Info Extractor

Linux Fundamentals

Centre For Cybersecurity

Edwin Lum
S8
CFC130124

TABLE OF CONTENTS

Introduction

With the exponential growth of digital infrastructure and increasing cyberattacks around the world, there is a need for a secure and efficient tool to extract system information. The primary objective of this project was to develop a Bash script to automate the process of gathering essential system information swiftly and efficiently.

This report introduces the functionality and purpose of the Bash script, showing its capabilities in extracting vital system information from a user's machine. The script encompasses a comprehensive set of functionalities, ranging from the retrieval of public and internal IP addresses to analysing system processes and file sizes within specified directories.

The development of this Bash script serves as a practical exercise in automating routine system monitoring tasks, thereby enhancing operational efficiency and bolstering the security posture of an organisation. By streamlining the process of information retrieval, users can proactively identify anomalies, potential security threats, and optimise system performance.

BASH

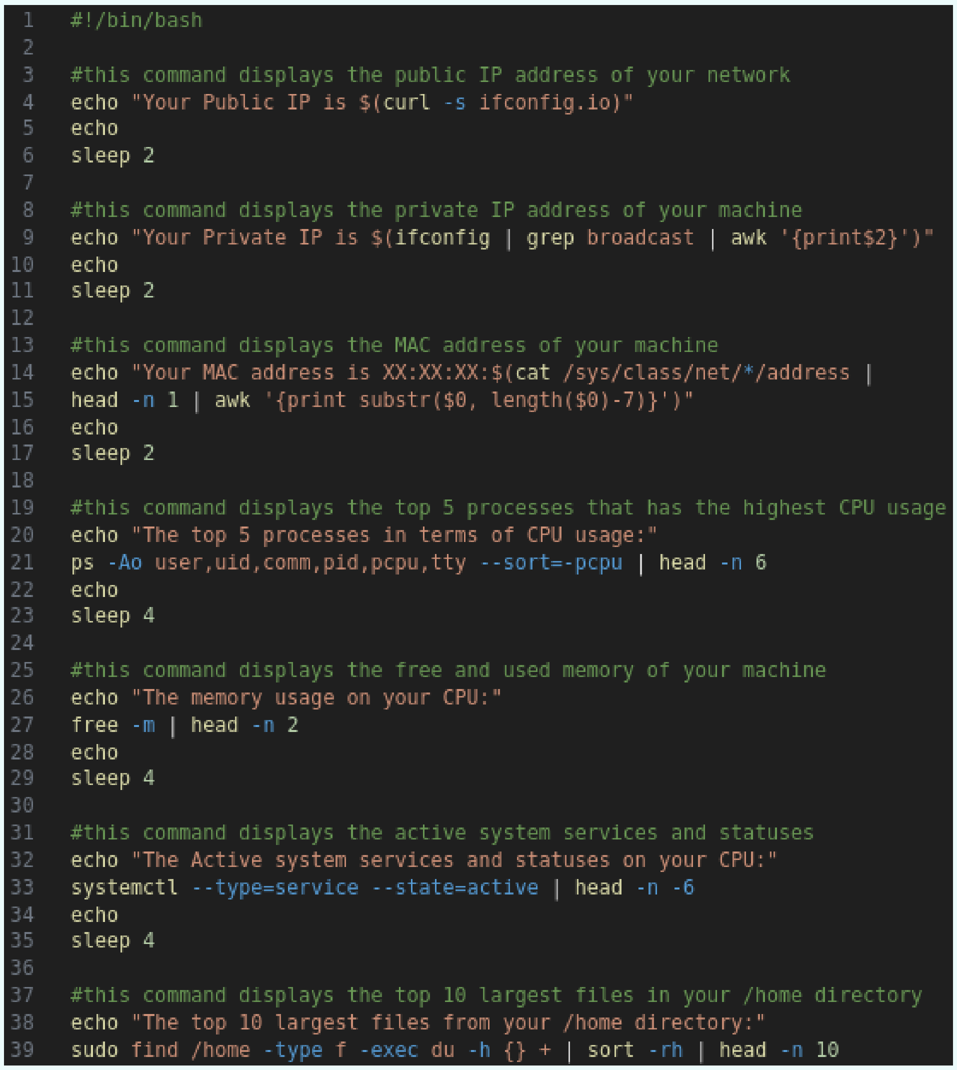
THE BOURNE-AGAIN SHELL

Methodologies

Bash Script

This Bash script was written to help users to automatically retrieve the following system information:

