

# Capstone Project Report

---

## 1. Title

Bash Scripting Suite for System Maintenance

## 2. Objective

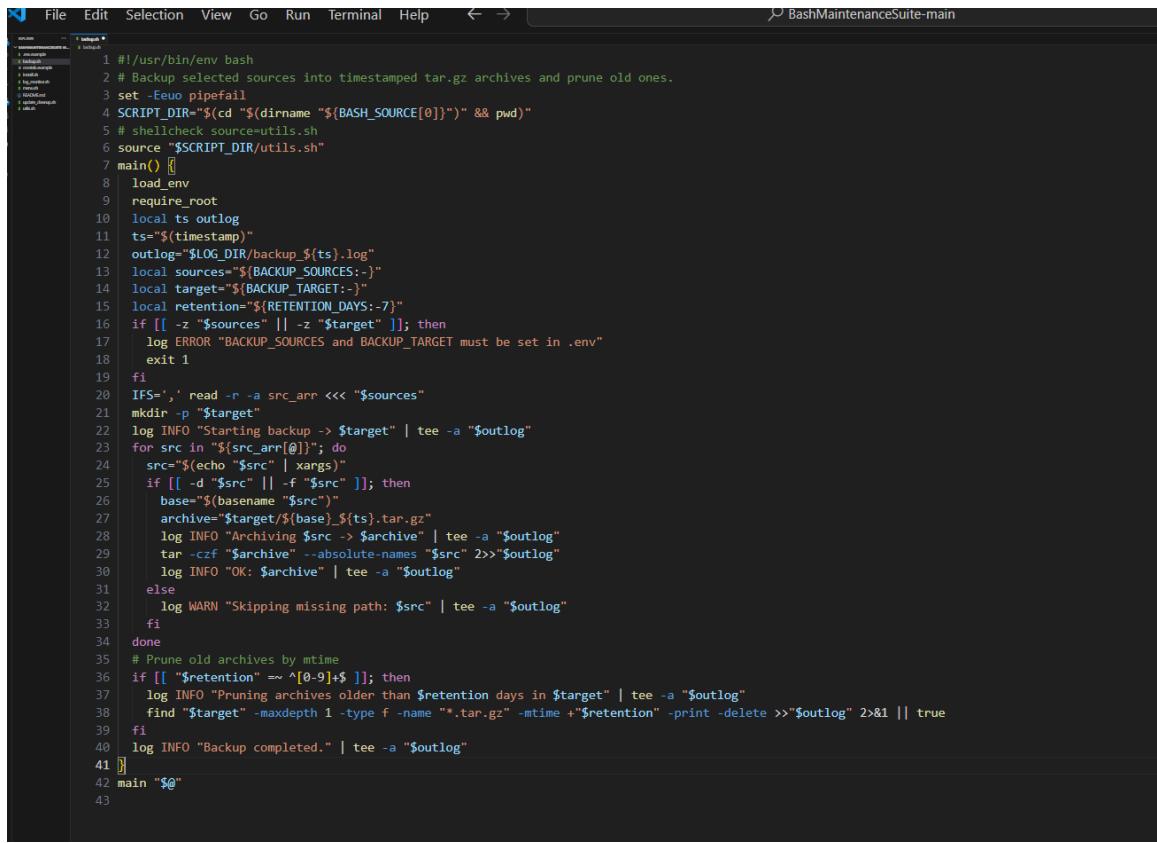
This project automates Linux system maintenance tasks such as backups, updates, cleanup, and log monitoring using Bash scripting. It is designed to simplify routine administrative work through automation and improve system reliability by scheduling essential maintenance tasks.

