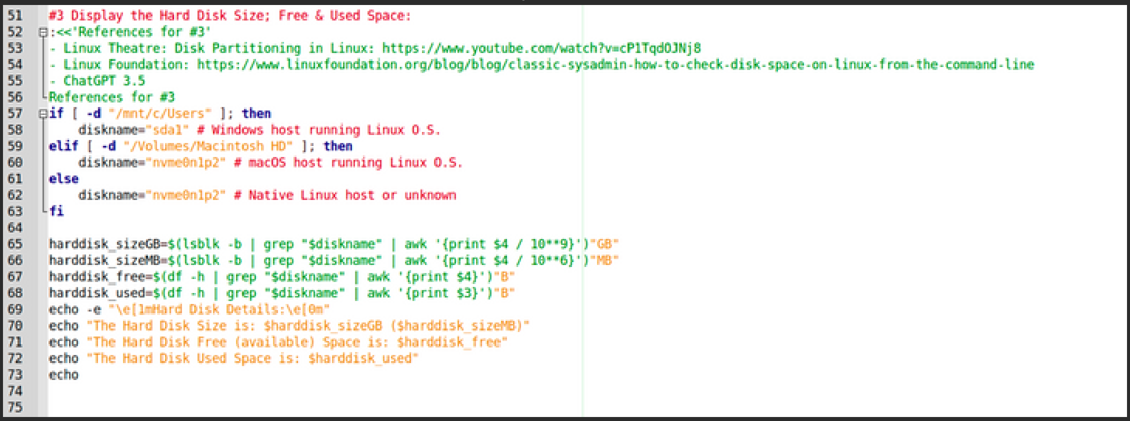
```bash
#!/bin/bash

#i. Introductory Statement
echo
echo -e "\e[1mWelcome to Keith Tan's Linux O.S. Information Generator.\e[0m \nHere are your machine's O.S. Details:"
echo

#1. Display the Linux Version:
: <<'References for #1'
- ChatGPT 3.5
References for #1
echo -e "\e[1mLinux Version Details:\e[0m"
linux_ver=$(cat /etc/os-release | head -n5 | grep -v PRETTY)
echo "$linux_ver"
echo

```

2. DISPLAY IP ADDRESSES + DEFAULT GATEWAY



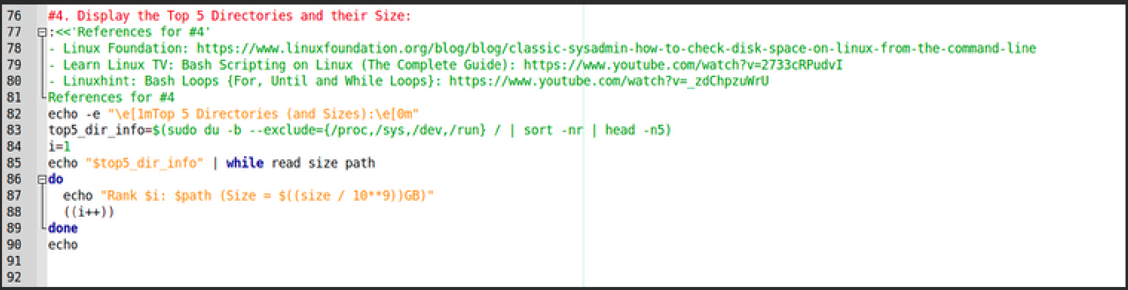
```bash
#2. Display the Private IP Address, Public IP Address, Default Gateway:
: <<'References for #2'
- Hackersploit: Shell Scripting - If & If/else: https://www.youtube.com/watch?v=qoem5hqCH6A
- Learn Linux TV: Bash Scripting on Linux (The Complete Guide): https://www.youtube.com/watch?v=2733cRPudvI
References for #2
echo -e "\e[1mIP Address Details:\e[0m"

#Private IP Address
private_IP=$(ifconfig | grep broadcast | awk '{print $2}')
echo "The Private IP Address is: $private_IP"

#Public IP Address:
urls=("ifconfig.whoops" "ifconfig.io" "ifconfig.co" "ifconfig.me")
public_IP=""

for url in "${urls[@]}";
    do
    public_IP=$(curl -s "$url")
    #input-validation:
    if [ -n "$public_IP" ]; then
        echo "The Public IP Address is: $public_IP"
        break  #once Public IP is obtained, quit looping.
    else
    #error-handling:
        echo "Failed to retrieve the Public IP Address from $url. Trying the next one..."
    fi
done

#Default Gateway:
def_gateway=$(route | grep default | awk '{print $2}')
echo "The Default Gateway is: $def_gateway"
echo


