Linux Fundamentals | Project 2: Information Extractor

Muhammad Termidzi Bin Azmi (S27)

CFC190324

Samson Xiao

14 April 2024

Table of contents

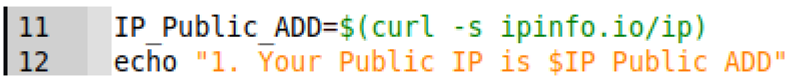
Introduction

This project entails the creation of a Bash script to capture and present various system metrics for a Linux operating system. The script provides insights into the Linux version in use, network details including private and public IP addresses as well as the default gateway. It extends to compute disk statistics, showcasing overall size, free, and used space. The script also highlights the five largest directories and monitors CPU usage, updating these statistics every ten seconds to reflect real-time system state.

This report aims to create a detailed analysis of Operation Info Extractor, where it centers around a straightforward goal: Gather System Info. To achieve this, we've developed an intelligent script via Bash that is capable of delving deep into the system and extracting critical details; mainly network information, system perfomance assessment, and files and folders analysis.

Methodologies

1. Retrieving Public IP Address

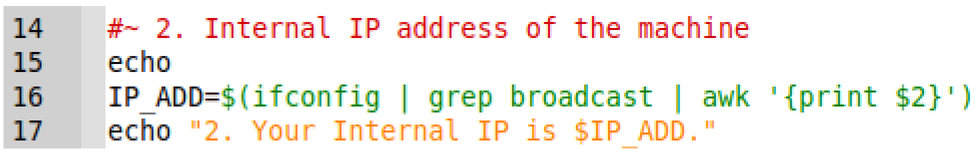


```
IP_Public_ADD=$(curl -s ipinfo.io/ip)
echo "1. Your Public IP is $IP_Public_ADD"
```

Methods used here:

- **curl**: This is a command-line tool used to transfer data from or to a server using various protocols (HTTP, FTP, etc.).
- **-s**: The flag stands for "silent" or "quiet." It suppresses any progress or error messages, making the output cleaner.
- **ipinfo.io/ip**: This is the URL endpoint that provides information about the public IP address of the system where the command is executed. It returns the public IP address associated with your network.

2. Internal IP address of the machine

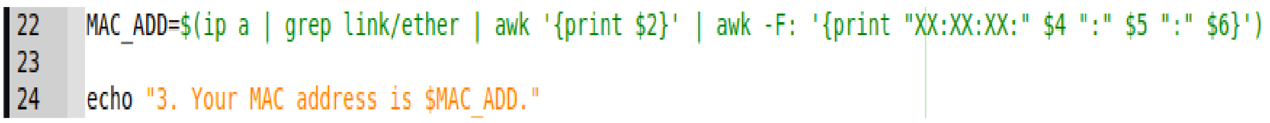


```
#~ 2. Internal IP address of the machine
echo
IP_ADD=$(ifconfig | grep broadcast | awk '{print $2}')
echo "2. Your Internal IP is $IP_ADD."
```

Methods used here:

- **ifconfig:** This command is used to configure network interfaces and display information about network interfaces on a Linux system. It provides details such as IP addresses, netmasks, and broadcast addresses.
- **grep broadcast:** The grep command searches for lines containing the word "broadcast" in the output received from ifconfig. It filters out only the lines related to broadcast addresses.
- **awk '{print $2}':** The awk command processes text data and extracts specific fields. In this case, it prints the second field (which corresponds to the broadcast address) from the filtered output obtained by grep.

3. The MAC address of the machine with half censored.



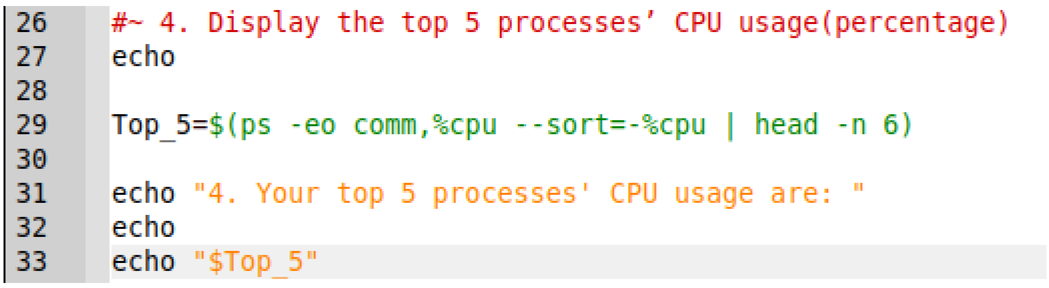
```
MAC_ADD=$(ip a | grep link/ether | awk '{print $2}' | awk -F: '{print "XX:XX:XX:" $4 ":" $5 ":" $6}')

echo "3. Your MAC address is $MAC_ADD."
```

Methods used here:

- **ip a:** This command displays information about network interfaces (IP addresses, MAC addresses, etc.) on a Linux system.

