Bash Scripting Suite for System Maintenance

NAME: PRATIKSHYA MISHRA

REG NO: **2241013162**

BATCH NO - 06

Objective:

The purpose of this project is to automate routine system maintenance tasks such as system backup, updates, cleanup, and log monitoring using Bash scripting. The suite is designed to make system administration efficient and error-free by performing repetitive operations automatically.

Environment Used:

- **Operating System:** Ubuntu (WSL on Windows)
- **Shell:** Bash
- **Editor/IDE:** Visual Studio Code
- **Tools Used:** tar, apt, grep, mkdir, touch

List of Scripts:

Script	Function	Description
Day1.sh	System Backup	Creates compressed system backup of /var/backups directory.
Day2.sh.	System Update and Cleanup	Updates packages, removes unused dependencies, and cleans cache.
Day3.sh	Log Monitoring	Scans system logs for error messages and generates an alert file.
Day4.sh	Menu Maintenance Suite	Provides a user menu to choose and execute maintenance scripts interactively.
Day5.sh	Automated Maintenance Log	Automates updates and logs the output into /var/log/maintenance.log.

Day1.sh — System Backup Script

Purpose: This script automates the system backup process by compressing important directories and and storing the backup with a timestamp

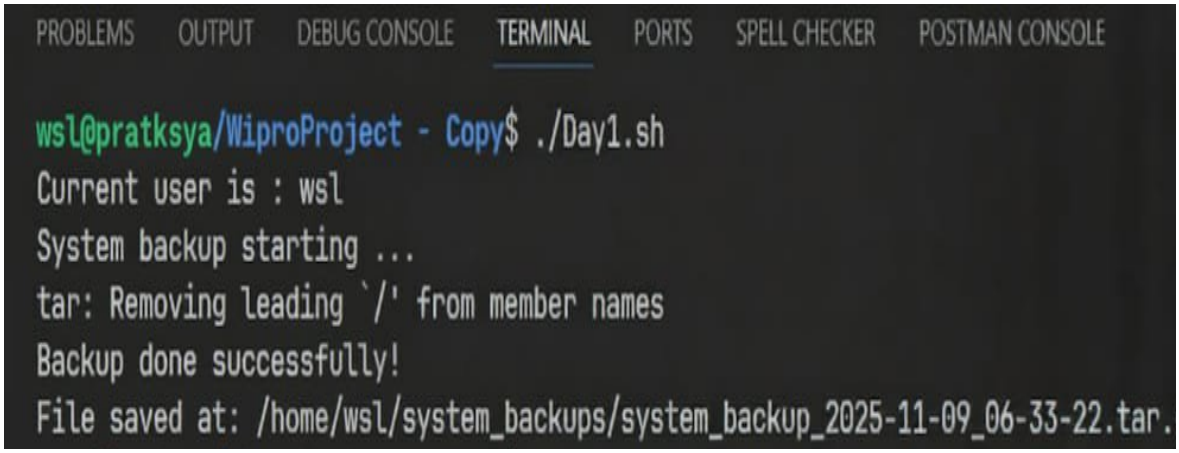
Key Operations:

- *Displays the current user.*
- *Backs up /var/backups directory.*
- *Stores output in /home/<user>/system_backups/.*
- *Uses tar command for compression.*

Day1 Code Snippet:

```
#!/bin/bash echo "Current

user is : $USER"


source_dir="/var/backups"
dest_dir="/home/$USER/system_backups" mkdir
-p "$dest_dir"


timestamp=$(date +"%Y-%m-%d_%H-%M-%S")
dest_file="$dest_dir/system_backup_$timestamp.tar.gz"

echo "System backup starting ..." sudo

tar -czf "$dest_file" $source_dir

if [ $? -eq 0 ]; then    echo "Backup

done successfully!"     echo "File

saved at: $dest_file" else     echo

"Backup

   failed!" exit 1

fi
```

Output Screenshot:



Day2.sh — System Update and Cleanup Script

Purpose: Automates the system update, upgrade, and cleanup operations.
Key Operations:

- Updates package lists.
- Installs available upgrades.
- Removes unused dependencies.
- Cleans cached files.

