November 22, 2025

Project 2 - Module 3

Team Members: Nimish Sabnis, Nimisha Madapura Rajashekhar, Hannah Caldwell Meurer

Step 1 - Create executable script: A complex bash script is created to monitor multiple aspects of the system. This script should be compatible with the Raspberry Pi OS system.

The functions the team generated to monitor the system are listed below.

1. Show Running Processes
2. Show Kernel Info
3. Show Disk Usage
4. Show System Uptime
5. Memory Summary
6. CPU Summary
7. Top 5 CPU Processes
8. Top 5 Memory Processes
9. Network Interfaces
10. Disk Usage Percentage
11. CPU Health Check
12. Memory Health Check
13. System Bottleneck Detector
14. Network Health Check
15. Long Running Processes
16. CPU Temperature
17. CPU Clock Speed

Step 2 - Convert script to an executable file:
The bash script is made into an executable using the chmod command. This changes the file permissions and adds the permission of making the file/script executable

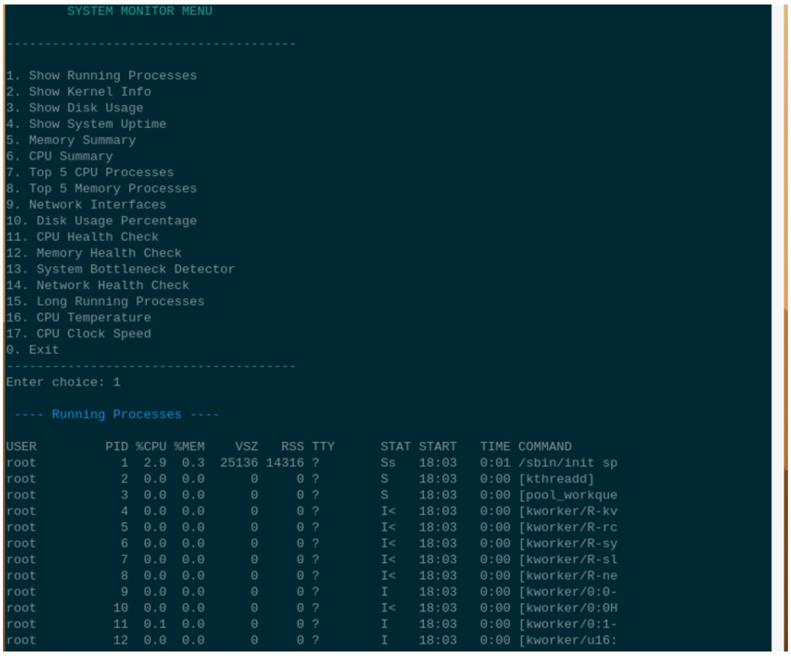
```
chmod +x [name of file/script]
```

Step 3 - Run :
From the folder that contains the file, run the executable with the step below

```
./[name of the file/script]
```

