Bash Scripting Suite for System Maintenance — Project Report

Diptesh Singh

Date: 2025-11-09

1. Title

Bash Scripting Suite for System Maintenance — Automated Linux Maintenance (Backup, Update, Log Monitor)

---

2. Objective

Your goal: *Write a suite of Bash scripts to automate system maintenance tasks such as backup, system updates, and log monitoring.*

This project implements: - **Automated backups** with timestamped archives and retention policy.
- **Package updates & cleanup** using the detected package manager.
- **Log scanning and continuous monitoring** with alert writing.
- **Interactive menu** to run tasks and handle permission escalation.
- **Logging and basic error handling**.
- **Minimal privilege elevation** — only the commands that require root are elevated.

---

3. High-level design & architecture

Design goals - Minimal sudo: only elevate specific operations (`tar` for protected files, `mkdir` into root-owned destinations, package manager commands) to avoid making log files root-owned.
- **Modular scripts**: single responsibility per script.
- **Clear logging** to `${LOG_DIR:-./maintenance_logs}` for audit and debugging.
- **Cron-friendly**: scripts accept absolute paths and behave predictably in a cron environment.

Components 1. `menu.sh` — top-level interactive CLI.
2. `system_backup.sh` — backup + retention.
3. `system_update_cleanup.sh` — package updates & cleanup.
4. `log_monitoring.sh` — one-shot and continuous monitoring.
5. `utils.sh` — shared helpers and logging.
6. `script.sh` — sample/test log file used for demo.

Architecture diagram (text)

```
[User] -> menu.sh -> { system_backup.sh, system_update_cleanup.sh, log_monitoring.sh }
|
utils.sh (log(), safe_sudo(), detect_pkg_mgr())
```

---

4. How to run (summary)

1. Make scripts executable:

```
chmod +x *.sh
```

2. (Optional) create a local log dir:

```
mkdir -p maintenance_logs
```

3. Run interactive menu:

```
./menu.sh
```

4. Or run scripts directly.

Important: when using `crontab`, always reference **absolute** paths because cron runs with a minimal environment.

---

5. Files — purpose, usage, code and screenshots

5.1 `menu.sh` — interactive menu

Purpose: interactive CLI that prompts for input and calls the other scripts. **How to use:**

```
./menu.sh
```

Behavior:

- Option 1: Run backup (prompts for source(s) and destination).
- Option 2: Update & cleanup.
- Option 3: One-shot log scan.
- Option 4: Continuous log monitor.
- Option 5: Exit.