- **grep link/ether:** The grep command searches for lines containing the phrase "link/ether" in the output received from ip a. It filters out only the lines related to MAC addresses (Ethernet addresses).
- **awk '{print $2}':** The first awk command processes text data and extracts the second field (which corresponds to the MAC address) from the filtered output obtained by grep.
- **awk -F: '{print "XX:XX:XX:" $4 ":" $5 ":" $6}':** The second awk command processes the MAC address obtained from the previous step. It splits the MAC address using the colon (:) delimiter and constructs a new MAC address with the format "XX:XX:XX:YY:ZZ:WW," where "XX" represents a placeholder for the original values.

4. Display the top 5 processes' CPU usage by percentage.

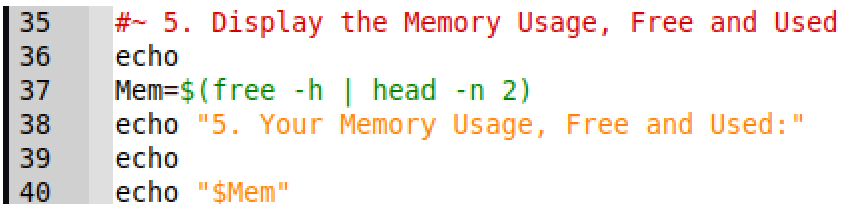


```
#~ 4. Display the top 5 processes' CPU usage(percentage)
echo

Top_5=$(ps -eo comm,%cpu --sort=-%cpu | head -n 6)

echo "4. Your top 5 processes' CPU usage are: "
echo
echo "$Top_5"
```

Methods used here:

- **ps -eo comm,%cpu:** This part of the command retrieves information about running processes.
- **ps:** The ps command stands for "process status" and is used to display information about active processes on a Unix-like operating system.
- **-eo:** The -eo flag specifies the format of the output.
- **comm:** Displays the command name (i.e., the process name).
- **%cpu:** Displays the percentage of CPU usage by each process.
- **--sort=-%cpu:** This part sorts the output based on the CPU usage percentage in descending order (highest to lowest). The - sign before %cpu indicates reverse sorting.
- **head -n 6:** Displays the first six lines of the sorted output. This effectively shows the top six processes with the highest CPU usage.

5. Display the Memory Usage, Free and Used



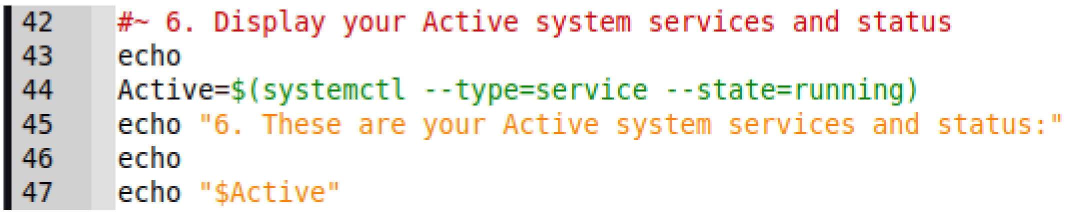
```
#~ 5. Display the Memory Usage, Free and Used
echo
Mem=$(free -h | head -n 2)
echo "5. Your Memory Usage, Free and Used:"
echo
echo "$Mem"
```

Methods used here:

- **free**: This command provides information about memory usage on a Linux system.
- **-h**: The -h flag stands for "human-readable." When used with free, it formats the output in a more readable format, using units like KB, MB, or GB instead of raw bytes.

- **head -n 2**: The head command displays the first few lines of its input. The -n 2 flag specifies that we want to see the first two lines of the output received from free.

6. **Display your Active system services and status.**

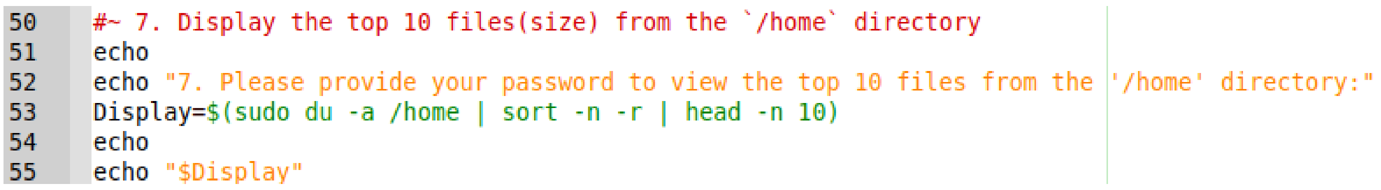


```
#~ 6. Display your Active system services and status
echo
Active=$(systemctl --type=service --state=running)
echo "6. These are your Active system services and status:"
echo
echo "$Active"
```

Methods used here:

- **systemctl**: This command is used to manage services and control the systemd system and service manager on a Linux system.
- **--type=service**: The --type flag specifies the type of unit to operate on. In this case, we're interested in services (background tasks or daemons).
- **--state=running**: The --state flag filters the output based on the state of the services. We want to see only the running services.