1. Public IP Address
2. Internal IP Address of the Machine
3. MAC Address of the Machine
4. Top 5 processes in terms of CPU usage
5. Memory Usage on the CPU
6. Active System Services and Statuses
7. Top 10 Largest Files from the /home Directory



```bash
#!/bin/bash

#this command displays the public IP address of your network
echo "Your Public IP is $(curl -s ifconfig.io)"
echo
sleep 2

#this command displays the private IP address of your machine
echo "Your Private IP is $(ifconfig | grep broadcast | awk '{print$2}')"
echo
sleep 2

#this command displays the MAC address of your machine
echo "Your MAC address is XX:XX:XX:$(cat /sys/class/net/*/address |
head -n 1 | awk '{print substr($0, length($0)-7)}')"
echo
sleep 2

#this command displays the top 5 processes that has the highest CPU usage
echo "The top 5 processes in terms of CPU usage:"
ps -Ao user,uid,comm,pid,pcpu,tty --sort=-pcpu | head -n 6
echo
sleep 4

#this command displays the free and used memory of your machine
echo "The memory usage on your CPU:"
free -m | head -n 2
echo
sleep 4

#this command displays the active system services and statuses
echo "The Active system services and statuses on your CPU:"
systemctl --type=service --state=active | head -n -6
echo
sleep 4

#this command displays the top 10 largest files in your /home directory
echo "The top 10 largest files from your /home directory:"
sudo find /home -type f -exec du -h {} + | sort -rh | head -n 10
```

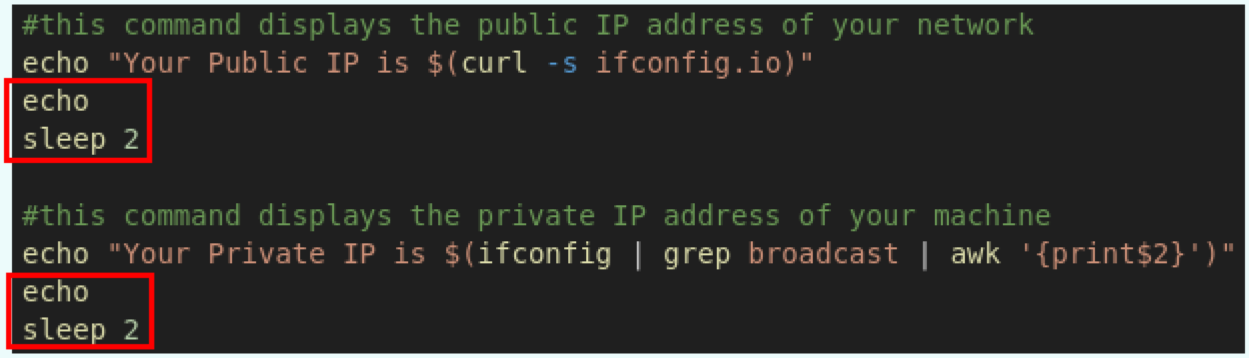
Figure 1: Bash Script for Info Extractor

The echo command was widely used in the script to inform the user which system information is being displayed before running the commands to display the information.

In general, one to two commands were written to clearly display each of the respective system information to the user.

The echo command used between each piece of system information displays an extra empty line, allowing the script to be visually clearer when run.

The sleep command gives the user time to read each line displayed when the script is run.