## 3. Tools and Technologies

- Programming Language: Bash (v4.0+)
- Platform: Linux (Ubuntu, Debian, Fedora, Arch)
- Tools: tar, apt, dnf, pacman, grep, journalctl, cron, systemd
- Editor: Visual Studio Code
- Version Control: GitHub

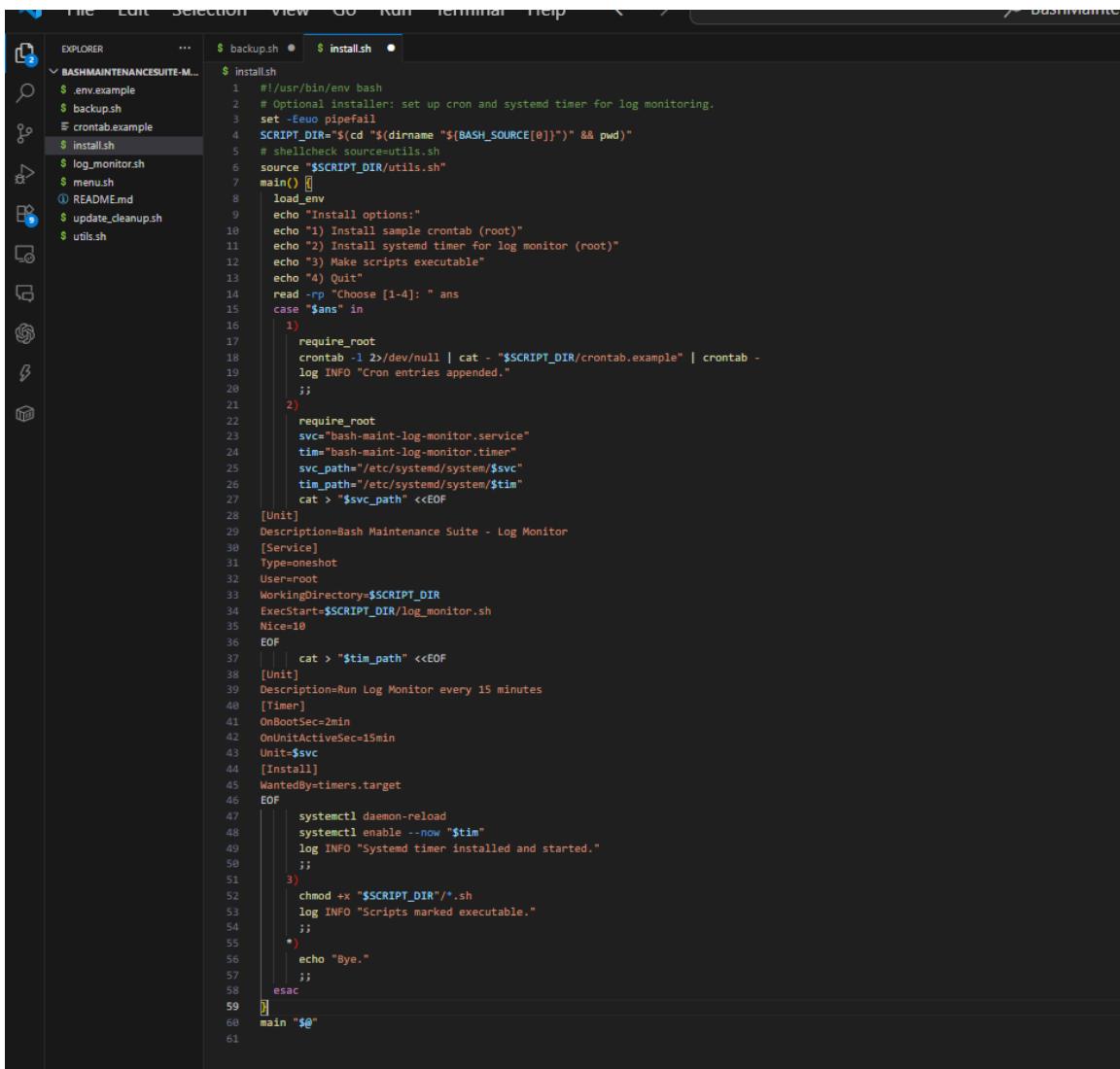
## 4. Codes

Below are the implemented Bash scripts used in the project:



```
File Edit Selection View Go Run Terminal Help ← → ⌘ BashMaintenanceSuite-main
1 #!/usr/bin/env bash
2 # Backup selected sources into timestamped tar.gz archives and prune old ones.
3 set -Euo pipefail
4 SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
5 # shellcheck source=utils.sh
6 source "$SCRIPT_DIR/utils.sh"
7 main() {
8     load_env
9     require_root
10    local ts outlog
11    ts="${timestamp}"
12    outlog="$LOG_DIR/backup_${ts}.log"
13    local sources="${BACKUP_SOURCES:-}"
14    local target="${BACKUP_TARGET:-}"
15    local retention="${RETENTION_DAYS:-7}"
16    if [[ -z "$sources" || -z "$target" ]]; then
17        log ERROR "BACKUP_SOURCES and BACKUP_TARGET must be set in .env"
18        exit 1
19    fi
20    IFS=',' read -r -a src_arr <<< "$sources"
21    mkdir -p "$target"
22    log INFO "Starting backup -> $target" | tee -a "$outlog"
23    for src in "${src_arr[@]}"; do
24        src="$echo \"$src\" | xargs"
25        if [[ -d "$src" || -f "$src" ]]; then
26            base=$(basename "$src")
27            archive="$target/${base}_${ts}.tar.gz"
28            log INFO "Archiving $src -> $archive" | tee -a "$outlog"
29            tar -czf "$archive" --absolute-names "$src" 2>>"$outlog"
30            log INFO "OK: $archive" | tee -a "$outlog"
31        else
32            log WARN "Skipping missing path: $src" | tee -a "$outlog"
33        fi
34    done
35    # Prune old archives by mtime
36    if [[ "$retention" =~ ^[0-9]+$ ]]; then
37        log INFO "Pruning archives older than $retention days in $target" | tee -a "$outlog"
38        find "$target" -maxdepth 1 -type f -name "*tar.gz" -mtime +"$retention" -print -delete >>"$outlog" 2>&1 || true
39    fi
40    log INFO "Backup completed." | tee -a "$outlog"
41 }
42 main "$@"
43
```