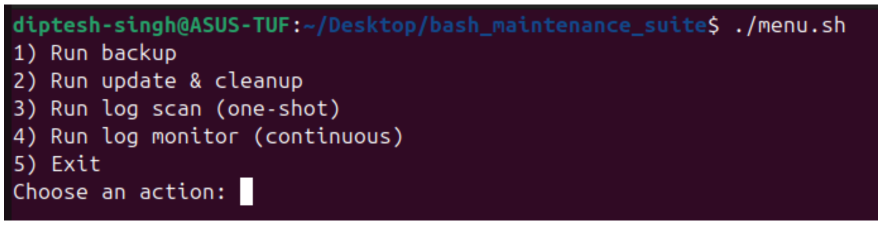
CODE — `menu.sh`

```
#!/usr/bin/env bash
# menu.sh - small interactive menu to run maintenance tasks
set -o errexit
set -o nounset
set -o pipefail

SCRIPT_DIR=$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)
```

```
source "$SCRIPT_DIR/utils.sh" || true

PS3="Choose an action: "

options=("Run backup" "Run update & cleanup" "Run log scan (one-shot)"
         "Run log monitor (continuous)" "Exit")

select opt in "${options[@]}"; do
  case $REPLY in
    1)
      read -rp "Enter source dir (comma separated for multiple): " srcs
      IFS=',' read -r -a raw_arr <<< "$srcs"

      normalized=()
      for s in "${raw_arr[@]}"; do
        # trim leading/trailing whitespace, preserve internal spaces
        s="${s#"${s%%[![:space:]]*}"}"
        s="${s%"${s##*[![:space:]]}"}"
        [ -n "$s" ] && normalized+=( "$s" )
      done

      if [ ${#normalized[@]} -eq 0 ]; then
        echo "No valid source provided. Cancelling."
        continue
      fi

      read -rp "Enter dest dir (default ./backups): " dest
      dest=${dest:-./backups}

      # --- Pre-check existence & permissions ---
      missing=()
      unreadable=()
      for s in "${normalized[@]}"; do
        if [ ! -e "$s" ]; then
          missing+=( "$s" )
        elif [ ! -r "$s" ]; then
          unreadable+=( "$s" )
        fi
      done

      # If any source is missing -> abort immediately (not a permission issue)
      if [ ${#missing[@]} -ne 0 ]; then
        echo "ERROR: The following source(s) do not exist:"
        for m in "${missing[@]}"; do echo "  - $m"; done
        echo "Please correct the paths and try again."
        continue
      fi
      # --- Show summary & issues (if any) ---
      echo ""
      echo "Sources to back up:"
      for s in "${normalized[@]}"; do echo "  - $s"; done
      echo "Destination: $dest"
      if [ ${#unreadable[@]} -ne 0 ]; then
        echo ""
        echo "Permission issues detected (these will require elevation):"
        for u in "${unreadable[@]}"; do echo "  * No read permission: $u"; done
      fi

      # --- Final confirmation ---
      read -rp $'\nProceed with backup now? [y/N]: ' proceed
      if [[ ! "${proceed,,}" =~ ^(y|yes)$ ]]; then
        echo "Backup cancelled by user."
        continue
      fi

      # --- Prepare arguments array for backup.sh ---
      args=()
      for s in "${normalized[@]}"; do args+=( -s "$s" ); done

      echo "Starting backup..."
      "$SCRIPT_DIR/system_backup.sh" "${args[@]}" -d "$dest"

      rc=$?
      if [ $rc -eq 0 ]; then
        echo "Backup finished successfully."
        log "Backup completed successfully for dest=$dest"
      else
        echo "Backup failed with exit code $rc."
        log "Backup failed with exit code $rc for dest=$dest"
      fi
      ;;
    2)
      "$SCRIPT_DIR/system_update_cleanup.sh"
      ;;
    3)
      read -rp "Log file (default /var/log/syslog): " lf
      lf=${lf:-/var/log/syslog}
      read -rp "Pattern (default 'ERROR|Failed'): " pat
      pat=${pat:-ERROR|Failed}
      echo "running one-shot scan"
      "$SCRIPT_DIR/log_monitoring.sh" -f "$lf" -p "$pat"
      ;;
    4)
      read -rp "Log file (default /var/log/syslog): " lf
      lf=${lf:-/var/log/syslog}
```

```
        read -rp "Pattern (default 'ERROR|Failed'): " pat
        pat=${pat:-ERROR|Failed}
        "$SCRIPT_DIR/log_monitoring.sh" -f "$lf" -p "$pat" -c
        ;;
      5)
        echo "Goodbye"
        break
        ;;
      *)
        echo "Invalid choice" ;;
  esac
done
```

Figure 1: Menu UI



```
diptesh-singh@ASUS-TUF:~/Desktop/bash_maintenance_suite$ ./menu.sh
1) Run backup
2) Run update & cleanup
3) Run log scan (one-shot)
4) Run log monitor (continuous)
5) Exit
Choose an action: 
```

Figure 1: Menu UI

*(Figure 1 — The interactive **`menu.sh`** CLI showing the available actions.)*

5.2 `system_backup.sh` — backup & retention

Purpose: create a **timestamped tar.gz** of provided sources, store in destination, and rotate (delete) archives older than retention days.

Usage examples:

```
# single-source backup to local backups folder (retain 14 days)
./system_backup.sh -s /home/you/testdata -d ./backups -r 14

# multiple sources, system destination (may trigger sudo)
./system_backup.sh -s /etc -s /var/log -d /var/backups/system -r 30
```

Flags

- `-s /path` — source (repeatable).
- `-d /dest` — destination directory (default `./backups`).
- `-r N` — retention (delete archives older than N days).
- `-n` — dry-run (prints the tar command without running it).

Behavior notes

- The script inspects sources and destination writability to decide whether `tar` requires `sudo`.
- Destination creation can be done with `sudo` only if parent is not writable.