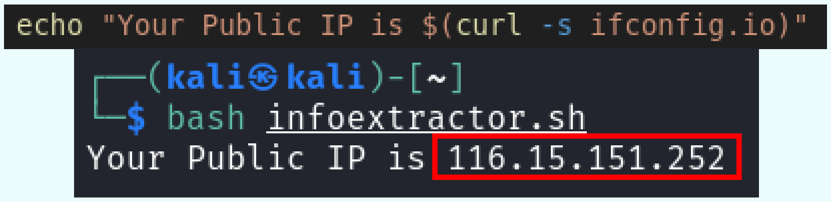
```
#this command displays the public IP address of your network
echo "Your Public IP is $(curl -s ifconfig.io)"
echo
sleep 2

#this command displays the private IP address of your machine
echo "Your Private IP is $(ifconfig | grep broadcast | awk '{print$2}')"
echo
sleep 2
```

Figure 2: echo and sleep commands

Public IP Address

The command, curl -s ifconfig.io, was used to display the public IP address of the network.

The -s flag ensures that the command operates in silent mode, removing headers that would otherwise show in the command-line.



```
echo "Your Public IP is $(curl -s ifconfig.io)"
```

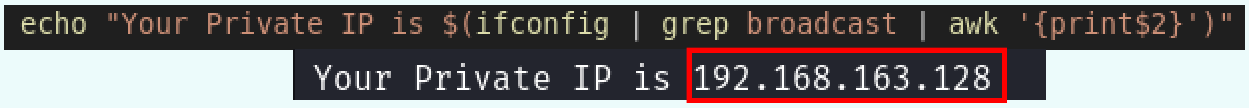
```
┌──(kali㉿kali)-[~]
└─$ bash infoextractor.sh
Your Public IP is 116.15.151.252
```

Figure 3: curl command used to extract external IP address

Internal IP Address

The command, ifconfig | grep broadcast | awk '{print$2}', was used to display the internal IP address of the system.

The grep command was used to filter for the specific line with the IP address and the awk command filtered for the column with the internal IP address.



```
echo "Your Private IP is $(ifconfig | grep broadcast | awk '{print$2}')"
```

```
Your Private IP is 192.168.163.128
```

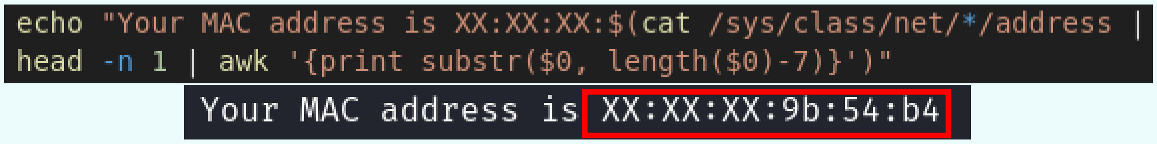
Figure 4: ifconfig command used to extract internal IP address

MAC Address

cat /sys/class/net/*/address | head -n 1 | awk '{ print substr($0, length($0)-7) }' was the command used to read the address file, in the relevant directory that contains the MAC address of the system.

The head command only displayed the first line containing the MAC address.

The awk command was used to display the last eight characters of the MAC address, allowing the command to hide the manufacturer in the MAC address.



```
echo "Your MAC address is XX:XX:XX:$(cat /sys/class/net/*/address |
head -n 1 | awk '{print substr($0, length($0)-7)}')"
```

```
Your MAC address is XX:XX:XX:9b:54:b4
```

