



PROJECT NAME –

## **Shell Scripting Automation**

### **Project Report**

Submitted By:

Name: **Karandeep**

UID: **23BCA10034**

Class: **23BCA2A**

Subject: **Linux Administration Lab**

Subject Code: **23CAP-305**

Submitted To:

Name: **Mr. Rajat Patial**

Designation: **Assistant Professor**

# **Project Title: Shell Scripting Automation for System Monitoring**

## **Aim**

To develop a **Bash shell script** that automatically displays system information such as uptime, CPU, memory, disk usage, and logged-in users — providing an automated, lightweight way to monitor system performance.

## **Objective**

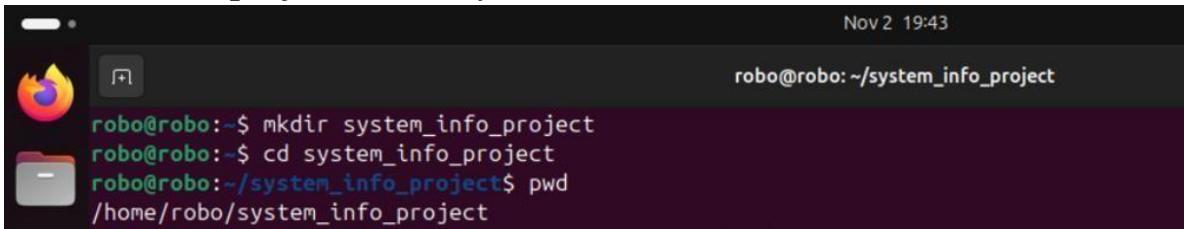
- To understand how to use Linux shell scripting for system automation.
- To create a simple monitoring script that provides important system metrics.
- To automate routine administrative tasks like resource checking.
- To demonstrate the use of core Linux commands for automation purposes.

## **Commands Used and Their Purpose**

Command	Purpose / Description
#!/bin/bash	Specifies the script interpreter (Bash).
echo	Prints formatted text or messages.
uptime -p	Shows how long the system has been running.
lscpu	Displays information about the CPU architecture.
grep "Model name"	Extracts the CPU model name from lscpu output.
who	Lists all users currently logged into the system.
date	Displays the current system date and time.
chmod +x	Grants execute permission to the script file.
./system_info.sh	Runs the script from the current directory.

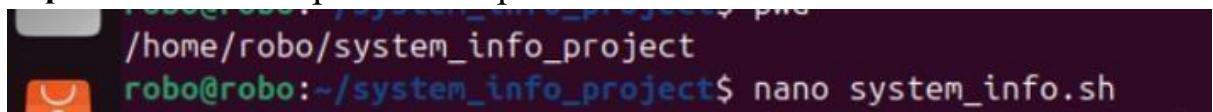
## Working and Implementation Steps

Step 1: Create a project directory



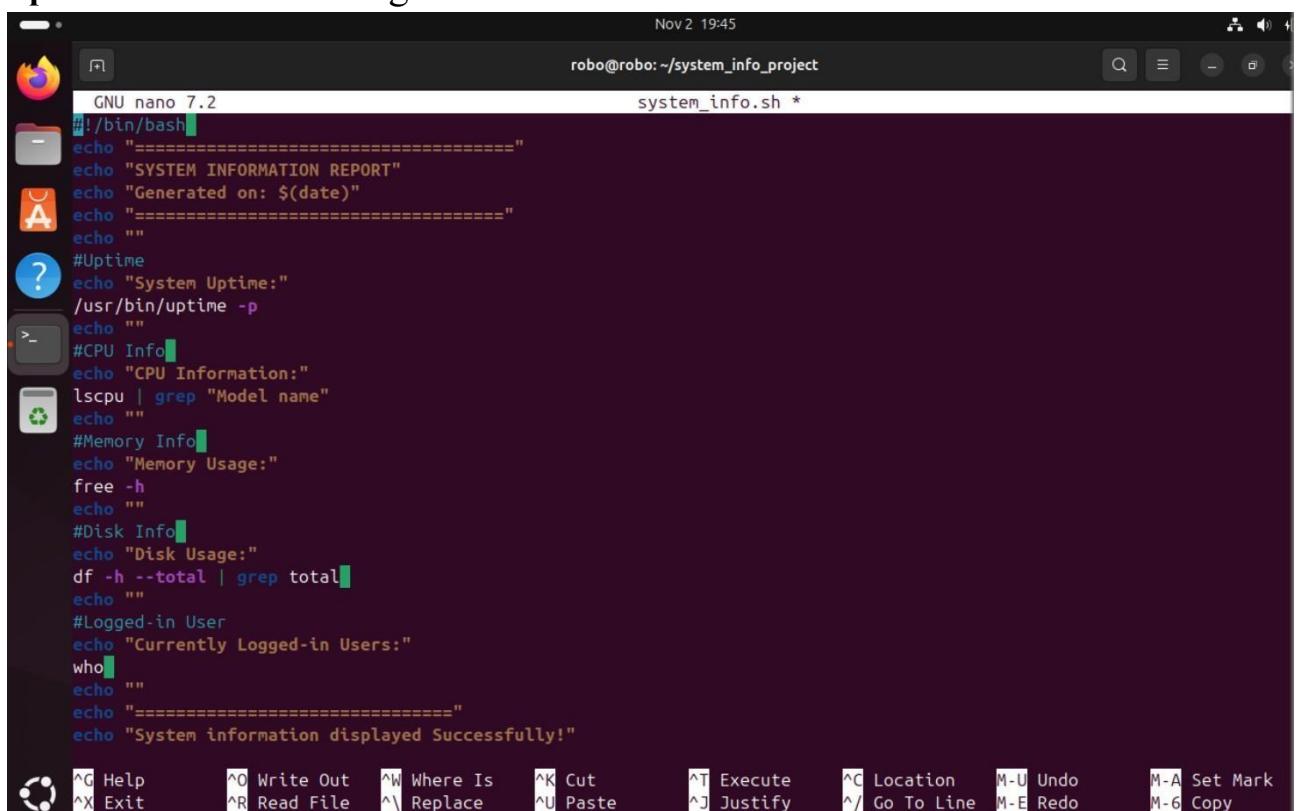
```
Nov 2 19:43
robo@robo:~/system_info_project
robo@robo:~$ mkdir system_info_project
robo@robo:~$ cd system_info_project
robo@robo:~/system_info_project$ pwd
/home/robo/system_info_project
```