7. **Display the top 10 files(size) from the `/home` directory.**



```
#~ 7. Display the top 10 files(size) from the `/home` directory
echo
echo "7. Please provide your password to view the top 10 files from the '/home' directory:"
Display=$(sudo du -a /home | sort -n -r | head -n 10)
echo
echo "$Display"
```

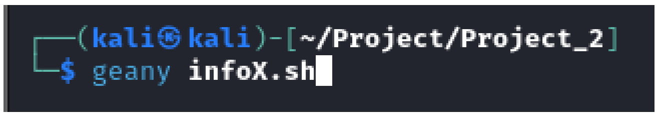
Methods used here:

- **sudo**: The sudo command stands for "superuser do" and is used to execute commands with superuser (root) privileges. It allows you to perform administrative tasks that require elevated permissions.
- **du**: The du command stands for "disk usage" and is used to estimate file and directory space usage on a Unix-like system. It provides information about the size of files and directories.
  - **-a**: The -a flag tells du to display the sizes of individual files and directories, including hidden ones.
- **/home**: This is the path to the directory you want to analyze. In this case, it's the /home directory, which typically contains user home directories.
- **sort -n -r**: The sort command arranges lines of text in a specified order. Here:
  - **-n**: Sorts numerically.
  - **-r**: Sorts in reverse order (largest to smallest).
- **head -n 10**: The head command displays the first few lines of its input. The -n 10 flag specifies that we want to see the first ten lines of the sorted output.