Figure 5: cat command used to extract MAC address of the machine

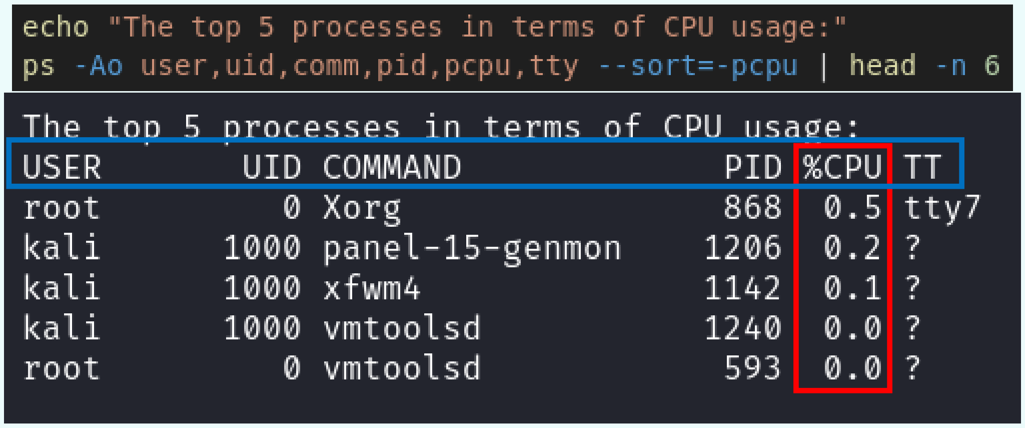
Top 5 processes in terms of CPU usage

The command, ps -Ao user,uid,comm,pid,pcpu,tty --sort=-pcpu | head -n 6, was used to display the top 5 processes in terms of CPU usage.

The -Ao flags allow the user to define which columns to be displayed.

The --sort flag sorts the processes by % CPU usage.

The head command shows the top 5 processes, including the header.



```
echo "The top 5 processes in terms of CPU usage:"
ps -Ao user,uid,comm,pid,pcpu,tty --sort=-pcpu | head -n 6
```

```
The top 5 processes in terms of CPU usage:
USER        UID COMMAND             PID %CPU TT
root          0 Xorg                868  0.5 tty7
kali       1000 panel-15-genmon    1206  0.2 ?
kali       1000 xfwm4              1142  0.1 ?
kali       1000 vmtoolsd           1240  0.0 ?
root          0 vmtoolsd            593  0.0 ?
```

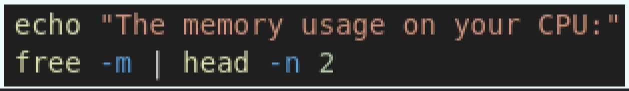
Figure 6: ps command used to extract and sort top 5 processes by % CPU usage

Memory Usage on the CPU

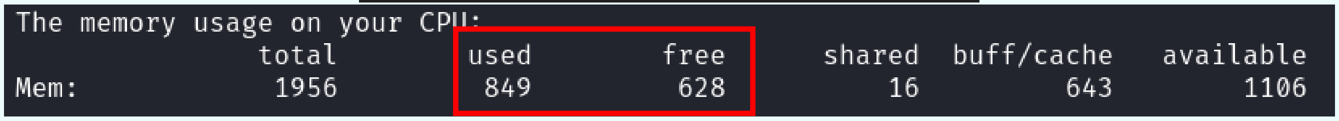
free -m | head -n 2 was the command used to display free and used memory.

The -m flag was used to display the memory in MB.

The head command allowed for the display of only the first 2 lines, showing the headers and the values for memory usage.



```
echo "The memory usage on your CPU:"
free -m | head -n 2
```



```
The memory usage on your CPU:
               total        used        free      shared  buff/cache   available
Mem:            1956         849         628          16         643        1106
```

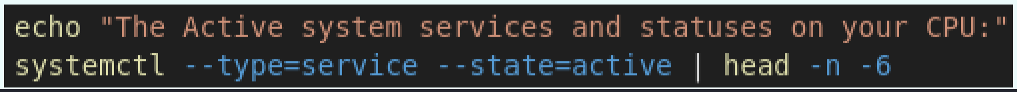
Figure 7: free command used to display memory usage in MB