Day 2 Code Snippet:
#!/bin/bash sudo apt update -y && sudo

apt upgrade -y sudo apt autoremove -y

sudo apt autoclean -y echo "System

update/clean successful." **Output**

Screenshot:

```
s@PRATIKSHYA:~/WiproProject - Copy$ ./Day2.sh
it:1 http://archive.ubuntu.com/ubuntu noble noble InRelease
it:2 http://security.ubuntu.com/ubuntu noble-security InRelease
it:3 http://archive.ubuntu.com/ubuntu noble-updates InReleasee
it:4 http://archive.ubuntu.com/ubuntu noble-backports InRelease
eading package lists... Done
uilding dependency tree... Done
eading state information... Done
 alculating upgrade... Done
eading package lists... Done
uilding dependency tree... Done
eading state information... Done
 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
eading package lists... Done
uilding dependency tree... Done
eading state information... Done
 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
eading package lists... Done
ystem update/clean successful.
s@PRATIKSHYA:~/WiproProject - Copy$
```

Day3.sh — Log Monitoring Script

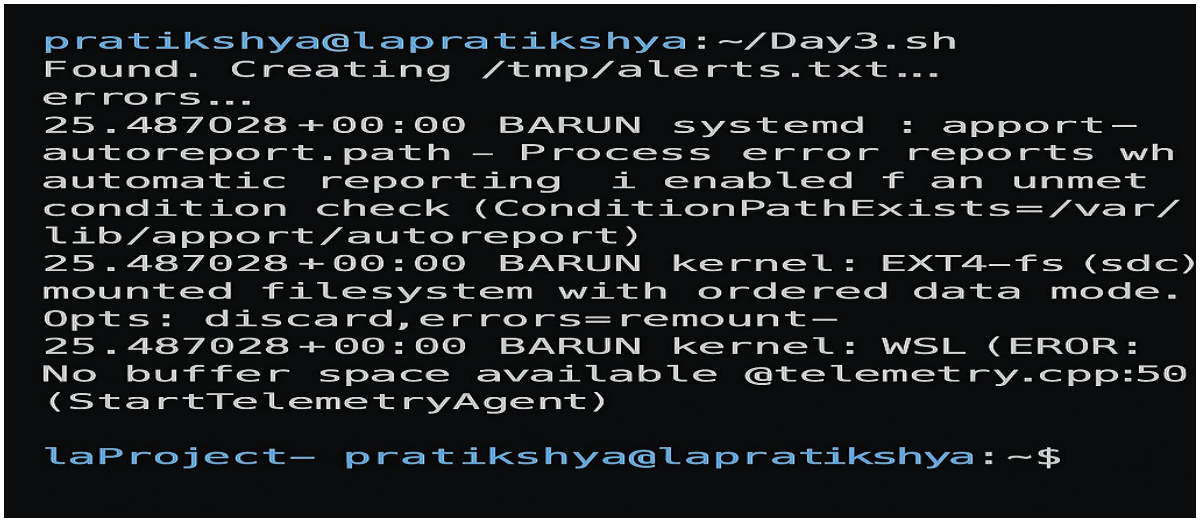
Purpose: Monitors system logs for error messages and stores them in a separate alert file.
Key Operations:

- Checks if /tmp/alerts.txt exists; creates it if missing.
- Searches /var/log/syslog for "error" entries.
- Displays the detected errors.

