Linux Info Dump

▼ Overview of Script Segments

1. Define Shell
2. Initialize Color Shortcuts
3. Prolouge for Users
4. Initialize all Variables and Display Results
5. Display CPU Usage

▼ #1 Define Shell

`#!/bin/bash` - define bash as the Shell to run this script

▼ #2 Initialize Color Shortcuts

The color shortcuts will be used to style outputs using `echo -e` command later in the script.

Reference tool: https://ansi.gabebanks.net/?ref=linuxhandbook.com

```
#------------------------
#Initialize the color shortcuts

DEF='\033[0;m'

BLINKING='\033[5;1m'
UNDERLINE='\033[4m'

CYAN1='\033[1;36m'
GREEN1='\033[1;32m'
ORANGE1='\033[1;33m'
WHITE1='\033[1;1m'
RED1='\033[1;31m'
```

▼ #3 Prolouge for Users

```
#-----------------------
#Prologue for user

echo "This script will display Linux Operating System Information of this machine."
echo "Note: sudo is required to run this script and you may be asked to input your password."
echo -e "\nProcessing your request..."
```

```
bash projectscript1.sh
This script will display Linux Operating System Information of this machine.
Note: sudo is required to run this script and you may be asked to input your password.

Processing your request ...
[sudo] password for kali:
```

This section serves as an introduction to the user about the script and any additional notes.

▼ #4 Initialize all the variables and Display Results

The first section "**Initialize all the Variables**" will run the necessary Shell commands to retrieve required data values and store it into a variable. The variable will be called to display results later.

- Consideration #1 - Some of the commands will print logs during run-time that affects the display format, hence, all the commands are consolidated in this section. This will allow a display of results with clean and standardized format.
- Consideration #2 - Some of the commands will require sudo and the user may be required to input their password to allow the use of sudo. At the start of the script, the `id` command is used to check if the user has sudo group defined before proceeding. If it is not defined, the script will give the user a message and exit the script.

```
checksudo="`id | grep sudo`"
if [ -z "$checksudo" ]
then
    #empty
    echo "ERROR: Request failed!"
    echo "Please make sure the User is under sudo group before running the script."
    exit
else
    #not empty
    topfivedir="`sudo du -ah / | sort -rh | head -5`"
fi
```

The second section "**Display Results**" will call the variables with stored values and print it on the interface using the `echo -e` command.

- The pre-defined color shortcuts are used to add styling.
- To improve user viewing expereince, the `sleep` command is used to print the display in intervals, instead of all at once.
- A header is used to organsie the formatting

```
echo -e "\n----------------------------"
echo -e "${WHITE1}Linux Operating System Information${DEF}"
echo -e "----------------------------\n"
```

```
----------------------------
Linux Operating System Information
----------------------------

Linux Version : Kali GNU/Linux Rolling - 2023.2

Private IP address : 192.168.169.128
Public IP address : 151.192.234.46
Default gateway : 192.168.169.2

Hard Disk Size :  78G
Hard Disk Used Space : 11G
Hard Disk Free Space : 63G

Top 5 Directories and Size:

11G     /
9.2G    /usr
5.2G    /usr/lib
3.5G    /usr/share
1.7G    /usr/lib/x86_64-linux-gnu
```

▼ Display Linux version

```
verlinuxname="`cat /etc/os-release | grep PRETTY_NAME | awk -F '"' '{print $(2)}'`"
verlinuxid="$(cat /etc/os-release | grep VERSION_ID | awk -F '"' '{print $(2)}')"

echo -e "Linux Version : ${CYAN1}$verlinuxname - $verlinuxid${DEF}"
```

```
Linux Version : Kali GNU/Linux Rolling - 2023.2
```

The `cat` command will display `/etc/os-release` path to view the Linux system info. Then, the `grep` command is used to narrow down the relevant rows.

Finally, the `awk` command is used with `-F` option and `"` as a customized delimiter to extract the specific Linux information value. This value is stored

in a variable.

▼ Display the private IP address, public IP address and the default gateway

```
ippublic="$(curl ifconfig.me)"
ipprivate="`ifconfig | grep inet | head -n 1 | awk '{print $(2)}'`"
ipdefaultgateway="`route | grep default | awk '{print $(2)}'`"

sleep 1
echo -e "\nPrivate IP address : ${ORANGE1}$ipprivate${DEF}"
echo -e "Public IP address : ${ORANGE1}$ippublic${DEF}"
echo -e "Default gateway : ${ORANGE1}$ipdefaultgateway${DEF}"
```

```
Private IP address : 192.168.169.128
Public IP address : 151.192.234.46
Default gateway : 192.168.169.2
```

▼ Public IP address

The `curl` command is used to query ifconfig.me site for the public IP address. This command will print logs during run-time, shown below.