```

3. DISPLAY HARD DISK SIZE, FREE & USED SPACES



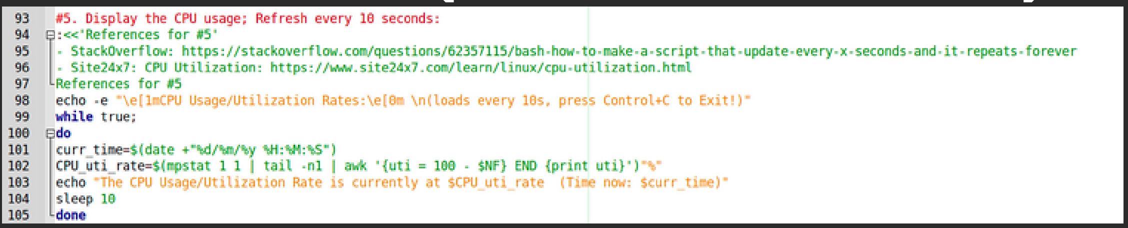
```bash
#3 Display the Hard Disk Size; Free & Used Space:
:<<'References for #3'
- Linux Theatre: Disk Partitioning in Linux: https://www.youtube.com/watch?v=cP1TqdOJNj8
- Linux Foundation: https://www.linuxfoundation.org/blog/blog/classic-sysadmin-how-to-check-disk-space-on-linux-from-the-command-line
- ChatGPT 3.5
References for #3
if [ -d "/mnt/c/Users" ]; then
    diskname="sda1" # Windows host running Linux O.S.
elif [ -d "/Volumes/Macintosh HD" ]; then
    diskname="nvme0n1p2" # macOS host running Linux O.S.
else
    diskname="nvme0n1p2" # Native Linux host or unknown
fi

harddisk_sizeGB=$(lsblk -b | grep "$diskname" | awk '{print $4 / 10**9}')"GB"
harddisk_sizeMB=$(lsblk -b | grep "$diskname" | awk '{print $4 / 10**6}')"MB"
harddisk_free=$(df -h | grep "$diskname" | awk '{print $4}')"B"
harddisk_used=$(df -h | grep "$diskname" | awk '{print $3}')"B"
echo -e "\e[1mHard Disk Details:\e[0m"
echo "The Hard Disk Size is: $harddisk_sizeGB ($harddisk_sizeMB)"
echo "The Hard Disk Free (available) Space is: $harddisk_free"
echo "The Hard Disk Used Space is: $harddisk_used"
echo


```

4. DISPLAY TOP 5 DIRECTORIES & SIZES



```bash
#4. Display the Top 5 Directories and their Size:
:<<'References for #4'
- Linux Foundation: https://www.linuxfoundation.org/blog/blog/classic-sysadmin-how-to-check-disk-space-on-linux-from-the-command-line
- Learn Linux TV: Bash Scripting on Linux (The Complete Guide): https://www.youtube.com/watch?v=2733cRPudvI
- Linuxhint: Bash Loops {For, Until and While Loops}: https://www.youtube.com/watch?v=_zdChpzuWrU
References for #4
echo -e "\e[1mTop 5 Directories (and Sizes):\e[0m"
top5_dir_info=$(sudo du -b --exclude={/proc,/sys,/dev,/run} / | sort -nr | head -n5)
i=1
echo "$top5_dir_info" | while read size path
do
  echo "Rank $i: $path (Size = $((size / 10**9))GB)"
  ((i++))
done
echo


```

5. DISPLAY CPU USAGE (REFRESH EVERY 10SECS)

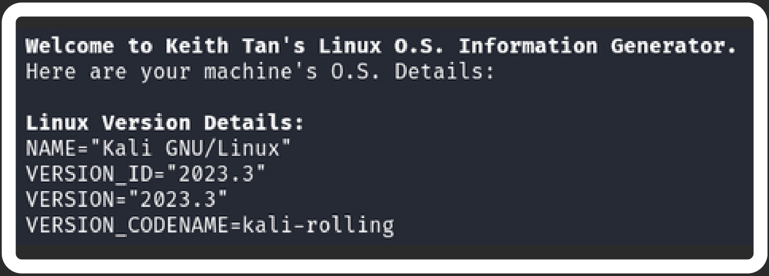


```bash
#5. Display the CPU usage; Refresh every 10 seconds:
:<<'References for #5'
- StackOverflow: https://stackoverflow.com/questions/62357115/bash-how-to-make-a-script-that-update-every-x-seconds-and-it-repeats-forever
- Site24x7: CPU Utilization: https://www.site24x7.com/learn/linux/cpu-utilization.html
References for #5
echo -e "\e[1mCPU Usage/Utilization Rates:\e[0m \n(loads every 10s, press Control+C to Exit!)"
while true;
do
curr_time=$(date +"%d/%m/%y %H:%M:%S")
CPU_uti_rate=$(mpstat 1 1 | tail -n1 | awk '{uti = 100 - $NF} END {print uti}')"%"
echo "The CPU Usage/Utilization Rate is currently at $CPU_uti_rate  (Time now: $curr_time)"
sleep 10
done
```

DOCUMENTATION

1. PREFACE + DISPLAY LINUX VERSION

FINAL OUTPUT EXAMPLE:



```
Welcome to Keith Tan's Linux O.S. Information Generator.
Here are your machine's O.S. Details:

Linux Version Details:
NAME="Kali GNU/Linux"
VERSION_ID="2023.3"
VERSION="2023.3"
VERSION_CODENAME=kali-rolling
```

#!/bin/bash

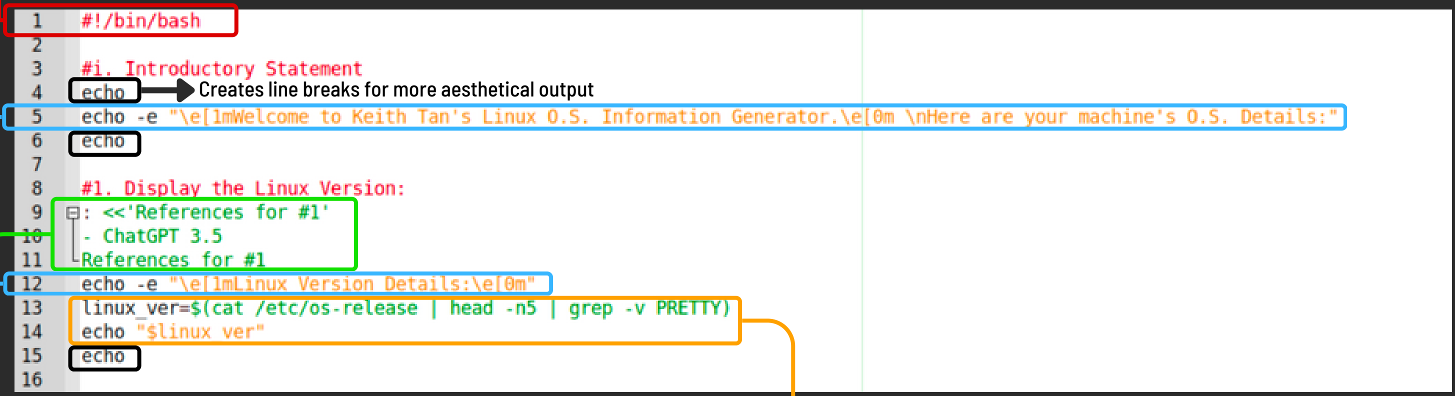
- A necessary shebang line that tells interpreter to use Bash to execute the script.