Discussion

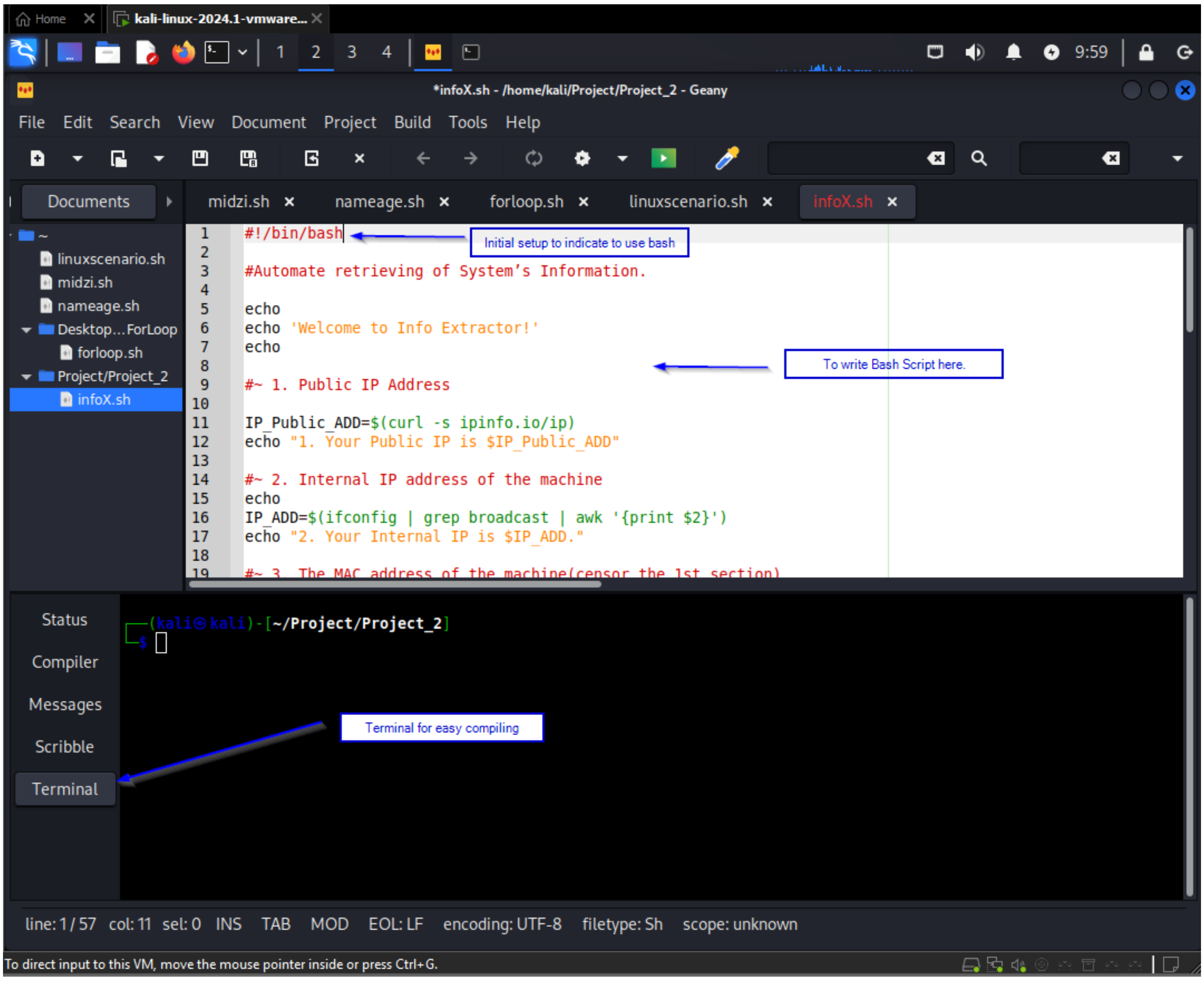
1. **Setting up Geany IDE**

- Using Geany - A free and open-source lightweight GUI text editor, including basic IDE feature.
- Download geany.
- Go into the intended folder to store the file.
- Type in “geany” into the linux command prompt followed by file name ending with .sh.



```
┌──(kali㉿kali)-[~/Project/Project_2]
└─$ geany infoX.sh
```

1. **Geany IDE navigation**



*infoX.sh - /home/kali/Project/Project_2 - Geany
File Edit Search View Document Project Build Tools Help
Documents
midzi.sh nameage.sh forloop.sh linuxscenario.sh infoX.sh
linuxscenario.sh
midzi.sh
nameage.sh
Desktop...ForLoop
forloop.sh
Project/Project_2
infoX.sh
Initial setup to indicate to use bash
To write Bash Script here.
Status
Compiler
Messages
Scribble
Terminal
Terminal for easy compiling

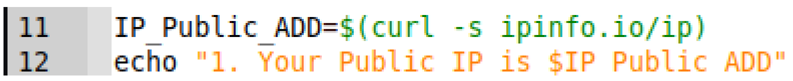
```
#!/bin/bash

#Automate retrieving of System's Information.

echo
echo 'Welcome to Info Extractor!'
echo

#~ 1. Public IP Address

IP_Public_ADD=$(curl -s ipinfo.io/ip)
echo "1. Your Public IP is $IP_Public_ADD"

#~ 2. Internal IP address of the machine
echo
IP_ADD=$(ifconfig | grep broadcast | awk '{print $2}')
echo "2. Your Internal IP is $IP_ADD."

#~ 3. The MAC address of the machine(censor the 1st section)
```

```
┌──(kali㉿kali)-[~/Project/Project_2]
└─$
```

```
line: 1 / 57   col: 11   sel: 0   INS   TAB   MOD   EOL: LF   encoding: UTF-8   filetype: Sh   scope: unknown
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
```

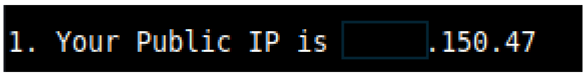
- Run with [bash infoX.sh] on terminal.

2. Retrieving Public IP Address



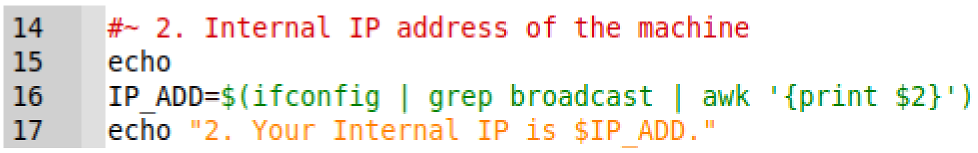
```
IP_Public_ADD=$(curl -s ipinfo.io/ip)
echo "1. Your Public IP is $IP_Public_ADD"
```

- The curl command fetches data from ipinfo.io/ip.
- The -s flag ensures that the output is concise and without any additional messages.
- The result is the public IP address of the system where the command is run.



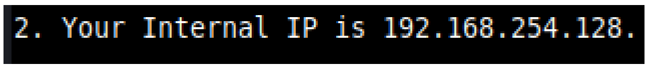
```
1. Your Public IP is [redacted].150.47
```

3. Internal IP address of the machine



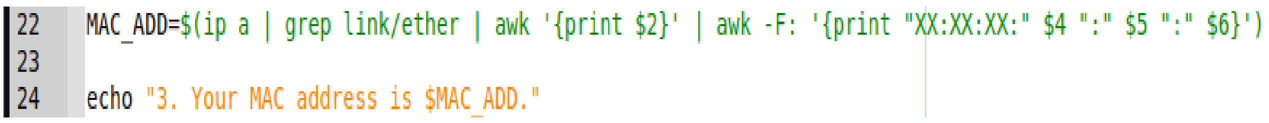
```
#~ 2. Internal IP address of the machine
echo
IP_ADD=$(ifconfig | grep broadcast | awk '{print $2}')
echo "2. Your Internal IP is $IP_ADD."
```

- The ifconfig command fetches information about network interfaces.
- The grep broadcast filters out lines containing the word “broadcast.”
- The awk '{print $2}' extracts the broadcast address from the filtered output.