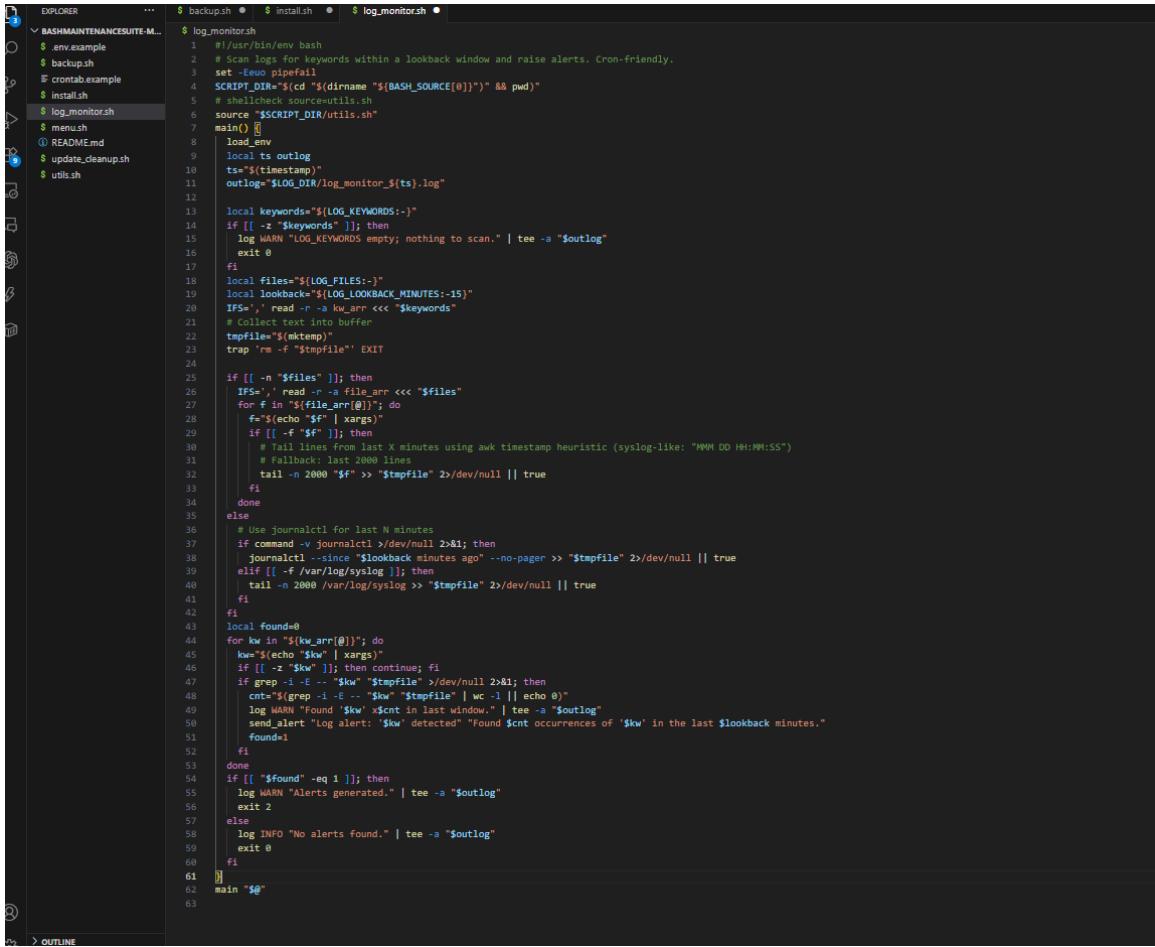
backup.sh



The screenshot shows a terminal window with the title 'BashMaintenance'. The terminal is displaying the contents of the 'install.sh' script. The script is a shell script that installs a cron job and a systemd timer for log monitoring. It includes options for installing sample crontab, setting up a log monitor, making scripts executable, and quitting. It also handles root privileges, writes to configuration files, and starts the service.

```
$ installsh
1 #!/usr/bin/env bash
2 # Optional installer: set up cron and systemd timer for log monitoring.
3 set -Eeu pipefail
4 SCRIPT_DIR=$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)
5 # shellcheck source=utils.sh
6 source "$SCRIPT_DIR/utils.sh"
7 main() {
8     load_env
9     echo "Install options:"
10    echo "1) Install sample crontab (root)"
11    echo "2) Install systemd timer for log monitor (root)"
12    echo "3) Make scripts executable"
13    echo "4) Quit"
14    read -rp "Choose [1-4]: " ans
15    case $ans in
16        1)
17            require_root
18            crontab -l 2>/dev/null | cat - "$SCRIPT_DIR/crontab.example" | crontab -
19            log INFO "Cron entries appended."
20            ;;
21        2)
22            require_root
23            svc="bash-maint-log-monitor.service"
24            tim="bash-maint-log-monitor.timer"
25            svc_path="/etc/systemd/system/$svc"
26            tim_path="/etc/systemd/system/$tim"
27            cat > "$svc_path" <<EOF
28            [Unit]
29            Description=Bash Maintenance Suite - Log Monitor
30            [Service]
31            Type=oneshot
32            User=root
33            WorkingDirectory=$SCRIPT_DIR
34            ExecStart=$SCRIPT_DIR/log_monitor.sh
35            Nice=10
36            EOF
37            cat > "$tim_path" <<EOF
38            [Unit]
39            Description=Run Log Monitor every 15 minutes
40            [Timer]
41            OnBootSec=2min
42            OnUnitActiveSec=15min
43            Unit=$svc
44            [Install]
45            WantedBy=timers.target
46            EOF
47            systemctl daemon-reload
48            systemctl enable --now "$tim"
49            log INFO "Systemd timer installed and started."
50            ;;
51        3)
52            chmod +x "$SCRIPT_DIR"/*.sh
53            log INFO "Scripts marked executable."
54            ;;
55        *)
56            echo "Bye."
57            ;;
58    esac
59 }
60 main "$@"
61
```