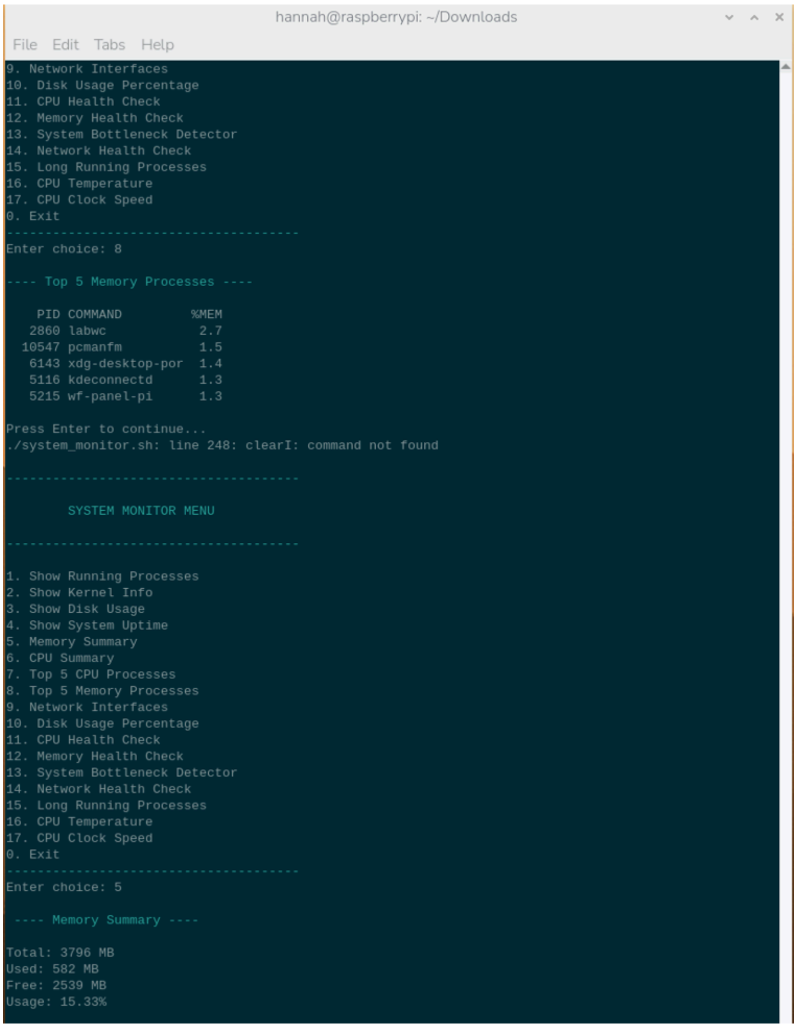
November 22, 2025

Terminal Results:



```
        SYSTEM MONITOR MENU

--------------------------------------

1. Show Running Processes
2. Show Kernel Info
3. Show Disk Usage
4. Show System Uptime
5. Memory Summary
6. CPU Summary
7. Top 5 CPU Processes
8. Top 5 Memory Processes
9. Network Interfaces
10. Disk Usage Percentage
11. CPU Health Check
12. Memory Health Check
13. System Bottleneck Detector
14. Network Health Check
15. Long Running Processes
16. CPU Temperature
17. CPU Clock Speed
0. Exit
--------------------------------------
Enter choice: 1

 ---- Running Processes ----

USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  2.9  0.3  25136 14316 ?        Ss   18:03   0:01 /sbin/init sp
root           2  0.0  0.0      0     0 ?        S    18:03   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        S    18:03   0:00 [pool_workque
root           4  0.0  0.0      0     0 ?        I<   18:03   0:00 [kworker/R-kv
root           5  0.0  0.0      0     0 ?        I<   18:03   0:00 [kworker/R-rc
root           6  0.0  0.0      0     0 ?        I<   18:03   0:00 [kworker/R-sy
root           7  0.0  0.0      0     0 ?        I<   18:03   0:00 [kworker/R-sl
root           8  0.0  0.0      0     0 ?        I<   18:03   0:00 [kworker/R-ne
root           9  0.0  0.0      0     0 ?        I    18:03   0:00 [kworker/0:0-
root          10  0.0  0.0      0     0 ?        I<   18:03   0:00 [kworker/0:0H
root          11  0.1  0.0      0     0 ?        I    18:03   0:00 [kworker/0:1-
root          12  0.0  0.0      0     0 ?        I    18:03   0:00 [kworker/u16:
```

Figure 1: The executable running on Linux Terminal



```
9. Network Interfaces
10. Disk Usage Percentage
11. CPU Health Check
12. Memory Health Check
13. System Bottleneck Detector
14. Network Health Check
15. Long Running Processes
16. CPU Temperature
17. CPU Clock Speed
0. Exit
--------------------------------------
Enter choice: 8

---- Top 5 Memory Processes ----

    PID COMMAND         %MEM
   2860 labwc            2.7
  10547 pcmanfm          1.5
   6143 xdg-desktop-por  1.4
   5116 kdeconnectd      1.3
   5215 wf-panel-pi      1.3

Press Enter to continue...
./system_monitor.sh: line 248: clearI: command not found

--------------------------------------

        SYSTEM MONITOR MENU

--------------------------------------

1. Show Running Processes
2. Show Kernel Info
3. Show Disk Usage
4. Show System Uptime
5. Memory Summary
6. CPU Summary
7. Top 5 CPU Processes
8. Top 5 Memory Processes
9. Network Interfaces
10. Disk Usage Percentage
11. CPU Health Check
12. Memory Health Check
13. System Bottleneck Detector
14. Network Health Check
15. Long Running Processes
16. CPU Temperature
17. CPU Clock Speed
0. Exit
--------------------------------------
Enter choice: 5

 ---- Memory Summary ----

Total: 3796 MB
Used: 582 MB
Free: 2539 MB
Usage: 15.33%
```

Figure 2: Example of functions in the script

Execution at Boot Time:

To run the script at boot time, a user needs to utilize `systemd` or `crontab` to execute the script. The team added the following line to the crontab file to execute the script and output a log file.

```
@reboot /bin/bash /home/hannah/Downloads/system_monitor.sh >>
/home/hannah/bootlog.txt 2>&1
```

For the script to behave the same at boot time as it does in the terminal, edits were necessary. The boot environment behaves differently from a standard terminal. During boot, the script needs to pause so the user can enter commands; otherwise, the boot process will continue, and the script will run in the background without notifying the user. There were also noticeable differences in formatting: color assignments are ignored, and special characters should be removed. Therefore, if a future client wishes to have a script that is usable in both environments, the code must account for these requirements during development