echo -e

echo: displays message onto the terminal

-e: allows message to perform the following escape sequences -

- \e[1m *<bolds text>* \e[0m
- *<text 1>* \n *<text 2 starts on a new line>*



```
#!/bin/bash

#i. Introductory Statement
echo
echo -e "\e[1mWelcome to Keith Tan's Linux O.S. Information Generator.\e[0m \nHere are your machine's O.S. Details:"
echo

#1. Display the Linux Version:
: <<'References for #1'
- ChatGPT 3.5
References for #1
echo -e "\e[1mLinux Version Details:\e[0m"
linux_ver=$(cat /etc/os-release | head -n5 | grep -v PRETTY)
echo "$linux_ver"
echo

```

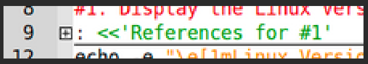
: << '<Comment Title>'

<Comments>

<Comments>

Comment Title

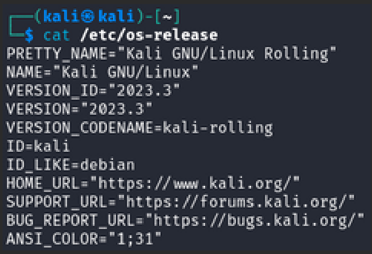
- This command creates multi-line comments without having to use # at the start of every comment line. Also allows collapsing for neater code:



```
: <<'References for #1'
```

cat /etc/os-release

- The cat command opens the standard configuration file /etc/os-release, which contains information about the machine's Linux O.S.



```
┌──(kali㉿kali)-[~]
└─$ cat /etc/os-release
PRETTY_NAME="Kali GNU/Linux Rolling"
NAME="Kali GNU/Linux"
VERSION_ID="2023.3"
VERSION="2023.3"
VERSION_CODENAME=kali-rolling
ID=kali
ID_LIKE=debian
HOME_URL="https://www.kali.org/"
SUPPORT_URL="https://forums.kali.org/"
BUG_REPORT_URL="https://bugs.kali.org/"
ANSI_COLOR="1;31"
```

| head -n5 | grep -v PRETTY

- Text manipulation is conducted to filter only the essential Linux O.S. Information.
- The head -n5 command selects only the top 5 rows.
- The grep -v PRETTY command omits the row with 'PRETTY' in it.

linux_ver=$( )

echo

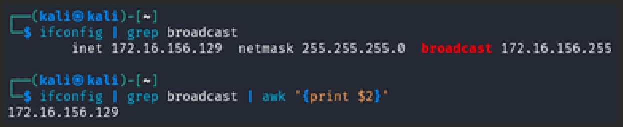
- New variable linux_ver is created to store the filtered Linux O.S. information within $( ).
- echo displays the final output.

DOCUMENTATION

2. DISPLAY IP ADDRESSES + DEFAULT GATEWAY

private_IP=$(ifconfig | grep broadcast | awk '{print $2}')

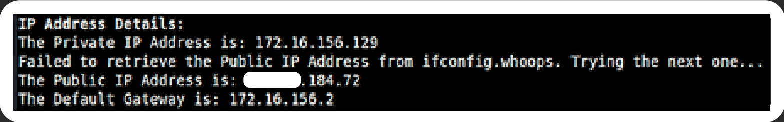
- ifconfig displays network configurations like private IP address
- grep broadcast filters to the row containing 'broadcast'
- awk '{print $2}' selects 2nd column for private IP address (after 'inet')



```
┌──(kali㉿kali)-[~]
└─$ ifconfig | grep broadcast
        inet 172.16.156.129  netmask 255.255.255.0  broadcast 172.16.156.255

┌──(kali㉿kali)-[~]
└─$ ifconfig | grep broadcast | awk '{print $2}'
172.16.156.129
```

- The private IP address is stored into new variable private_IP

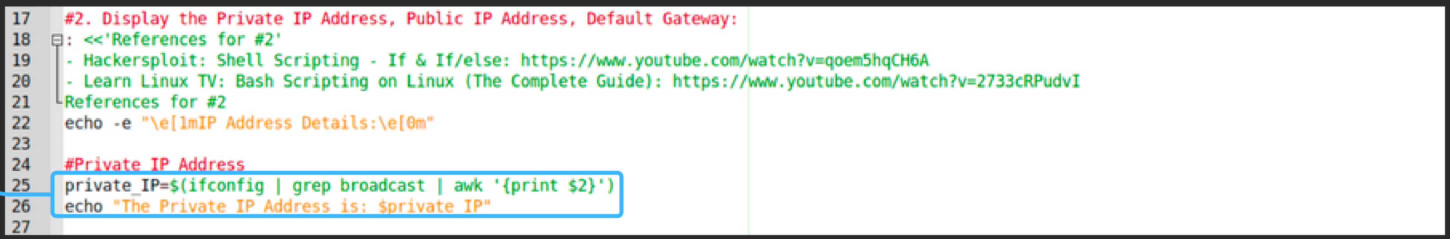
FINAL OUTPUT EXAMPLE:



```
IP Address Details:
The Private IP Address is: 172.16.156.129
Failed to retrieve the Public IP Address from ifconfig.whoops. Trying the next one...
The Public IP Address is: [redacted].184.72
The Default Gateway is: 172.16.156.2
```