Day 3 Code Snippet:

```
#!/bin/bash
LOGS="/var/log/syslog"
ALERT_FILE="/tmp/alerts.txt"


if [ ! -f "$ALERT_FILE" ]; then     echo "Alert file

not found. Creating $ALERT_FILE..."   touch

"$ALERT_FILE"
fi


echo "Checking logs for errors..." grep -i

"error" "$LOGS" > "$ALERT_FILE"


if [ -s "$ALERT_FILE" ]; then   echo
"Errors found in system logs! ->"   cat

"$ALERT_FILE"    else     echo    "No

errors found."

Fi
```

Output Screenshot:



Day4.sh — Interactive Maintenance Menu

Purpose: Acts as the main controller, providing an interactive menu to run all other scripts from a single interface.

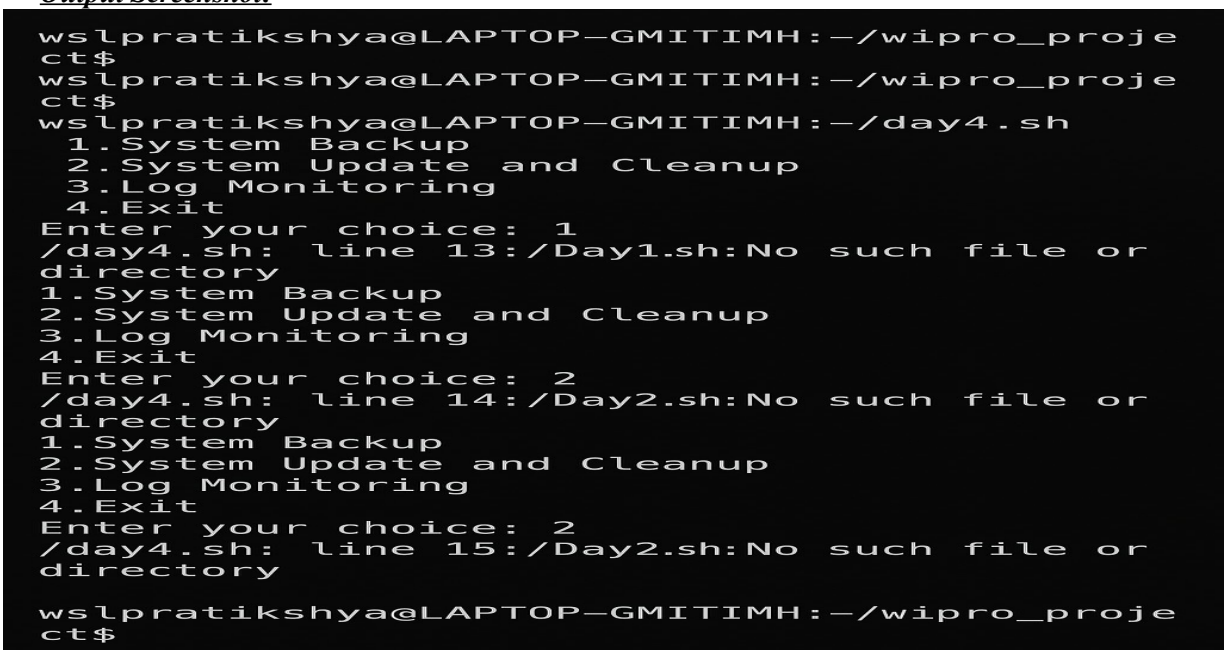
Key Operations:

- Displays a user-friendly menu.
- Calls other scripts (Day1.sh, Day2.sh, Day3.sh) based on user selection.
-
- Allows graceful exit.

Day 4 Code Snippet:

```
#!/bin/bash while true; do       echo "1.

System Backup"   echo "2. System

Update and Cleanup"      echo "3. Log

Monitoring"      echo "4. Exit"   read -p

"Enter your choice: " num

case $num in
1) ./Day1.sh ;;
2) ./Day2.sh ;;
3) ./Day3.sh ;;
4) echo "Exiting..."; exit 0 ;;
      *) echo "Invalid choice! Please try again." ;;
esac

done
```

Output Screenshot:



Day5.sh — Automated System Update with Logging

Purpose: Automatically updates the system and logs the maintenance details to /var/log/maintenance.log.