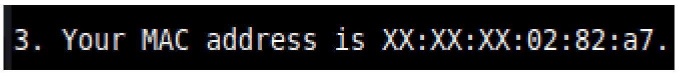
```
2. Your Internal IP is 192.168.254.128.
```

4. The MAC address of the machine with half censored.



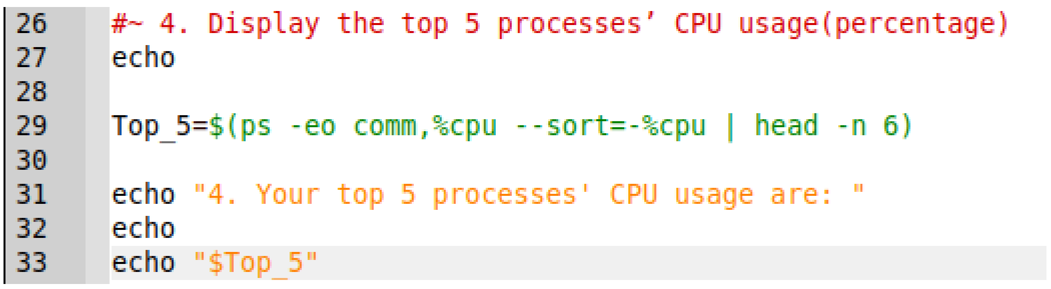
```
MAC_ADD=$(ip a | grep link/ether | awk '{print $2}' | awk -F: '{print "XX:XX:XX:" $4 ":" $5 ":" $6}')

echo "3. Your MAC address is $MAC_ADD."
```

- The ip a command fetches information about network interfaces.
- The grep link/ether filters out lines containing MAC addresses.
- The first awk extracts the MAC address.
- The second awk constructs a new MAC address with placeholders.
- The result will be a modified MAC address with the format “XX:XX:XX:YY:ZZ:WW.”

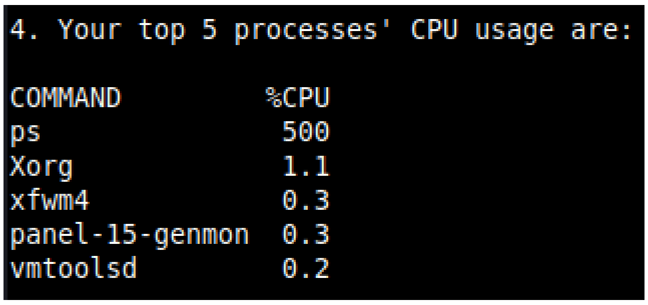


```
3. Your MAC address is XX:XX:XX:02:82:a7.
```

5. Display the top 5 processes’ CPU usage by percentage.

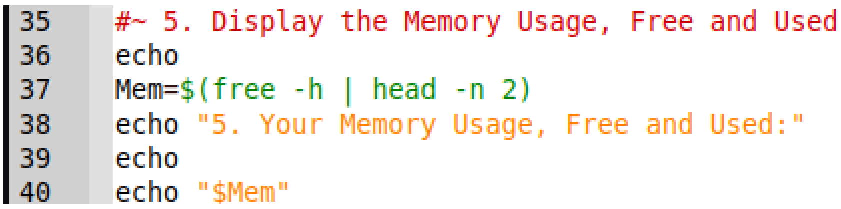


```
#~ 4. Display the top 5 processes’ CPU usage(percentage)
echo

Top_5=$(ps -eo comm,%cpu --sort=-%cpu | head -n 6)

echo "4. Your top 5 processes' CPU usage are: "
echo
echo "$Top_5"
```



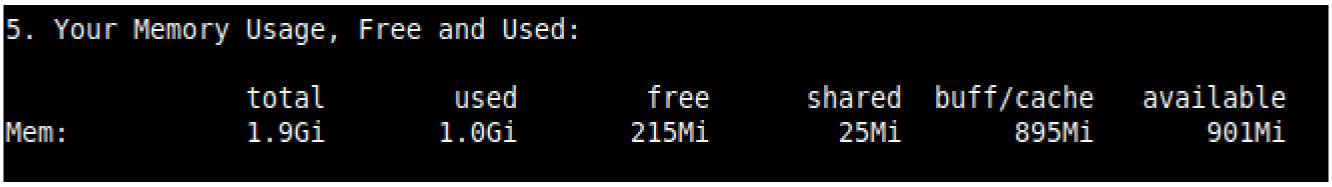
```
4. Your top 5 processes' CPU usage are:

COMMAND         %CPU
ps               500
Xorg             1.1
xfwm4            0.3
panel-15-genmon  0.3
vmtoolsd         0.2
```