echo

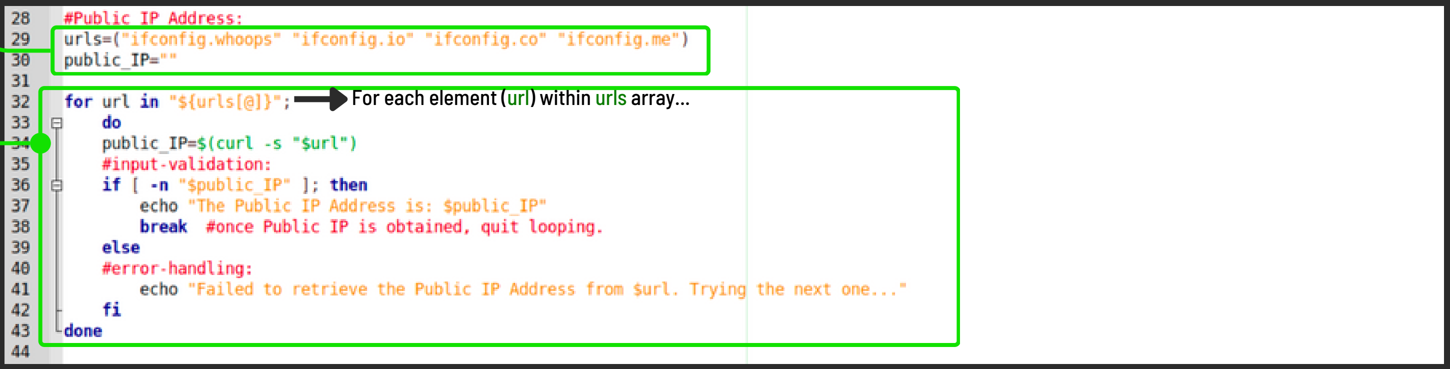
- Displays the text message and contents within private_IP



```
17  #2. Display the Private IP Address, Public IP Address, Default Gateway:
18  : <<'References for #2'
19  - Hackersploit: Shell Scripting - If & If/else: https://www.youtube.com/watch?v=qoem5hqCH6A
20  - Learn Linux TV: Bash Scripting on Linux (The Complete Guide): https://www.youtube.com/watch?v=2733cRPudvI
21  References for #2
22  echo -e "\e[1mIP Address Details:\e[0m"
23  
24  #Private IP Address
25  private_IP=$(ifconfig | grep broadcast | awk '{print $2}')
26  echo "The Private IP Address is: $private_IP"
27  
```

PRIVATE IP ADDR

- Global Scope: New array urls is created and stored with 4 elements of domain names.
- Global Scope: New variable public_IP is created with empty string (to receive string within 'for' loop later).



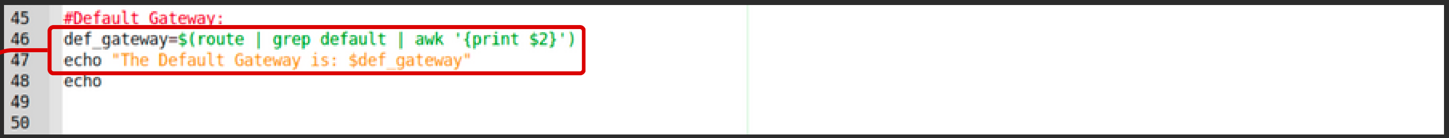
```
28  #Public IP Address:
29  urls=("ifconfig.whoops" "ifconfig.io" "ifconfig.co" "ifconfig.me")
30  public_IP=""
31  
32  for url in "${urls[@]}";
33      do
34      public_IP=$(curl -s "$url")
35      #input-validation:
36      if [ -n "$public_IP" ]; then
37          echo "The Public IP Address is: $public_IP"
38          break  #once Public IP is obtained, quit looping.
39      else
40      #error-handling:
41          echo "Failed to retrieve the Public IP Address from $url. Trying the next one..."
42      fi
43  done
44  
```

For each element (url) within urls array...

PUBLIC IP ADDR

Explaining the 'for' loop + 'if-else' (Lines 32-43):

- **Lines 32-33:** url represents elements within the array urls. For each element (url) within urls array, do the following...
- **Line 34:** curl retrieves the public IP address from $url (which stores domain names). -s prevents request/retrieval progress information from displaying. The output (a public IP address) is stored into the variable public_IP.
- **Lines 36-42:** Displays an 'if-else' statement.
  - Lines 36-38: If the stored value within public_IP (which should contain a public IP address if retrieval by curl from url is successful) is <u>non-empty</u> (denoted by -n)...
    - echo to display text message and value within public_IP
    - break and stop the Loop
  - Lines 39-42: This line will run if public IP address retrieval by curl from url is unsuccessful.
    - echo to display text message
    - escape if-else and return to 'for' loop (Line 32-33), where the next url in urls array is tested.



```
45  #Default Gateway:
46  def_gateway=$(route | grep default | awk '{print $2}')
47  echo "The Default Gateway is: $def_gateway"
48  echo
49  
50  
```

DEFAULT GATEWAY

def_gateway=$(route | grep default | awk '{print $2}')