install.sh



The screenshot shows a terminal window with the following content:

```
#!/usr/bin/env bash
# scan logs for keywords within a lookback window and raise alerts. Cron-friendly.
set -euo pipefail
SCRIPT_DIR=$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)
shellcheck sourceutils.sh
source "$SCRIPT_DIR/utils.sh"
main() {
load_env
local ts outlog
ts=$(date "+%Y-%m-%d %H:%M:%S")
outlog="$LOG_DIR/log_monitor_${ts}.log"

local keywords="$LOG_KEYWORDS:-"
if [[ -z "$keywords" ]]; then
log WARN "LOG_KEYWORDS empty; nothing to scan." | tee -a "$outlog"
exit 0
fi
local files="$LOG_FILES:-"
local lookback="$LOG_LOOKBACK_MINUTES:-15"
IFS=' '
read -r -a kw_arr <<< "$keywords"
# Collect text into buffer
tmpfile=$(mktemp)
trap 'rm -f "$tmpfile"' EXIT
if [[ -n "$files" ]]; then
IFS=' '
read -r -a file_arr <<< "$files"
for f in "${file_arr[@]}"; do
f=$(echo $f | xargs)
if [[ -f "$f" ]]; then
# Tail lines from last X minutes using awk timestamp heuristic (syslog-like: "MMM DD HH:MM:SS")
# Fallback: last 2000 lines
tail -n 2000 "$f" >> "$tmpfile" 2>/dev/null || true
fi
done
else
# Use journalctl for last N minutes
if command -v journalctl >/dev/null 2>&1; then
journalctl --since "$lookback minutes ago" --no-pager >> "$tmpfile" 2>/dev/null || true
elif [[ -f "/var/log/syslog" ]]; then
tail -n 2000 /var/log/syslog >> "$tmpfile" 2>/dev/null || true
fi
fi
local found=0
for kw in "${kw_arr[@]}"; do
kw=$(echo $kw | xargs)
if [[ -z "$kw" ]]; then continue; fi
if grep -i -E -- "$kw" "$tmpfile" >/dev/null 2>&1; then
cnt=$(grep -i -E -- "$kw" "$tmpfile" | wc -l || echo 0)
log WARN "Found '$kw' $cnt in last window." | tee -a "$outlog"
send_alert "Log alert: '$kw' detected" "Found $cnt occurrences of '$kw' in the last $lookback minutes."
found=1
fi
done
if [[ "$found" -eq 1 ]]; then
log WARN "Alerts generated." | tee -a "$outlog"
exit 2
else
log INFO "No alerts found." | tee -a "$outlog"
exit 0
fi
fi
main "$@"

```

logmonitor.sh

The screenshot shows a terminal window with four tabs open: backup.sh, install.sh, log\_monitor.sh, and menu.sh. The menu.sh tab is active and displays the following script:

```
$ menu.sh
1  #!/usr/bin/env bash
2  set -euo pipefail
3  SCRIPT_DIR=$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
4  # shellcheck source=utils.sh
5  source "$SCRIPT_DIR/utils.sh"
6
7  show_menu() {
8  | cat <<'EOF'
9  =====
10 | System Maintenance Suite
11 =====
12 1) Run Backup
13 2) Run Update & Cleanup
14 3) Run Log Monitor Scan (now)
15 4) Show Logs directory
16 5) Exit
17 EOF
18 }
19
20 main() {
21   load_env
22   while true; do
23     show_menu
24     read -rp "Choose an option [1-5]: " choice
25     case "$choice" in
26       1) sudo "$SCRIPT_DIR/backup.sh" ;;
27       2) sudo "$SCRIPT_DIR/update_cleanup.sh" ;;
28       3) "$SCRIPT_DIR/log_monitor.sh" ;;
29       4) echo "Logs at: $LOG_DIR"; ls -1 "$LOG_DIR";;
23       5) echo "Bye!"; exit 0 ;;
24       *) echo "Invalid choice" ;;
25     esac
26     read -rp "Press Enter to continue..." _
27   done
28 }
29
30 main "$@"
31
```