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    14  100    14    0     0     55      0 --:--:-- --:--:-- --:--:--    55
```

▼ Private IP address

The `ifconfig` command is used to view the ip address info. Then, `grep` command is used to filter the relevant row with keyword `inet`.

As there may be multiple rows with the same keyword, `head -n 1` command with is used to isolate only the first row showing the private IP address.

Finally, `awk` is used to extract only the private IP addres value to be stored.

```
└─$ ifconfig | grep inet
        inet 192.168.169.128  netmask 255.255.255.0  broadcast 192.168.169.255
        inet6 fe80::20c:29ff:fea7:406c  prefixlen 64  scopeid 0x20<link>
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
```

▼ Default Gateway

The `route` command is used to view the . Then, the `grep default` command is used to filter the relevant row showing Default Gateway. Fianlly `awk` is used to extract only the Default Gateway value to be stored.

```
└─$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         192.168.169.2   0.0.0.0         UG    100    0        0 eth0
192.168.169.0   0.0.0.0         255.255.255.0   U     100    0        0 eth0
```

▼ Display the hard disk size, free and used space

```
harddisksize="`df -h | grep sda | awk '{print $(2)}'`"
harddisksizefree="`df -h | grep sda | awk '{print $(4)}'`"
harddisksizeused="`df -h | grep sda | awk '{print $(3)}'`"
```

```
sleep 1
echo -e "\nHard Disk Size : ${WHITE1} $harddisksize ${DEF}"
echo -e "Hard Disk Used Space : ${RED1}$harddisksizeused${DEF}"
echo -e "Hard Disk Free Space : ${GREEN1}$harddisksizefree${DEF}"
```

```
Hard Disk Size :  78G
Hard Disk Used Space : 11G
Hard Disk Free Space : 63G
```

The `df -h` command is used to view the hard disk information, with option of human-readable format. Then, `grep sda` is used to filter only the main file system. Finally, `awk` is used to extract the columns accordingly for Total Size, Used Space and Available Space to be stored in variables.

```
┌──(kali㉿kali)-[~/Desktop]
└─$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            945M     0  945M   0% /dev
tmpfs           198M  1.2M  197M   1% /run
/dev/sda1        78G   11G   63G  15% /
tmpfs           987M     0  987M   0% /dev/shm
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           198M   72K  198M   1% /run/user/1000
```

▼ Display the top(5) directories and their size

```
36      topfivedir="`sudo du -ah / | sort -rh | head -5`"

74  sleep 1
75  echo -e "\n${UNDERLINE}Top 5 Directories and Size:${DEF}\n\n$topfivedir"
```

```
Top 5 Directories and Size:

11G     /
9.2G    /usr
5.2G    /usr/lib
3.5G    /usr/share
1.7G    /usr/lib/x86_64-linux-gnu
```

The command `sudo du /` is used to query the all the directories in the machine, with customized option `-ah` to include all directories and in human-readabale format. Then, the command `sort` with customized options `-rh` is used to make sure the sorting sequence is from largest to smallest and based on human-readable values. Finally, `head -n 5` command is used to only take the top 5 largest directories.

▼ #5 Display the CPU usage

```
77  #-----------------------------------------------
78  #Display CPU Usage, refreshed every 10 seconds
79  sleep 1
80  echo -e "\n${UNDERLINE}Displaying CPU Usage${DEF}\n(${BLINKING}refreshed every 10 seconds${DEF})"
81  echo -e "\nTo stop running, press CTRL+C\n"
82  sar -u 10
83
```

```
Displaying CPU Usage
(refreshed every 10 seconds)

To stop running, press CTRL+C

Linux 5.15.0-kali3-amd64 (kali)         07/11/2023      _x86_64_        (4 CPU)

07:33:24 AM     CPU     %user     %nice   %system   %iowait    %steal     %idle
07:33:34 AM     all      0.61      0.00      1.46      0.00      0.00     97.93
07:33:44 AM     all      0.20      0.00      0.91      0.03      0.00     98.87
07:33:54 AM     all      0.15      0.00      1.11      0.03      0.00     98.72
07:34:04 AM     all      0.23      0.00      0.88      0.00      0.00     98.89
07:34:14 AM     all      0.28      0.00      0.80      0.03      0.00     98.89
07:34:24 AM     all      0.20      0.00      0.96      0.00      0.00     98.84
```

The command `sar` is used to display Linux CPU usage info and with customized option `-u 10` defining the refresh to run every 10 seconds.