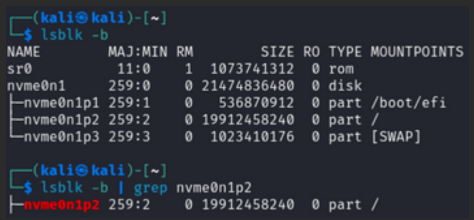
- route displays the kernel routing table, inclusive of the default gateway IP address.
- grep default filters the row with 'default'.
- awk '{print $2}' filters to the 2nd column, to be left with the default gateway IP address, which is then stored in new variable def_gateway

echo displays the text message and value within def_gateway

DOCUMENTATION

3. DISPLAY HARDDISK SIZE, FREE & USED SPACES

- harddisk_sizeGB=$(lsblk -b | grep "$diskname" | awk '{print $4 / 10**9}')"GB"
- harddisk_sizeMB=$(lsblk -b | grep "$diskname" | awk '{print $4 / 10**6}')"MB"
  - lsblk lists information about block devices (hard drives & partitions).
    - -b displays block device sizes in Bytes.
  - grep "$diskname": Lines 57-63 if-else statement determines the disk name to grep based on host machine's O.S. (that runs the Linux machine), then stores result into new variable diskname. For Mac O.S. host, command grep "$diskname" filters the row containing the disk name called 'nvme0n1p2'.
    - nvme0n1: The full NVM Express SSD, with 3 partitions denoted by 'p1', 'p2', 'p3'. For this project, assume hard disk refers to 'p2'.
      - p1: 1st Partition, mounted as EFI system partition (contains files & data for booting system in UEFI.)
      - p2: 2nd Partition, mounted as root '/' filesystem (contains core system files & directories needed for O.S. to function.)
      - p3: 3rd Partition, the 'swap' partition (Linux memory management system, provides extra memory space when RAM is fully used.)



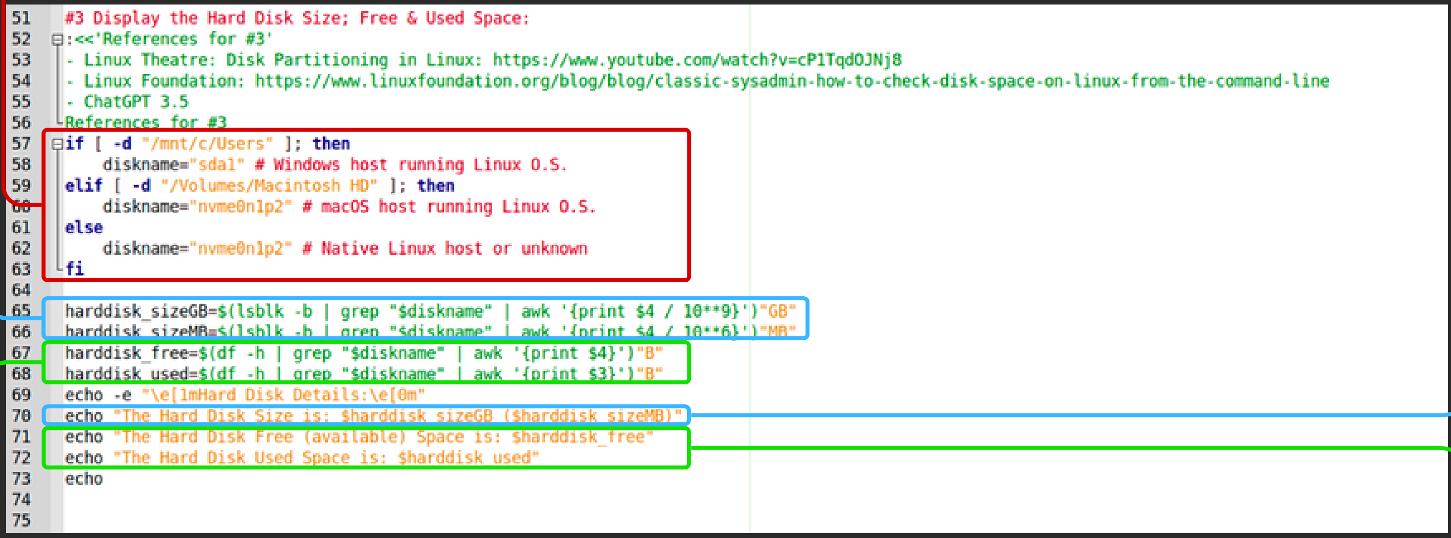
```
┌──(kali㉿kali)-[~]
└─$ lsblk -b
NAME        MAJ:MIN RM        SIZE RO TYPE MOUNTPOINTS
sr0          11:0    1  1073741312  0 rom
nvme0n1     259:0    0 21474836480  0 disk
├─nvme0n1p1 259:1    0   536870912  0 part /boot/efi
├─nvme0n1p2 259:2    0 19912458240  0 part /
└─nvme0n1p3 259:3    0  1023410176  0 part [SWAP]

┌──(kali㉿kali)-[~]
└─$ lsblk -b | grep nvme0n1p2
├─nvme0n1p2 259:2    0 19912458240  0 part /
```

- sr0: CV/DVD-ROM/Blu-ray Drive
- MAJ:MIN: Major & minor device numbers
- RM: Removable? (0=false; 1=true)
- SIZE: Capacity in Bytes
- RO: Read-only? (0=false; 1=true)
- TYPE 'rom': Read-only memory device
- TYPE 'disk': Block storage device, usually for primary data storage
- MOUNTPOINTS: Directories to access data on mounted device (refer to TYPE)

- awk '{print $4 / 10**9}"GB" & awk '{print $4 / 10**6}"MB" (understand -h auto-converts, but wanted to try converting myself)
  - Filters 4th column (SIZE) to get a single numerical result in Bytes. Converts Bytes to MB & GB respectively. Adds string "MB" / "GB" for units.