6. **Display the Memory Usage, Free and Used.**



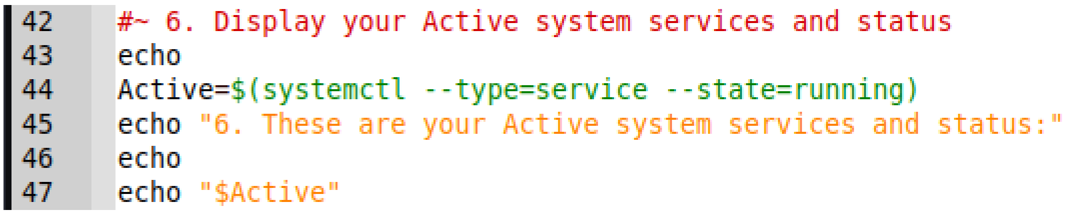
```
#~ 5. Display the Memory Usage, Free and Used
echo
Mem=$(free -h | head -n 2)
echo "5. Your Memory Usage, Free and Used:"
echo
echo "$Mem"
```

- The free command fetches information about memory usage.
- The -h flag formats the output in a human-readable format.
- The head -n 2 command displays the first two lines of the formatted output.



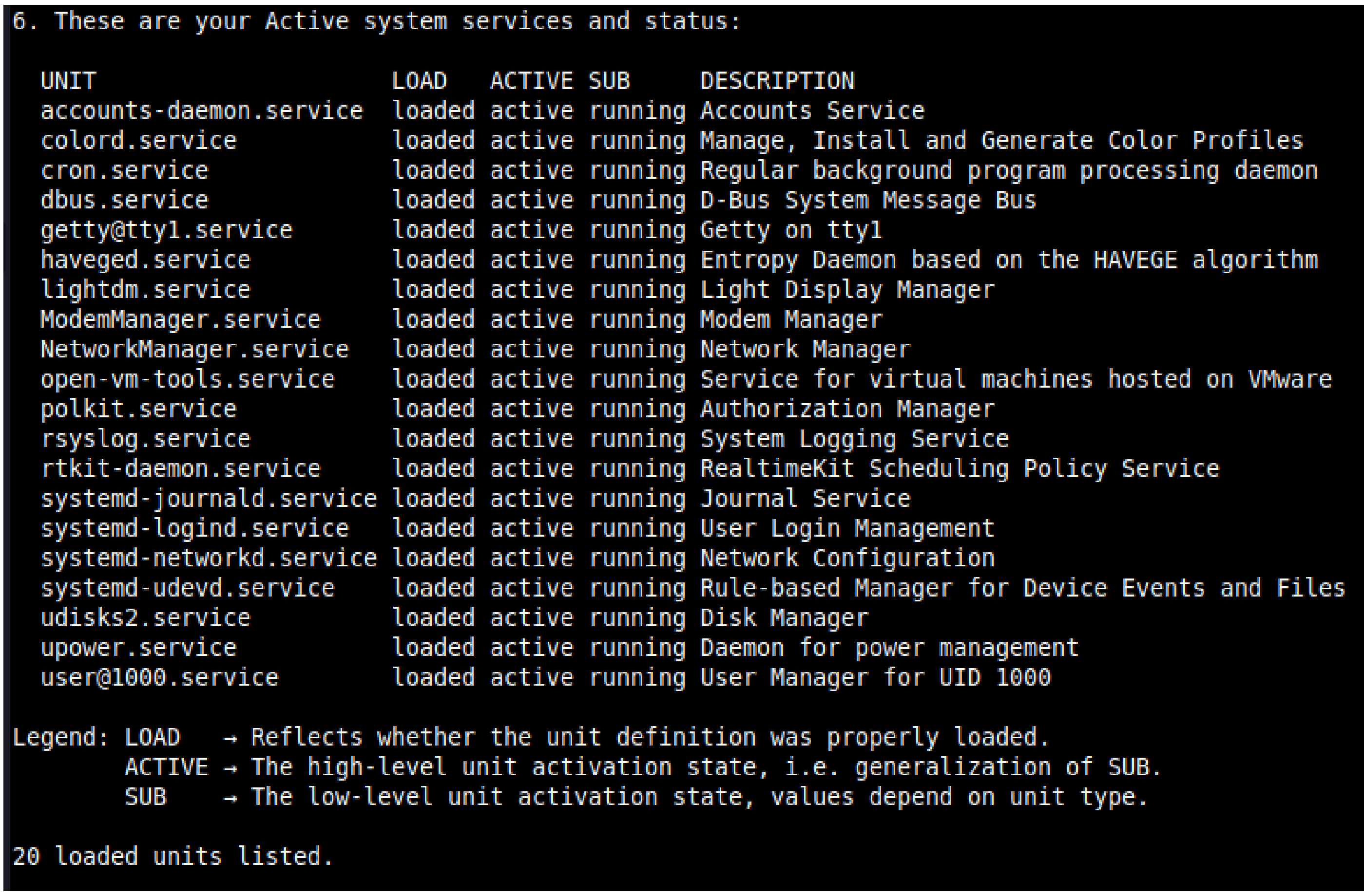
```
5. Your Memory Usage, Free and Used:

               total        used        free      shared  buff/cache   available
Mem:           1.9Gi       1.0Gi       215Mi        25Mi       895Mi       901Mi
```