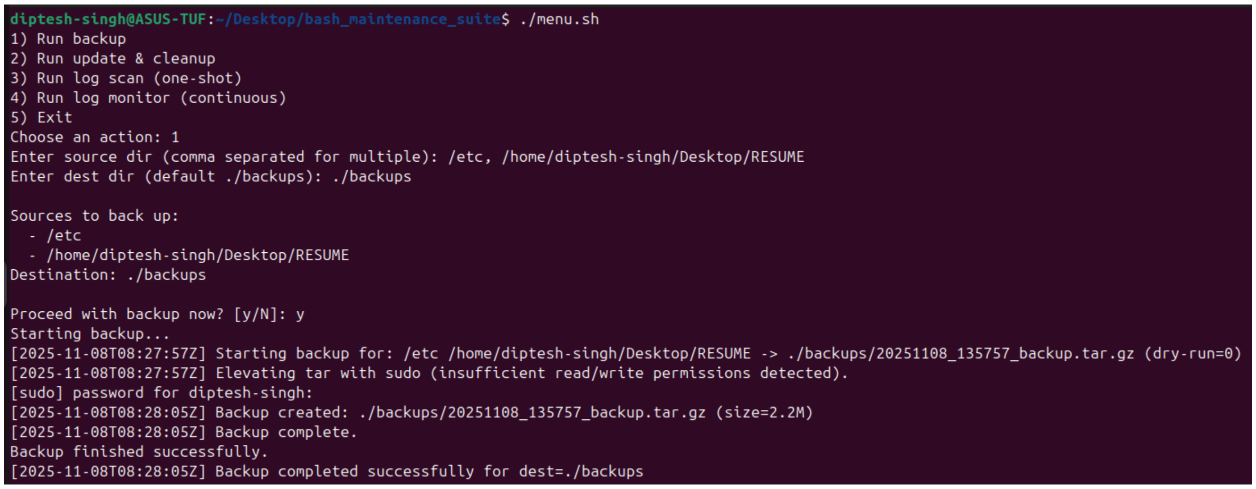
CODE — **system_backup.sh**

```
#!/usr/bin/env bash
# system_backup.sh - simple timestamped tar backup with retention
# Usage: ./system_backup.sh -s /path -s /another -d /dest -r 14 -n
set -o errexit
set -o nounset
set -o pipefail

SCRIPT_DIR=$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)
source "$SCRIPT_DIR/utils.sh" || true

SRC=()
DEST="${DEST:-./backups}"
RETENTION_DAYS=${RETENTION_DAYS:-14}
DRY_RUN=0

usage() {
  echo "Usage: $0 -s /src -s /src2 -d /dest -r retention_days [-n dry-run]"
  exit 1
}

while getopts ":s:d:r:n" opt; do
  case $opt in
    s) SRC+=("$OPTARG") ;;
    d) DEST="$OPTARG" ;;
    r) RETENTION_DAYS="$OPTARG" ;;
    n) DRY_RUN=1 ;;
    *) usage ;;
  esac
done

if [ ${#SRC[@]} -eq 0 ]; then
  echo "No source directories provided. Use -s /path"
  usage
fi


TIMESTAMP=$(date +"%Y%m%d_%H%M%S")
ARCHIVE_NAME="${TIMESTAMP}_backup.tar.gz"
ARCHIVE_PATH="$DEST/$ARCHIVE_NAME"

log "Starting backup for: ${SRC[*]} -> $ARCHIVE_PATH (dry-run=$DRY_RUN)"

if [ "$DRY_RUN" -eq 1 ]; then
  echo "DRY RUN: tar -czf $ARCHIVE_PATH ${SRC[*]}"
  exit 0
fi
```

```
# Decide if tar needs sudo:
# - any source unreadable by current user
# - OR destination directory not writable by current user (so tar can't create the archive)
NEED_SUDO_TAR=0
DEST_SUDO=0
for s in "${SRC[@]}"; do
  # If top-level source is not readable → sudo needed
  if [ ! -r "$s" ]; then
    NEED_SUDO_TAR=1
    break
  fi

  # if path is a system directory, force sudo
  case "$s" in
    /etc*|/var/log*|/root*|/usr/local*|/var/backups*)
      NEED_SUDO_TAR=1
      break
      ;;
  esac
done

# Destination writability check
if [ -e "$DEST" ]; then
  if [ ! -w "$DEST" ]; then
    NEED_SUDO_TAR=1
  fi
else
  parent=$(dirname "$DEST")
  if [ ! -w "$parent" ]; then
    NEED_SUDO_TAR=1
    DEST_SUDO=1
  else
    :
  fi
fi

# --- Ensure destination exists (create with correct privileges) ---
if [ -e "$DEST" ]; then
  :
else
  if [ "$DEST_SUDO" -eq 1 ]; then
    echo "Creating destination directory with sudo: $DEST"
    sudo mkdir -p -- "$DEST"
  else
    mkdir -p -- "$DEST"
  fi
fi
```

```
# Run tar; only elevate tar when needed. STDERR redirection remains in user shell
# (log stays user-owned).
if [ "$NEED_SUDO_TAR" -eq 1 ]; then
  log "Elevating tar with sudo (insufficient read/write permissions detected)."
  sudo tar -czf "$ARCHIVE_PATH" --warning=no-file-changed \
    --absolute-names "${SRC[@]}" 2>"$LOG_DIR/backup_stderr.log" || {
    log "ERROR: tar failed (sudo). See $LOG_DIR/backup_stderr.log"
    exit 2
  }
else
  tar -czf "$ARCHIVE_PATH" --warning=no-file-changed \
    --absolute-names "${SRC[@]}" 2>"$LOG_DIR/backup_stderr.log" || {
    log "ERROR: tar failed. See $LOG_DIR/backup_stderr.log"
    exit 2
  }
fi

if [ -f "$ARCHIVE_PATH" ]; then
  log "Backup created: $ARCHIVE_PATH (size=$(du -h "$ARCHIVE_PATH" | cut -f1))"
else
  log "ERROR: Expected archive not found: $ARCHIVE_PATH"
  exit 3
fi

# Rotation: delete backups older than retention
find "$DEST" -maxdepth 1 -type f -name "*_backup.tar.gz" -mtime +$RETENTION_DAYS \
  -print -exec rm -f {} \; | while read -r removed; do
  log "Removed old backup: $removed"
done

log "Backup complete."
```