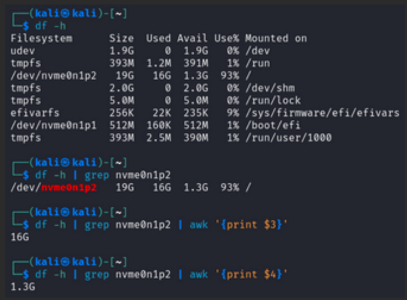
Hard disk sizes are stored in variables harddisk_sizeGB and harddisk_sizeMB.

Line 70: echo displays the hard disk sizes with text message.



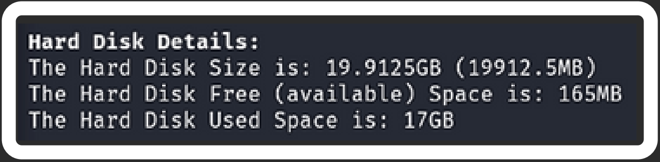
```
51  #3 Display the Hard Disk Size; Free & Used Space:
52  :<<'References for #3'
53  - Linux Theatre: Disk Partitioning in Linux: https://www.youtube.com/watch?v=cP1TqdOJNj8
54  - Linux Foundation: https://www.linuxfoundation.org/blog/blog/classic-sysadmin-how-to-check-disk-space-on-linux-from-the-command-line
55  - ChatGPT 3.5
56  References for #3
57  if [ -d "/mnt/c/Users" ]; then
58      diskname="sda1" # Windows host running Linux O.S.
59  elif [ -d "/Volumes/Macintosh HD" ]; then
60      diskname="nvme0n1p2" # macOS host running Linux O.S.
61  else
62      diskname="nvme0n1p2" # Native Linux host or unknown
63  fi
64
65  harddisk_sizeGB=$(lsblk -b | grep "$diskname" | awk '{print $4 / 10**9}')"GB"
66  harddisk_sizeMB=$(lsblk -b | grep "$diskname" | awk '{print $4 / 10**6}')"MB"
67  harddisk_free=$(df -h | grep "$diskname" | awk '{print $4}')"B"
68  harddisk_used=$(df -h | grep "$diskname" | awk '{print $3}')"B"
69  echo -e "\e[1mHard Disk Details:\e[0m"
70  echo "The Hard Disk Size is: $harddisk_sizeGB ($harddisk_sizeMB)"
71  echo "The Hard Disk Free (available) Space is: $harddisk_free"
72  echo "The Hard Disk Used Space is: $harddisk_used"
73  echo
74
75
```

- harddisk_free=$(df -h | grep "$diskname" | awk '{print $4}') "B"
- harddisk_used=$(df -h | grep nvme0n1p2 | awk '{print $3}') "B"
  - df (a.k.a. Disk Free) displays information about disk space for mounted filesystems/devices.
    - -h (human-readable) formats the result to include units
      (K: kilobytes; M: megabytes; G: gigabytes; T: terabytes)
  - grep "$diskname": filters the row containing 'nvme0n1p2'.
  - awk '{print $3}' }' & awk '{print $4}'
    - Filters 3rd column (Used) and 4th column (Avail) to get a single numerical results.



```
┌──(kali㉿kali)-[~]
└─$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            1.9G     0  1.9G   0% /dev
tmpfs           393M  1.2M  391M   1% /run
/dev/nvme0n1p2   19G   16G  1.3G  93% /
tmpfs           2.0G     0  2.0G   0% /dev/shm
tmpfs           5.0M     0  5.0M   0% /run/lock
efivarfs        256K   22K  235K   9% /sys/firmware/efi/efivars
/dev/nvme0n1p1  512M  160K  512M   1% /boot/efi
tmpfs           393M  2.5M  390M   1% /run/user/1000

┌──(kali㉿kali)-[~]
└─$ df -h | grep nvme0n1p2
/dev/nvme0n1p2   19G   16G  1.3G  93% /

┌──(kali㉿kali)-[~]
└─$ df -h | grep nvme0n1p2 | awk '{print $3}'
16G

┌──(kali㉿kali)-[~]
└─$ df -h | grep nvme0n1p2 | awk '{print $4}'
1.3G
```

FINAL OUTPUT EXAMPLE:



```
Hard Disk Details:
The Hard Disk Size is: 19.9125GB (19912.5MB)
The Hard Disk Free (available) Space is: 165MB
The Hard Disk Used Space is: 17GB
```

Line 71-72: echo displays the hard disk free and used spaces with text message.

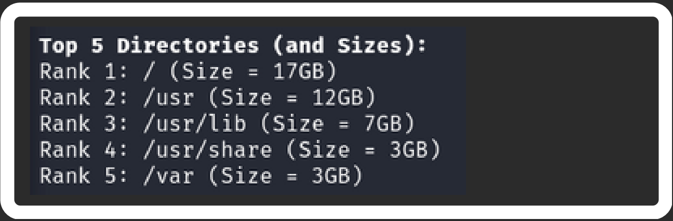
- "B" adds string 'B' for fuller units

Hard disk free spaces and hard disk used spaces are stored in variables harddisk_free & harddisk_used respectively.