7. **Display your Active system services and status.**



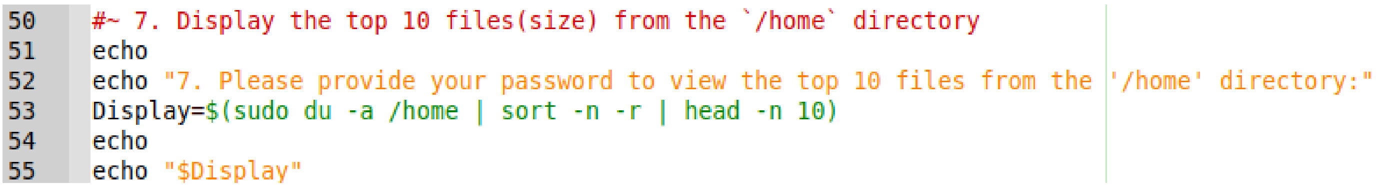
```
#~ 6. Display your Active system services and status
echo
Active=$(systemctl --type=service --state=running)
echo "6. These are your Active system services and status:"
echo
echo "$Active"
```

- The systemctl command fetches information about services managed by systemd.
- The --type=service flag ensures that we're specifically looking at services.
- The --state=running flag filters the output to show only running services.



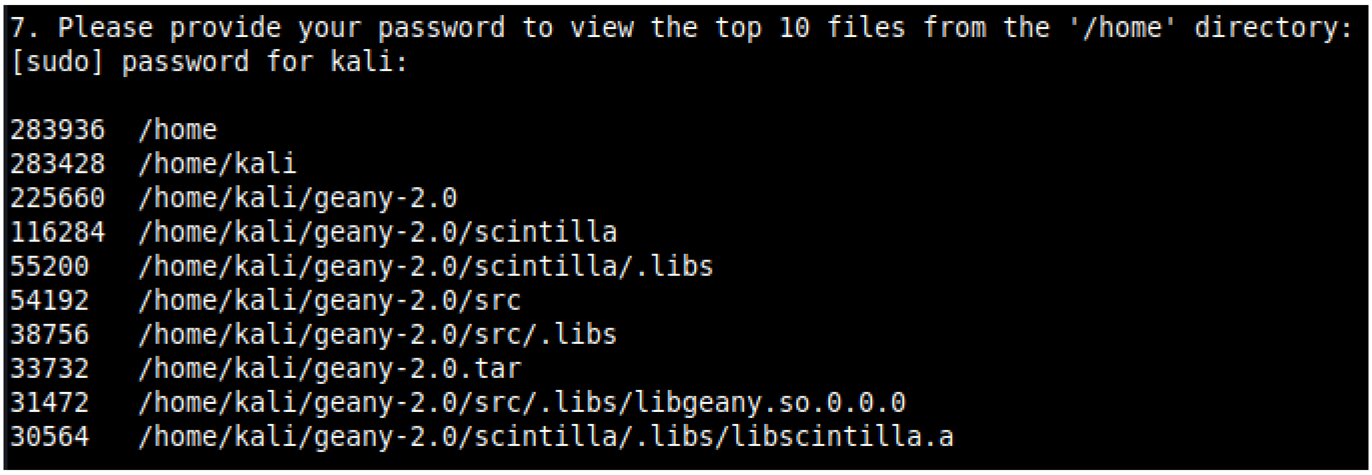
```
6. These are your Active system services and status:

  UNIT                         LOAD   ACTIVE SUB     DESCRIPTION
  accounts-daemon.service      loaded active running Accounts Service
  colord.service               loaded active running Manage, Install and Generate Color Profiles
  cron.service                 loaded active running Regular background program processing daemon
  dbus.service                 loaded active running D-Bus System Message Bus
  getty@tty1.service           loaded active running Getty on tty1
  haveged.service              loaded active running Entropy Daemon based on the HAVEGE algorithm
  lightdm.service              loaded active running Light Display Manager
  ModemManager.service         loaded active running Modem Manager
  NetworkManager.service       loaded active running Network Manager
  open-vm-tools.service        loaded active running Service for virtual machines hosted on VMware
  polkit.service               loaded active running Authorization Manager
  rsyslog.service              loaded active running System Logging Service
  rtkit-daemon.service         loaded active running RealtimeKit Scheduling Policy Service
  systemd-journald.service     loaded active running Journal Service
  systemd-logind.service       loaded active running User Login Management
  systemd-networkd.service     loaded active running Network Configuration
  systemd-udevd.service        loaded active running Rule-based Manager for Device Events and Files
  udisks2.service              loaded active running Disk Manager
  upower.service               loaded active running Daemon for power management
  user@1000.service            loaded active running User Manager for UID 1000

Legend: LOAD   → Reflects whether the unit definition was properly loaded.
        ACTIVE → The high-level unit activation state, i.e. generalization of SUB.
        SUB    → The low-level unit activation state, values depend on unit type.

20 loaded units listed.
```