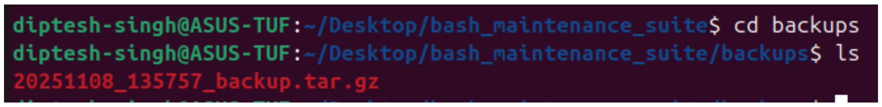
Figure 2: Backup command execution



```
diptesh-singh@ASUS-TUF:~/Desktop/bash_maintenance_suite$ ./menu.sh
1) Run backup
2) Run update & cleanup
3) Run log scan (one-shot)
4) Run log monitor (continuous)
5) Exit
Choose an action: 1
Enter source dir (comma separated for multiple): /etc, /home/diptesh-singh/Desktop/RESUME
Enter dest dir (default ./backups): ./backups

Sources to back up:
  - /etc
  - /home/diptesh-singh/Desktop/RESUME
Destination: ./backups

Proceed with backup now? [y/N]: y
Starting backup...
[2025-11-08T08:27:57Z] Starting backup for: /etc /home/diptesh-singh/Desktop/RESUME -> ./backups/20251108_135757_backup.tar.gz (dry-run=0)
[2025-11-08T08:27:57Z] Elevating tar with sudo (insufficient read/write permissions detected).
[sudo] password for diptesh-singh:
[2025-11-08T08:28:05Z] Backup created: ./backups/20251108_135757_backup.tar.gz (size=2.2M)
[2025-11-08T08:28:05Z] Backup complete.
Backup finished successfully.
[2025-11-08T08:28:05Z] Backup completed successfully for dest=./backups
```

(Figure 2 — Running `system_backup.sh` (via `menu.sh`) to start a backup.)

Figure 3: Backup success (archive created)

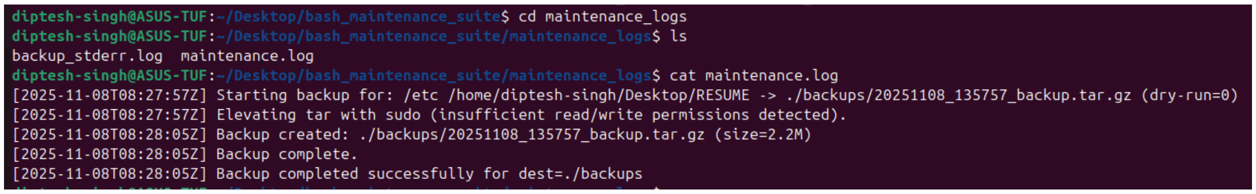


```
diptesh-singh@ASUS-TUF:~/Desktop/bash_maintenance_suite$ cd backups
diptesh-singh@ASUS-TUF:~/Desktop/bash_maintenance_suite/backups$ ls
20251108_135757_backup.tar.gz
```

Figure 2: Backup Success

(Figure 3 — Destination folder listing showing the created timestamped archive.)