DOCUMENTATION

4. DISPLAY TOP 5 DIRECTORIES & SIZES

FINAL OUTPUT EXAMPLE:

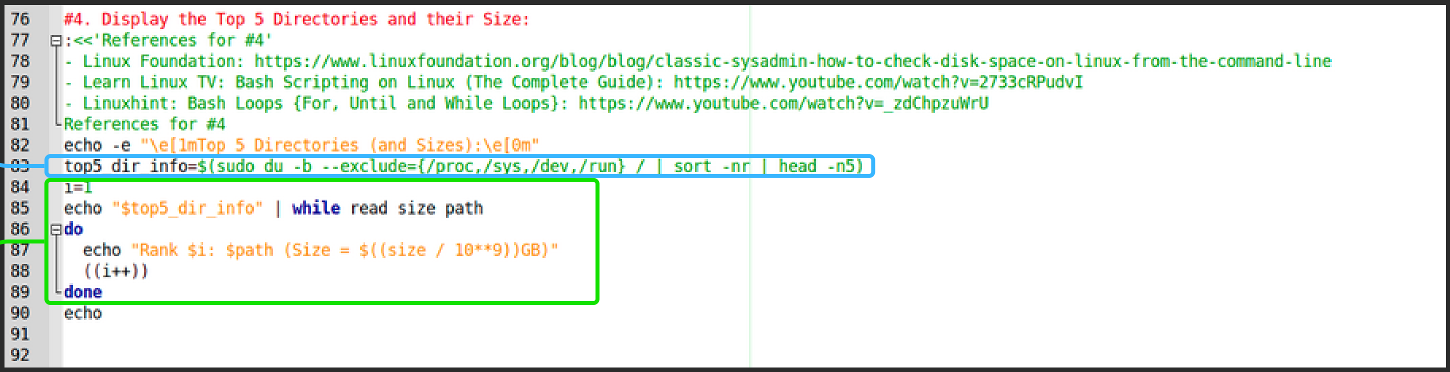


```
Top 5 Directories (and Sizes):
Rank 1: / (Size = 17GB)
Rank 2: /usr (Size = 12GB)
Rank 3: /usr/lib (Size = 7GB)
Rank 4: /usr/share (Size = 3GB)
Rank 5: /var (Size = 3GB)
```

top5_dir_info=$(sudo du -b --exclude={/proc,/sys,/dev,/run} / | sort -nr | head -n5)

- sudo du (a.k.a. Disk Usage) shows space used by all files and directories (displays paths).
  - -b (Bytes): formats the result to display all results in Bytes for easy sorting (next step)
    - *Note to Self: DO NOT use -a (all): as it will show counts for BOTH directories + files*
- --exclude={/proc,/sys,/dev,/run} excludes results from the following directories because:
  - /proc is a virtual filesystem that contains information on running processes, and does not represent actual disk usage.
  - /sys is a virtual filesystem that provides information and control interfaces for kernel and device parameters, and does not represent actual disk usage.
  - /dev contains device files, which are special files used to interact with hardware devices and system components, and are not actual data files that consume disk space.
  - /run contains runtime data and state information (temporary files, sockets etc), and does not represent actual disk usage.
- / specifies the starting directory for the disk space calculation, to be the root directory. This allows the command to calculates the sizes of all directories and subdirectories starting from the root folder.
- sort -nr: sort results by numerical value + reversed (highest sizes on top, lowest sizes at bottom)
- head -n5 displays first 5 rows to filter out the top 5 biggest directory sizes.

The results for top 5 biggest directory sizes are stored into new variable top5_dir_sizes.



```
76  #4. Display the Top 5 Directories and their Size:
77  :<<'References for #4'
78  - Linux Foundation: https://www.linuxfoundation.org/blog/blog/classic-sysadmin-how-to-check-disk-space-on-linux-from-the-command-line
79  - Learn Linux TV: Bash Scripting on Linux (The Complete Guide): https://www.youtube.com/watch?v=2733cRPudvI
80  - Linuxhint: Bash Loops {For, Until and While Loops}: https://www.youtube.com/watch?v=_zdChpzuWrU
81  References for #4
82  echo -e "\e[1mTop 5 Directories (and Sizes):\e[0m"
83  top5_dir_info=$(sudo du -b --exclude={/proc,/sys,/dev,/run} / | sort -nr | head -n5)
84  i=1
85  echo "$top5_dir_info" | while read size path
86  do
87    echo "Rank $i: $path (Size = $((size / 10**9))GB)"
88    ((i++))
89  done
90  echo
91
92
```

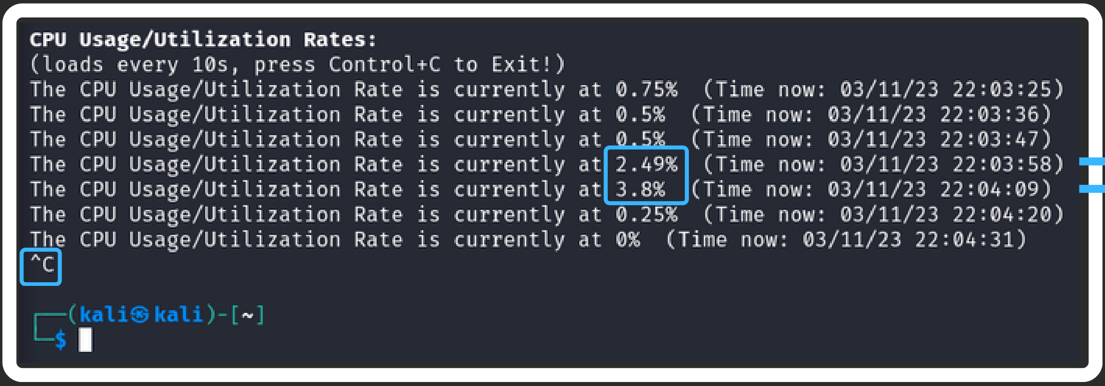
Explaining the 'while' loop (Lines 84-89):

- **Line 84:** i=1 command assigns 1 to the variable i, to increment 'Rank $i' within 'while' loop and literally rank the top 5 directories.
- **Line 85:** echo "$top5_dir_info" displays 5 rows with each 1 row showing 1 directory, and this result (total 5 rows) is piped '|' into while which indicates the start of the 'while' loop.
  - read will grab row-by-row, starting by grabbing the 1st row (out of total 5 rows).
  - From the 1st row, new variables size and path are created:
    - Variable size is stored with 1st row's 1st column data (a.k.a. directory's size in Bytes)
    - Variable path is stored with 1st row's 2nd column data (a.k.a. directory's path)
- **Lines 86-89:** What will be executed as 'while' loop runs?
  - Line 86-87: echo will display 1st row data:
    - text message (string)
    - $i (starts at 1, to display "Rank 1")
    - $path (directory's path)
    - $size (directory's size in GB)
  - Line 88: After displaying 1st row data, increment i++ (otherwise i + 1) will increment i such that the next cycle of while will display as "Rank 2".
- **Line 85 (again):** while loop executes again, because there are still 4 more rows. Cycle repeats, until the 5th (last) row is executed.
- **Line 90:** while loop is terminated.