Recording Differences:

Function Name	**Boot Time**	**Terminal**
System Uptime	`16:45:54 up 0 min,  0 users,  load average: 2.08, 0.43, 0.14`	17:11:28 up 25 min, 2 users, load average: 0.77, 0.88, 0.67

Memory Summary	`Total: 3796 MB`
	`Used: 254 MB`
	`Free: 3459 MB`
	`Usage: 6.69%`
		Total: 3796 MB
		Used: 995 MB
		Free: 1552 MB
		Usage: 26.21%

CPU Summary	`User: 43.2%  System: 34.6%  Idle: 3.7%`	User: 6.1% System: 2.0% Idle: 91.8%

Top 5 CPU Processes	`PID COMMAND %CPU`
	`815 systemd-hostnam 52.1`
	`336 cloud-init 37.3`
	`694 polkitd 19.2`
	`829 ModemManager 18.7`
	`699 systemd-logind 15.2`
		PID COMMAND %CPU
		3157 ps 400
		1099 labwc 5.0
		1587 mousepad 5.0
		1965 chromium 4.5
		1871 chromium 1.9

Top 5 Memory Processes	`PID COMMAND`	PID COMMAND %MEM

	%MEM 336 cloud-init 1.2 170 plymouthd 0.7 1 systemd 0.3 370 systemd-udevd 0.2 423 (udev-worker) 0.2	1965 chromium 8.2 1810 chromium 6.9 1871 chromium 3.8 1099 labwc 3.1 1974 chromium 2.6
Network Interfaces	lo UNKNOWN 127.0.0.1/8 ::1/128 eth0 DOWN wlan0 DOWN	lo UNKNOWN 127.0.0.1/8 ::1/128 eth0 UP 192.168.1.2/24 fe80::90c:90e9:cb84:e2a4/64 wlan0 UP 192.168.1.103/24 2603:9001:5500:e9fc::1e5c/128 2603:9001:5500:e9fc:162c:9bf2:ef69:db8b/64 fd00:be96:e584:c451:809c:c6e5:5b0f:8a90/64 fe80::f717:a854:3ecd:aa62/64
Disk Usage Percentage	Filesystem Use% udev 0% tmpfs 2% /dev/mmcblk0p2 15% tmpfs 0% tmpfs 1% tmpfs 0% tmpfs 0% /dev/mmcblk0p1 16%	Filesystem Use% udev 0% tmpfs 2% /dev/mmcblk0p2 15% tmpfs 1% tmpfs 1% tmpfs 0% tmpfs 6% /dev/mmcblk0p1 16% tmpfs 1% tmpfs 0%
CPU Health Check	CPU Used: [33m80.0%[0m Status: [33mWARNING[0m	CPU Used: 6.7% Status: HEALTHY
Memory Health Check	Memory Usage: [33m6.00%[0m Status: [32mHEALTHY[0m	Memory Usage: 26.00% Status: HEALTHY
System Bottleneck Detector	CPU Idle: 47.8% Free Memory: 3413 MB IO Wait: 0.0% Bottleneck: NONE	CPU Idle: 91.7% Free Memory: 1549 MB IO Wait: 0.0% Bottleneck: NONE
Network Health Check	Established Connections: 0 Waiting Connections: 0 Status: [32mNORMAL[0m	Established Connections: 2 Waiting Connections: 0 Status: NORMAL

CPU Temperature	Temperature: 48.1 °C	Temperature: 45.2 °C
CPU Clock Speed	CPU Frequency: 1800.00 MHz	CPU Frequency: 600.00 MHz

References:

1. Kloudvm. “Simple Bash Script to Monitor CPU, Memory and Disk Usage on Linux in 10 Lines of Code” .kloudvm.medium.com. https://kloudvm.medium.com/simple-bash-script-to-monitor-cpu-memory-and-disk-usage-on-linux-in-10-lines-of-code-e4819fe38bf1 (accessed Nov. 16, 2015).
2. L. Rendek, “Bash script to monitor CPU and Memory usage on Linux”, linuxconfig.org. https://linuxconfig.org/bash-script-to-monitor-cpu-and-memory-usage-on-linux (accessed Nov. 16, 2015).
3. J. Ellingwood and A. Singh Walia, “Manage Systemd Services with systemctl on Linux”, DigitalOcean.com. https://www.digitalocean.com/community/tutorials/how-to-use-systemctl-to-manage-systemd-services-and-units (accessed Nov. 16, 2015).
4. Yogesh. (2014, October 17) “Get the load, cpu usage and time of executing a bash script”. stackoverflow.com. https://stackoverflow.com/questions/26425154/get-the-load-cpu-usage-and-time-of-executing-a-bash-script (accessed Nov. 16, 2015).