Active System Services and Statuses

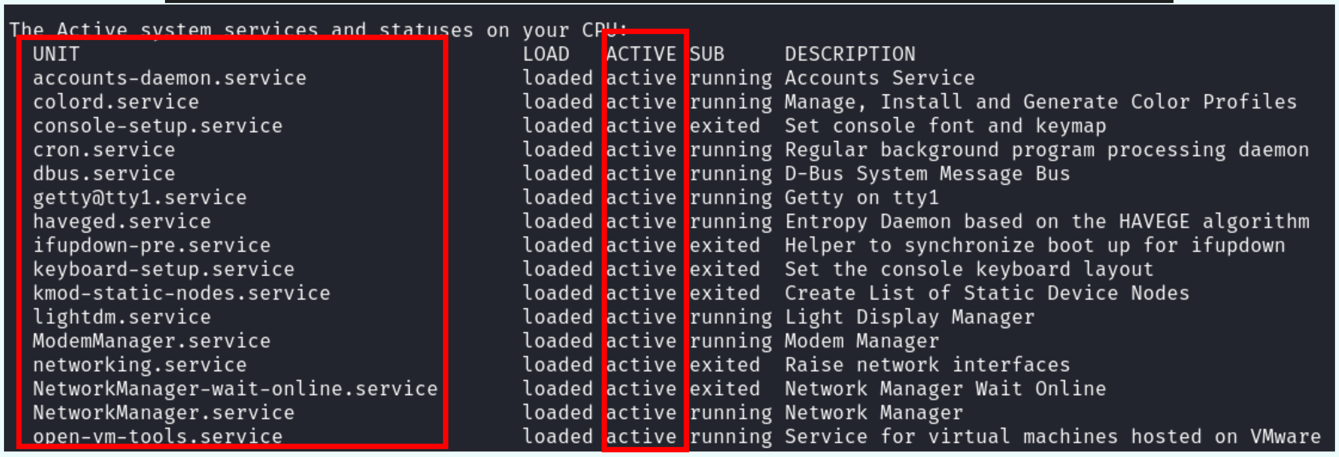
The command, systemctl --type=service --state=active | head -n -6, was used to display a list of all loaded system units, services and showing their statuses.

The --type flag filtered for services and the --state flag filtered for the active services.

The head command was used to show all the information except for the last 6 lines that do not show any of the required information.



```
echo "The Active system services and statuses on your CPU:"
systemctl --type=service --state=active | head -n -6
```



```
The Active system services and statuses on your CPU:
  UNIT                                 LOAD   ACTIVE SUB     DESCRIPTION
  accounts-daemon.service              loaded active running Accounts Service
  colord.service                       loaded active running Manage, Install and Generate Color Profiles
  console-setup.service                loaded active exited  Set console font and keymap
  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
  ifupdown-pre.service                 loaded active exited  Helper to synchronize boot up for ifupdown
  keyboard-setup.service               loaded active exited  Set the console keyboard layout
  kmod-static-nodes.service            loaded active exited  Create List of Static Device Nodes
  lightdm.service                      loaded active running Light Display Manager
  ModemManager.service                 loaded active running Modem Manager
  networking.service                   loaded active exited  Raise network interfaces
  NetworkManager-wait-online.service   loaded active exited  Network Manager Wait Online
  NetworkManager.service               loaded active running Network Manager
  open-vm-tools.service                loaded active running Service for virtual machines hosted on VMware
```