8. **Display the top 10 files(size) from the `/home` directory.**



```
#~ 7. Display the top 10 files(size) from the `/home` directory
echo
echo "7. Please provide your password to view the top 10 files from the '/home' directory:"
Display=$(sudo du -a /home | sort -n -r | head -n 10)
echo
echo "$Display"
```

- The sudo du -a /home command calculates the disk usage of files and directories in the /home directory.
- The sort -n -r command sorts the output numerically in reverse order (largest to smallest).
- The head -n 10 command displays the first ten lines of the sorted output.



```
7. Please provide your password to view the top 10 files from the '/home' directory:
[sudo] password for kali:

283936  /home
283428  /home/kali
225660  /home/kali/geany-2.0
116284  /home/kali/geany-2.0/scintilla
55200   /home/kali/geany-2.0/scintilla/.libs
54192   /home/kali/geany-2.0/src
38756   /home/kali/geany-2.0/src/.libs
33732   /home/kali/geany-2.0.tar
31472   /home/kali/geany-2.0/src/.libs/libgeany.so.0.0.0
30564   /home/kali/geany-2.0/scintilla/.libs/libscintilla.a
```

Conclusion

In this Information Extractor project, we delved into various aspects of our system configuration and performance. Here are the key findings and methodologies employed:

1. **Public IP Address:**

Successfully retrieved the public IP address using the curl command and the ipinfo.io/ip URL endpoint. This information is crucial for network troubleshooting and external communication.

2. **Private IP Address (Internal IP):**

By analyzing network interfaces with the ifconfig command, we identified the private IP address assigned to the system's network interface. Understanding internal network addressing aids in local communication and network management.

3. **MAC Address (Partially Censored):**

The ip a command revealed the MAC address (Ethernet address) of the machine. For security reasons, we partially censored the MAC address by replacing part of it with placeholders.

4. **CPU Usage:**

We displayed the percentage of CPU usage for the top 5 processes using the ps command. Monitoring CPU utilization helps optimize system performance and resource allocation.

5. **Memory Usage Statistics:**

The free command provided details about total and available memory. Understanding memory usage aids in efficient memory management.

6. **Active System Services:**

We listed active system services and their statuses using the systemctl command. Monitoring services ensures smooth system operation and timely troubleshooting.

7. **Top 10 Largest Files in /home:**

By calculating disk usage with the du command and sorting the results, we identified the largest files in the /home directory. Locating large files assists in storage optimization and cleanup.

Recommendations

1. Leverage bash scripting to automate routine tasks and gather system information. Create custom scripts for tasks like retrieving IP addresses, monitoring resource usage, or managing services.

2. Ensure that any scripts or tools used for diagnostics do not inadvertently expose sensitive data.

3. Use the identified largest files (from /home) to periodically clean up unnecessary or large files. Regularly check for unused services and disable or remove any that are not essential.

4. These diagnostics empower us to maintain a healthy and efficient system, troubleshoot issues, and make informed decisions regarding resource allocation and system management.

5. Use cron jobs or systemd timers to schedule script execution at specific intervals.

References

1. 10+ commands to list all systemctl services with status
   https://www.golinuxcloud.com/systemctl-list-services/

2. How To Find My Public IP Address from Linux Command Line - TecAdmin
   https://tecadmin.net/5-commands-to-get-public-ip-using-linux-terminal/

3. How to obtain MAC address in Linux - Linux Tutorials - Learn Linux Configuration
   https://linuxconfig.org/how-to-obtain-mac-address-in-linux

4. Show top five CPU consuming processes with `ps` - Unix & Linux Stack Exchange
   https://unix.stackexchange.com/questions/13968/show-top-five-cpu-consuming-processes-with-ps

5. How to Use the free Command on Linux (howtogeek.com)
   https://www.howtogeek.com/456943/how-to-use-the-free-command-on-linux/

6. How To Find Largest Top 10 Files and Directories On Linux / UNIX / BSD - nixCraft (cyberciti.biz)
   https://www.cyberciti.biz/faq/how-do-i-find-the-largest-filesdirectories-on-a-linuxunixbsd-filesystem/How To Find Largest Top 10 Files and Directories On Linux / UNIX / BSD - nixCraft (cyberciti.biz)