DOCUMENTATION

5. DISPLAY CPU USAGE (REFRESH EVERY 10S)

FINAL OUTPUT EXAMPLE:



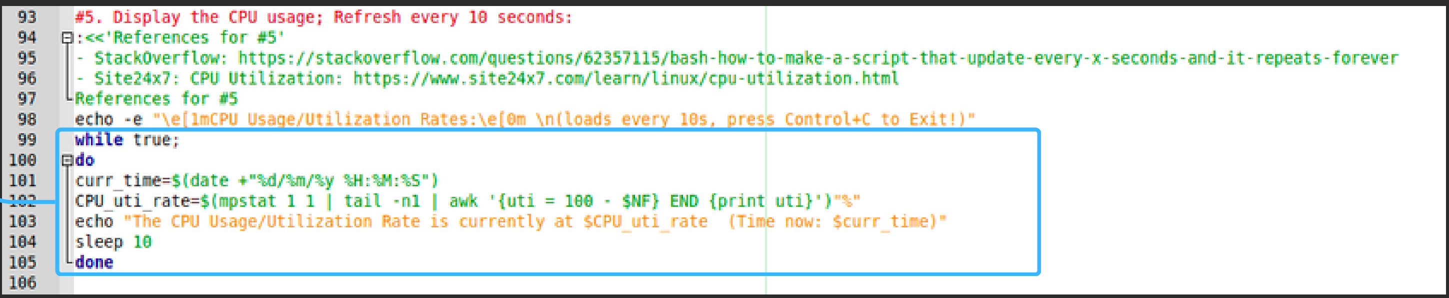
```
CPU Usage/Utilization Rates:
(loads every 10s, press Control+C to Exit!)
The CPU Usage/Utilization Rate is currently at 0.75%  (Time now: 03/11/23 22:03:25)
The CPU Usage/Utilization Rate is currently at 0.5%  (Time now: 03/11/23 22:03:36)
The CPU Usage/Utilization Rate is currently at 0.5%  (Time now: 03/11/23 22:03:47)
The CPU Usage/Utilization Rate is currently at 2.49%  (Time now: 03/11/23 22:03:58)
The CPU Usage/Utilization Rate is currently at 3.8%  (Time now: 03/11/23 22:04:09)
The CPU Usage/Utilization Rate is currently at 0.25%  (Time now: 03/11/23 22:04:20)
The CPU Usage/Utilization Rate is currently at 0%  (Time now: 03/11/23 22:04:31)
^C

┌──(kali㉿kali)-[~]
└─$ 
```

When moved cursor fast.

When scrolled fast up and down.

Explaining the 'while' loop (Lines 99-105):

- **Line 99:** while true starts a infinite loop, where it keeps running until user quits manually.
- **Line 100-105:** do contains commands that will execute within while loop
  - Line 101:
    - date command shows the current (now) date and time.
    - + "%d/%m/%y %H:%M:%S" formats how the date and time is displayed:
      - %d: day
      - %m: month
      - %y: year
      - %H: hours
      - %M: minutes
      - %S: seconds
    - Final formatted current (now) date and time is stored in new variable curr_time.
  - Line 102:
    - mpstat 1 1 collects and displays CPU statistics 1 time, every 1 second.
    - tail -n1 filters the bottom row
    - awk
      - '{uti = 100 - $NF}: $NF filters the last column for %idle, and therefore, new variable uti is stored with 100(%) minus %idle.
      - END {print uti}'): uti (represents CPU Usage Rate) is computed.
    - Final value for CPU Usage Rate is stored in new variable CPU_uti_rate.
  - Line 103:
    - echo
      - prints text message (string)
      - $CPU_uti_rate (CPU Usage Rate)
      - $curr_time (formatted time now)
  - Line 104:
    - sleep 10: meaning that while loop pauses/stops running for 10 seconds
    - After 10 seconds, Line 91-95 runs another cycle, again and again...
  - Line 105:
    - done: indicating termination of while loop
    - In a while true loop, this line never runs. It will only run when user manually exits (eg: Control + C)



```
93  #5. Display the CPU usage; Refresh every 10 seconds:
94  :<<'References for #5'
95  - StackOverflow: https://stackoverflow.com/questions/62357115/bash-how-to-make-a-script-that-update-every-x-seconds-and-it-repeats-forever
96  - Site24x7: CPU Utilization: https://www.site24x7.com/learn/linux/cpu-utilization.html
97  References for #5
98  echo -e "\e[1mCPU Usage/Utilization Rates:\e[0m \n(loads every 10s, press Control+C to Exit!)"
99  while true;
100 do
101 curr_time=$(date +"%d/%m/%y %H:%M:%S")
102 CPU_uti_rate=$(mpstat 1 1 | tail -n1 | awk '{uti = 100 - $NF} END {print uti}')"%"
103 echo "The CPU Usage/Utilization Rate is currently at $CPU_uti_rate  (Time now: $curr_time)"
104 sleep 10
105 done
106
```

END.