Figure 4: Backup log (maintenance.log excerpt)



```
diptesh-singh@ASUS-TUF:~/Desktop/bash_maintenance_suite$ cd maintenance_logs
diptesh-singh@ASUS-TUF:~/Desktop/bash_maintenance_suite/maintenance_logs$ ls
backup_stderr.log  maintenance.log
diptesh-singh@ASUS-TUF:~/Desktop/bash_maintenance_suite/maintenance_logs$ cat maintenance.log
[2025-11-08T08:27:57Z] Starting backup for: /etc /home/diptesh-singh/Desktop/RESUME -> ./backups/20251108_135757_backup.tar.gz (dry-run=0)
[2025-11-08T08:27:57Z] Elevating tar with sudo (insufficient read/write permissions detected).
[2025-11-08T08:28:05Z] Backup created: ./backups/20251108_135757_backup.tar.gz (size=2.2M)
[2025-11-08T08:28:05Z] Backup complete.
[2025-11-08T08:28:05Z] Backup completed successfully for dest=./backups
```

(Figure 4 — Lines from `maintenance_logs/maintenance.log` showing backup start and completion.)

5.3 `system_update_cleanup.sh` — update & cleanup

Purpose: detect the package manager and run update/upgrade/autoremove/autoclean using `safe_sudo()`.

Usage:

```
./system_update_cleanup.sh
```

Behavior notes

- `detect_pkg_mgr()` returns `apt`, `dnf`, `yum`, `pacman`, or `unknown`.

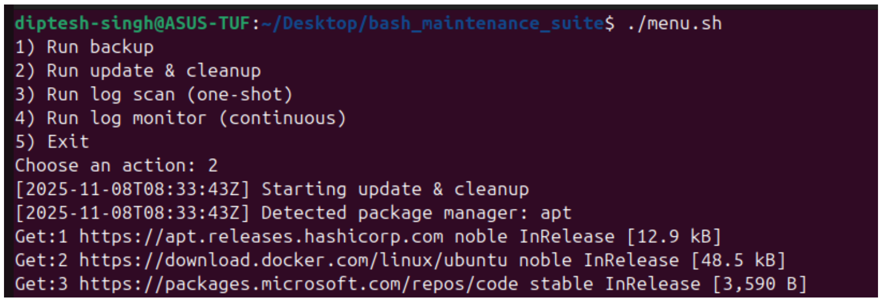
- `safe_sudo` handles calling `sudo` if the script is run by a non-root user.

CODE — `system_update_cleanup.sh`

```bash
#!/usr/bin/env bash
# system_update_cleanup.sh - system update and cleanup helper
set -o errexit
set -o nounset
set -o pipefail

SCRIPT_DIR=$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)
source "$SCRIPT_DIR/utils.sh" || true

log "Starting update & cleanup"
PKG=$(detect_pkg_mgr)
log "Detected package manager: $PKG"

case "$PKG" in
  apt)
    safe_sudo apt-get update
    safe_sudo apt-get -y upgrade
    safe_sudo apt-get -y autoremove
    safe_sudo apt-get -y autoclean
    ;;
  dnf)
    safe_sudo dnf -y upgrade
    safe_sudo dnf -y autoremove
    ;;
  yum)
    safe_sudo yum -y update
    safe_sudo yum -y autoremove || true
    ;;
  pacman)
    safe_sudo pacman -Syu --noconfirm
    ;;
  *)
    log "Unknown package manager. Exiting."
    exit 1
    ;;
esac

log "Update & cleanup finished."
```

Figure 5: Update & cleanup output

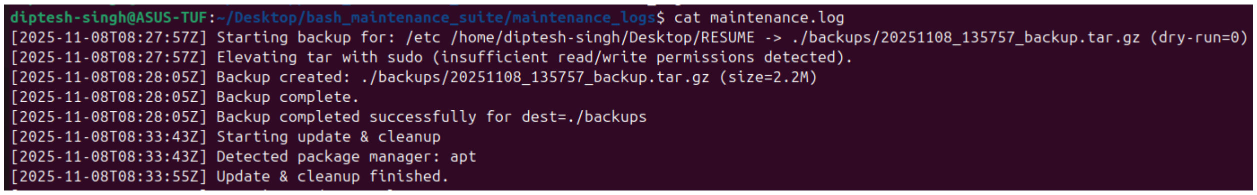
(Figure 5 — Sample output from system_update_cleanup.sh showing package manager activity.)



```
diptesh-singh@ASUS-TUF:~/Desktop/bash_maintenance_suite$ ./menu.sh
1) Run backup
2) Run update & cleanup
3) Run log scan (one-shot)
4) Run log monitor (continuous)
5) Exit
Choose an action: 2
[2025-11-08T08:33:43Z] Starting update & cleanup
[2025-11-08T08:33:43Z] Detected package manager: apt
Get:1 https://apt.releases.hashicorp.com noble InRelease [12.9 kB]
Get:2 https://download.docker.com/linux/ubuntu noble InRelease [48.5 kB]
Get:3 https://packages.microsoft.com/repos/code stable InRelease [3,590 B]
```