menu.sh

The screenshot shows a terminal window with a file browser on the left and a code editor on the right. The file browser lists several files in a directory named 'BASHMAINTENANCESUITE-M...'. The code editor displays a script named 'utils.sh' with the following content:

```
1 #!/usr/bin/env bash
2 set -euo pipefail
3 SCRIPT_DIR=$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)
4 PROJECT_ROOT=$SCRIPT_DIR
5 LOG_DIR="$PROJECT_ROOT/logs"
6 mkdir -p "$LOG_DIR"
7 timestamp() { date +"%Y-%m-%d_%H-%M-%S"; }
8 log() {
9     local level="${1:-INFO}"; shift || true
10    local msg="$"
11    local ts
12    ts=$(timestamp)
13    echo "[${ts}] [${level}] ${msg}"
14 }
15 log_to_file() {
16     local file="$1"; shift
17     local ts
18     ts=$(timestamp)
19     printf "[%s] %s\n" "$ts" "$msg" >> "$file"
20 }
21 require_root() {
22     if [[ "$EUID-$(id -u)" -ne 0 ]]; then
23         log ERROR "This action requires root. Re-run with sudo."
24         exit 1
25     fi
26 }
27 load_env() {
28     local env_file="$PROJECT_ROOT/.env"
29     if [[ -f "$env_file" ]]; then
30         # shellcheck disable=SC1090
31         source "$env_file"
32     else
33         log WARN "No .env found. Using defaults where possible."
34     fi
35 }
36 detect_pkg_manager() {
37     local forced="${PKG_MANAGER:-}"
38     if [[ -n "$forced" ]]; then
39         echo "$forced"
40         return
41     fi
42     if command -v apt >/dev/null 2>&1; then echo "apt" && return; fi
43     if command -v dnf >/dev/null 2>&1; then echo "dnf" && return; fi
44     if command -v pacman >/dev/null 2>&1; then echo "pacman" && return; fi
45     echo ""
46 }
47 send_alert() {
48     local subject="$1"; shift
49     local body="$*"
50     local recipient="${ALERT_RECIPIENT:-}"
51     local log_file="$LOG_DIR/alerts_${(date +%Y-%m-%d)}.log"
52     log_to_file "$log_file" "$subject :: $body"
53     if [[ -n "$recipient" ]] && command -v mail >/dev/null 2>&1; then
54         printf "%s\n" "$body" | mail -s "$subject" "$recipient" || true
55     fi
56 }
57
```

ulta.sh

```
✗ BASHMAINTENANCESUITE-M... $ update_cleanup.sh
$ ./env.example
$ backup.sh
$ crontab.example
$ installsh
$ log_monitor.sh
$ menush
$ README.md
$ update_cleanup.sh
$ utils.sh

update_cleanup.sh
1 #!/usr/bin/env bash
2 # Update packages and clean caches. Supports apt, dnf, pacman.
3 set -Euo pipefail
4 SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
5 # shellcheck source=utils.sh
6 source "$SCRIPT_DIR/utils.sh"
7 main() {
8     load_env
9     require_root
10    local poutlog ts
11    ts=$(timestamp)
12    outlog="$LOG_DIR/update_cleanup_${ts}.log"
13    pm=$(detect_pkg_manager)
14    log INFO "Detected package manager: ${pm:-unknown}" | tee -a "$outlog"
15    case "$pm" in
16        apt)
17            export DEBIAN_FRONTEND=noninteractive
18            apt update | tee -a "$outlog"
19            apt -y upgrade | tee -a "$outlog"
20            apt -y autoremove | tee -a "$outlog"
21            apt -y autoclean | tee -a "$outlog"
22            ;;
23        dnf)
24            dnf -y upgrade | tee -a "$outlog"
25            dnf -y autoremove | tee -a "$outlog" || true
26            dnf -y clean all | tee -a "$outlog"
27            ;;
28        pacman)
29            pacman -Syu --noconfirm | tee -a "$outlog"
30            pacache -r -k 3 2>>"$outlog" || true # keep last 3 versions
31            ;;
32        *)
33            log ERROR "Unsupported or undetected package manager." | tee -a "$outlog"
34            ;;
35    esac
36    # General cleanup: journald & tmp
37    if command -v journalctl >/dev/null 2>&1; then
38        journalctl --vacuum-time=14d >>"$outlog" 2>&1 || true
39    fi
40    find /tmp -type f -mtime +7 -delete 2>>"$outlog" || true
41    log INFO "Update & cleanup completed." | tee -a "$outlog"
42    ;;
43 }
44 main "$@"
45
```

updatecleanup.sh

## 5. Full Execution Screenshots

The following screenshots demonstrate the successful execution of the scripts and their outputs:

## Backup File Output

```
Ali: ~
```

```
(akshya㉿kali)-[~/Documents/Wipro Project/BashMaintenanceSuite-main]
File Actions Edit View Help

└─$ cd BashMaintenanceSuite
cp .env.example .env
# Edit .env to match your system
chmod +x *.sh
./menu.sh
cd: no such file or directory: BashMaintenanceSuite
System Maintenance Suite
_____
1) Run Backup
2) Run Update & Cleanup
3) Run Log Monitor Scan (now)
4) Show Logs directory
5) Exit
Choose an option [1-5]: 1
[sudo] password for akshya:
[2025-11-08_15-48-31] [INFO] Starting backup → /var/backups
[2025-11-08_15-48-32] [INFO] Archiving /etc → /var/backups/etc_2025-11-08_15-48-31.tar.gz
[2025-11-08_15-48-38] [INFO] OK: /var/backups/etc_2025-11-08_15-48-31.tar.gz
[2025-11-08_15-48-38] [INFO] Archiving /home → /var/backups/home_2025-11-08_15-48-31.tar.gz
[2025-11-08_15-48-46] [INFO] OK: /var/backups/home_2025-11-08_15-48-31.tar.gz
[2025-11-08_15-48-46] [INFO] Pruning archives older than 7 days in /var/backups
[2025-11-08_15-48-46] [INFO] Backup completed.
Press Enter to continue ...
```

## System Update and Cleanup Output

```
Press Enter to continue ...
System Maintenance Suite
_____
1) Run Backup
2) Run Update & Cleanup
3) Run Log Monitor Scan (now)
4) Show Logs directory
5) Exit
Choose an option [1-5]: 2
[2025-11-08_15-48-58] [INFO] Detected package manager: apt
WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Ign:1 http://http.kali.org/kali kali-rolling InRelease
Ign:1 http://http.kali.org/kali kali-rolling InRelease
Ign:1 http://http.kali.org/kali kali-rolling InRelease
Err:1 http://http.kali.org/kali kali-rolling InRelease
  Temporary failure resolving 'http.kali.org'
Reading package lists...
Building dependency tree...
Reading state information...
All packages are up to date.
Warning: Failed to fetch http://http.kali.org/kali/dists/kali-rolling/InRelease  Temporary failure resolving 'http.kali.org'
Warning: Some index files failed to download. They have been ignored, or old ones used instead.

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Reading package lists...
Building dependency tree...
Reading state information...
Calculating upgrade...
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0
WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Reading package lists...
Building dependency tree...
Reading state information...
Summary:
```

```
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0
WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Reading package lists...
Building dependency tree...
Reading state information...
[2025-11-08_15-50-20] [INFO] Update & cleanup completed.
Press Enter to continue ...

System Maintenance Suite
_____
1) Run Backup
2) Run Update & Cleanup
3) Run Log Monitor Scan (now)
4) Show Logs directory
5) Exit
Choose an option [1-5]: 3
[2025-11-08_15-50-27] [INFO] No alerts found.
Press Enter to continue ...
```

## ② Log Monitoring Output

```
File Actions Edit View Help
Press Enter to continue ...
=====
System Maintenance Suite
=====
1) Run Backup
2) Run Update & Cleanup
3) Run Log Monitor Scan (now)
4) Show Logs directory
5) Exit
Choose an option [1-5]: 3
[2025-11-08_15-50-27] [INFO] No alerts found.
Press Enter to continue ...
```

## ③ Show log directory

```
=====
System Maintenance Suite
=====
1) Run Backup
2) Run Update & Cleanup
3) Run Log Monitor Scan (now)
4) Show Logs directory
5) Exit
Choose an option [1-5]: 4
Logs at: /home/akshya/Documents/Wipro Project/BashMaintenanceSuite-main/logs
backup_2025-11-08_15-48-31.log
log_monitor_2025-11-08_15-50-27.log
update_cleanup_2025-11-08_15-48-58.log
Press Enter to continue ...
```

## ④ Exit

```
=====
System Maintenance Suite
=====
1) Run Backup
2) Run Update & Cleanup
3) Run Log Monitor Scan (now)
4) Show Logs directory
5) Exit
Choose an option [1-5]: 5
Bye!
```

## **6. GitHub Repository**

**Link- “ <https://github.com/akshya44/Linux-System-Maintenance-Suite.git>“**

## **7. Conclusion**

The Bash Scripting Suite for System Maintenance provides a practical and efficient automation solution for routine Linux system tasks. It improves system uptime, reduces manual effort, and demonstrates real-world DevOps and shell scripting skills.