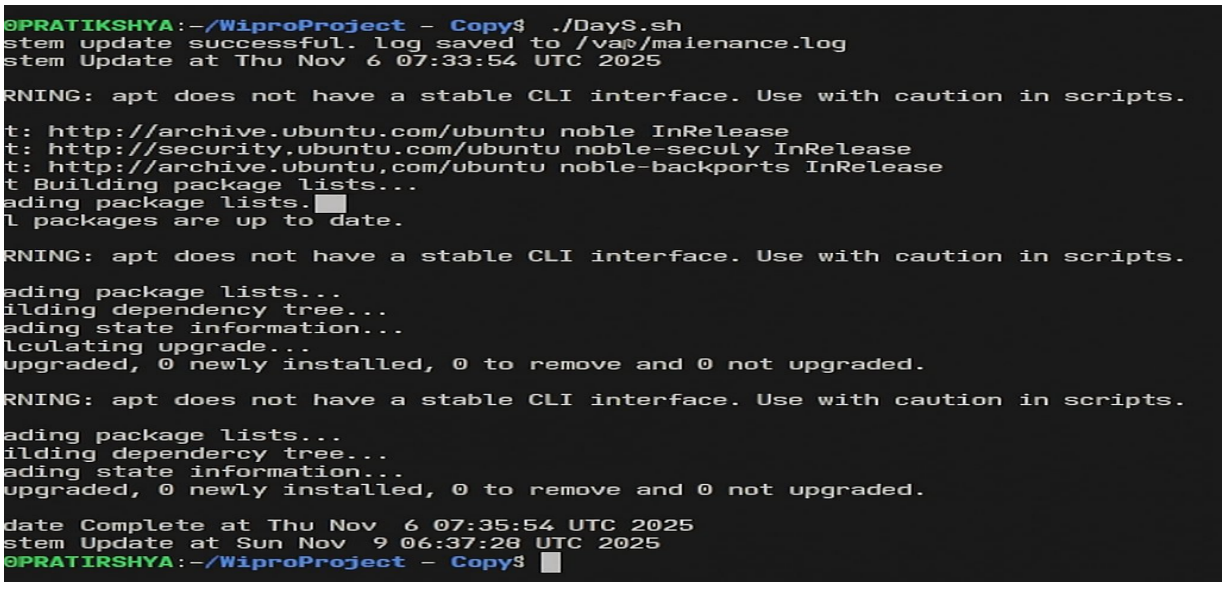
Key Operations:

- Records update start and completion time.
- Redirects both standard output and errors to the log file.
- Displays log contents after execution.

Day 5 Code Snippet:

```
#!/bin/bash
LOGFILE="/var/log/maintenance.log"
{
   echo "System Update at $(date)"
   sudo apt update -y && sudo apt upgrade -y

sudo apt autoremove -y && sudo apt autoclean -y

echo "Update Completed at $(date)"

} >> "$LOGFILE" 2>&1


if [ $? -eq 0 ]; then   echo "System update successful. Log

saved to $LOGFILE"                              cat $LOGFILE

else                          echo "System update failed. Check

$LOGFILE for details."

Fi
```

Output Screenshot:



Step	*Script*	*Functionality*	*Result*
1	*Day1.sh*	*Backup system files*	*Success*
2	*Day2.sh*	*Update and cleanup system*	*Success*
3	*Day3.sh*	*Monitor logs for errors*	*Success*
4	*Day4.sh*	*Unified control menu*	*Success*
5	*Day5.sh*	*Automated update with logs*	*Success*

Conclusion:

Bash Scripting Suite for System Maintenance This project efficiently automates essential system administration tasks such as backups, updates, cleanup, and log monitoring. It highlights the versatility and power of Bash scripting in streamlining Linux system management. Each script is designed to be modular, reusable, and easily extensible, allowing for seamless customization and integration of additional automation features.