Figure 8: systemctl command used to display active services and statuses

Top 10 Largest Files from the /home Directory

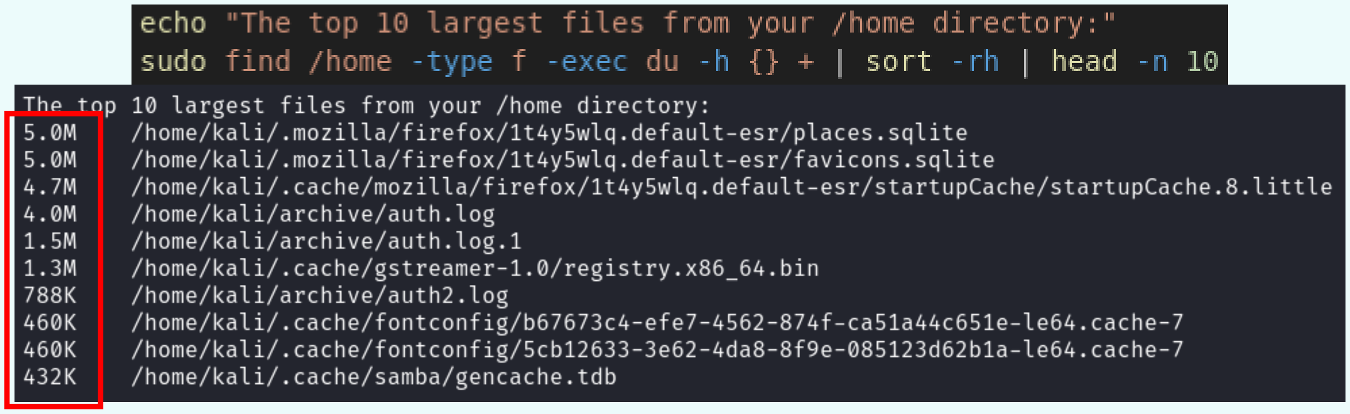
The command, sudo find /home -type f -exec du -h {} + | sort -rh | head -n 10, was used to find the 10 largest files from the /home directory.

sudo was used to give the current user sufficient privileges to search for files in directories that could require escalated privileges.

The -type flag was used to search for files.

The du command was used to estimate the space usage of files, while the -h flag displays the size in human-readable format. The -exec, along with {}, expands the du command to the filename of each file found with the find command.

The information found were then sorted and parsed using the sort command and head command respectively, to show only the 10 largest files.



```
echo "The top 10 largest files from your /home directory:"
sudo find /home -type f -exec du -h {} + | sort -rh | head -n 10
```

```
The top 10 largest files from your /home directory:
5.0M    /home/kali/.mozilla/firefox/1t4y5wlq.default-esr/places.sqlite
5.0M    /home/kali/.mozilla/firefox/1t4y5wlq.default-esr/favicons.sqlite
4.7M    /home/kali/.cache/mozilla/firefox/1t4y5wlq.default-esr/startupCache/startupCache.8.little
4.0M    /home/kali/archive/auth.log
1.5M    /home/kali/archive/auth.log.1
1.3M    /home/kali/.cache/gstreamer-1.0/registry.x86_64.bin
788K    /home/kali/archive/auth2.log
460K    /home/kali/.cache/fontconfig/b67673c4-efe7-4562-874f-ca51a44c651e-le64.cache-7
460K    /home/kali/.cache/fontconfig/5cb12633-3e62-4da8-8f9e-085123d62b1a-le64.cache-7
432K    /home/kali/.cache/samba/gencache.tdb
```

Figure 9: find command used to display the 10 largest files

Discussion

There are many different approaches and commands available to extract the required system information using the Linux command-line.

Open-source intelligence (OSINT) was used to aid in the optimisation of commands to extract the system information and present them clearly for users who run the Bash script.

An alternative method to find the MAC address of the machine was to use the command, ifconfig, then the grep command to find the line with ether, and the awk command to isolate the column with the MAC address.

The top command could also be used to extract the CPU usage of the processes. As the top command shows a real-time view of running processes, with an interactive command mode, it was difficult to incorporate it into an automated script. An alternative of using the ps command was used to show the CPU usage of processes instead.

To check for memory use, the commands vmstat and cat /proc/meminfo were also taken into consideration. However, the meminfo file does not display the used memory and the free command displays the information in a clearer manner compared to vmstat.

Conclusion

One of the advantages of using a Bash script is to automate the process of gathering critical system information. By eliminating manual intervention, this script minimises human errors, improving the reliability and accuracy of the information retrieved. The automated script also streamlines the information retrieval process as users only need to run the script, hence enhancing operational efficiency as well.