Step 2: Create and open the script file



```
/home/robo/system_info_project
robo@robo:~/system_info_project$ nano system_info.sh
```

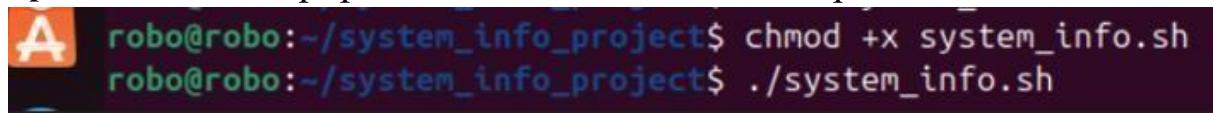
Step 3: Write the following code:



```
GNU nano 7.2
#!/bin/bash
=====
echo "SYSTEM INFORMATION REPORT"
echo "Generated on: $(date)"
echo =====
echo ""
#Uptime
echo "System Uptime:"
/usr/bin/uptime -p
echo ""
#CPU Info
echo "CPU Information:"
lscpu | grep "Model name"
echo ""
#Memory Info
echo "Memory Usage:"
free -h
echo ""
#Disk Info
echo "Disk Usage:"
df -h --total | grep total
echo ""
#Logged-in User
echo "Currently Logged-in Users:"
who
echo ""
echo =====
echo "System information displayed Successfully!"
```

Step 4: Save and exit Nano (Ctrl + O, Enter, Ctrl + X).

Step 5: Give the script permission to run. Run the script



```
A
robo@robo:~/system_info_project$ chmod +x system_info.sh
robo@robo:~/system_info_project$ ./system_info.sh
```

## Output

```
- robo@robo:~/system_info_project$ nano system_info.sh
robo@robo:~/system_info_project$ chmod +x system_info.sh
robo@robo:~/system_info_project$ ./system_info.sh
=====
A SYSTEM INFORMATION REPORT
Generated on: Sun Nov 2 07:42:24 PM UTC 2025
=====

System Uptime:
up 48 minutes

CPU Information:
Model name: 12th Gen Intel(R) Core(TM) i5-1240P

Memory Usage:
total        used        free      shared  buff/cache   available
Mem:       3.8Gi     1.0Gi     1.4Gi    33Mi     1.6Gi     2.7Gi
Swap:        0B         0B         0B

Disk Usage:
total      13G  6.6G  5.3G  56% -
Currently Logged-in Users:
robo      seat0      2025-11-02 18:54 (login screen)
robo      tty2      2025-11-02 18:54 (tty2)

=====
System information displayed Successfully!
=====
robo@robo:~/system_info_project$
```

## Advantages

Advantage	Description
<b>Automation</b>	Reduces repetitive manual checks by automating system info retrieval.
<b>Simplicity</b>	Easy to write and execute — perfect for beginners.
<b>Extensible</b>	Can be easily expanded with more system monitoring features.

<b>No Dependencies</b>	Uses only built-in Linux commands.
<b>Instant Output</b>	Displays essential system details in seconds.

## **Disadvantages**

<b>Disadvantage</b>	<b>Description</b>
<b>Limited Information</b>	Does not include OS or kernel version details.
<b>Text-Only Output</b>	No graphical or visual interface.
<b>Permission Restrictions</b>	Some system data may require elevated privileges.
<b>No Historical Tracking</b>	The script only shows current status unless redirected to a log file.

## **Use Cases**

- System Administrators: To quickly view system uptime, memory, and CPU status.
- Developers: To check system resource availability before deploying or testing.
- Support Teams: For quick diagnosis of performance-related issues.
- Server Maintenance: To monitor Linux server health automatically.

## Future Scope

- Redirect output to a log file for daily records.
- Email or notify admin if resource usage exceeds set limits.
- Integrate with cron jobs for scheduled automation.
- Extend functionality for remote servers using SSH.
- Create graphical dashboards using web interfaces or Python integration.

## Conclusion

This project successfully demonstrates the use of Bash shell scripting for automating system information retrieval.

By running simple Linux commands in a structured script, system administrators can monitor critical metrics efficiently.

It provides a strong foundation for advanced system automation and DevOps workflows.

Github Link: [GitHub - Karan49557/PBL\\_linux](https://github.com/Karan49557/PBL_linux)

The screenshot shows the GitHub repository page for 'PBL\_linux' owned by 'Karan49557'. The repository is public. At the top, there are navigation links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below the header, there's a summary showing 1 branch, 0 tags, and 2 commits. A file named 'Screenshots' is listed under the 'Screenshots' section. A PDF file named '23BCA10783.pdf' is also listed. A 'README' file is present, indicated by a 'README' link. A prominent 'Add a README' button is located at the bottom right of the README section. The right side of the interface features a vertical sidebar with icons for navigation and repository management.