This script also helps enhance the user’s situational awareness on cybersecurity, with information on the CPU and memory usage, enabling proactive identification of potential issues or security threats. This empowers users to potentially detect signs of malware when their CPU and memory usage becomes abnormally high.

In conclusion, the development of this Bash script underscores the importance of automation in cybersecurity, providing users with a versatile tool for system information retrieval to optimise resource utilisation of their machines. By harnessing the power of automation, users also play a pivotal role in the cybersecurity posture of the organisation.

Recommendations

OSINT was used to find errors in the commands used in the project and sources to better understand the use of certain flags. It is important to ensure that OSINT is used responsibly to prevent any loss of confidential information from the organisation.

The Bash script could be made more modular with the use of functions, to reduce clutter when the script is run and allow users to extract only the information they need.

The script was designed with simplicity in mind to empower users to play their roles in the cybersecurity posture of the organisation. To better cater to more advanced users, the script can be modified to allow users to input parameters to customise the output according to their preferences. For example, users could be allowed to input the directory they would like to find the 10 largest files or to search for their desired active system service.

This would however, generate an additional security concern and input validation would be required to prevent potential code injection attacks.

Providing training on the usage of the script will help users to use the script more effectively.

Updating Bash to the latest version would further harden the security of the script.

Using the set -e option will terminate the script if any command in the script fails, allowing the identifying and fixing of errors that could lead to security vulnerabilities.

Using variables for commands will ensure easier update of the commands in the future. By avoiding the need to find and modify commands in multiple places in the script, it helps to prevent errors and vulnerabilities that can arise from executing commands with user input or untrusted data.

Restricting permissions on the script will also provide a layer of access control, preventing unauthorised editing of the script.

Using the logger utility can also allow execution of the script to be logged into the syslog file, which allows for threat detection and digital forensics.

References

1. “Logo.” Bash Logo Media Assets, bashlogo.com/. Accessed 8 Feb. 2024.

2. Azar, Jimmy. “How to Extract the Last N Characters of a String in Bash.” Baeldung on Linux, 13 Sept. 2023, www.baeldung.com/linux/bash-extract-last-n-characters.

3. PlanasB, et al. “Show Top Five CPU Consuming Processes with `PS`.” Unix & Linux Stack Exchange, unix.stackexchange.com/questions/13968/show-top-five-cpu-consuming-processes-with-ps. Accessed 7 Feb. 2024.

4. Kaplarevic, Vladimir. “5 Commands to Check Memory Usage in Linux {easy Way}.” Knowledge Base by phoenixNAP, 5 Dec. 2023, phoenixnap.com/kb/linux-commands-check-memory-usage.

5. Kili, Aaron. “Aaron Kili.” How to List All Running Services Under Systemd in Linux, 16 Nov. 2023, www.tecmint.com/list-all-running-services-under-systemd-in-linux/.

6. msdeep14. “Du Command in Linux with Examples.” GeeksforGeeks, GeeksforGeeks, 14 Dec. 2023, www.geeksforgeeks.org/du-command-linux-examples/.

7. codeofnode, et al. “What Does ‘{} \;’ Mean in the Find Command?” Ask Ubuntu, askubuntu.com/questions/339015/what-does-mean-in-the-find-command#:~:text=If%20you%20run%20find%20with,the%20command%20executed%20by%20exec%20. Accessed 7 Feb. 2024.

8. “How to Fix High CPU Usage.” Intel, www.intel.com/content/www/us/en/gaming/resources/how-to-fix-high-cpu-usage.html. Accessed 8 Feb. 2024.

9. Kumar, Arun. “Securing Your Bash Scripts: Essential Security Tips.” FOSS Linux, FOSS Linux, 20 Feb. 2023, www.fosslinux.com/101589/bash-security-tips-securing-your-scripts-and-preventing-vulnerabilities.htm#:~:text=However%2C%20when%20writing%20Bash%20scripts,compromise%20your%20system%20or%20data.