Figure 3: Update Cleanup

Figure 6: Update & cleanup logs



```
diptesh-singh@ASUS-TUF:~/Desktop/bash_maintenance_suite/maintenance_logs$ cat maintenance.log
[2025-11-08T08:27:57Z] Starting backup for: /etc /home/diptesh-singh/Desktop/RESUME -> ./backups/20251108_135757_backup.tar.gz (dry-run=0)
[2025-11-08T08:27:57Z] Elevating tar with sudo (insufficient read/write permissions detected).
[2025-11-08T08:28:05Z] Backup created: ./backups/20251108_135757_backup.tar.gz (size=2.2M)
[2025-11-08T08:28:05Z] Backup complete.
[2025-11-08T08:28:05Z] Backup completed successfully for dest=./backups
[2025-11-08T08:33:43Z] Starting update & cleanup
[2025-11-08T08:33:43Z] Detected package manager: apt
[2025-11-08T08:33:55Z] Update & cleanup finished.
```

(Figure 6 — Log entries recorded in `maintenance_logs/maintenance.log` during a system update & cleanup operation.)

5.4 `log_monitoring.sh` — log scan & continuous monitoring

Purpose: one-shot log scan (tail + grep) or continuous monitoring using `tail -F` and pattern matching; write alerts to `maintenance_alerts.log`.

Usage examples:

```
# one-shot
./log_monitoring.sh -f /var/log/syslog -p 'ERROR|Failed'

# continuous
./log_monitoring.sh -f /var/log/syslog -p 'ERROR|Failed' -c

# test with script.sh
./log_monitoring.sh -f ./script.sh -p 'Failed'
```

Behavior notes

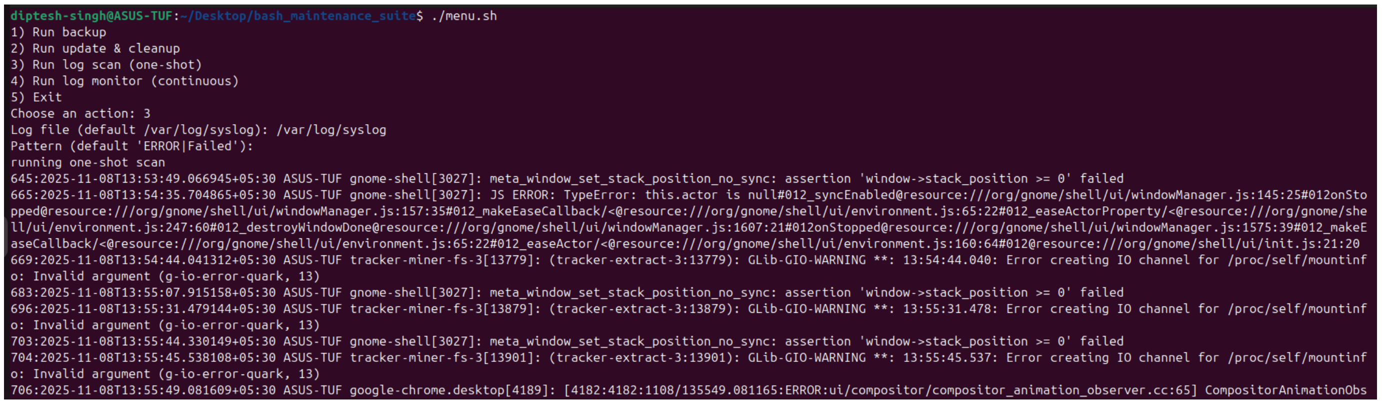
- If the target file is not readable, the script falls back to `sudo` when necessary.
- `ALERT_LOG` defaults to `./maintenance_alerts.log` in the project folder.

CODE — `log_monitoring.sh`

```
#!/usr/bin/env bash
# log_monitoring.sh - search log files for a pattern or run tail -F continuous monitoring
# Usage: ./log_monitoring.sh -f /var/log/syslog -p 'Failed password' -c
set -o errexit
set -o nounset
set -o pipefail

SCRIPT_DIR=$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)
source "$SCRIPT_DIR/utils.sh" || true

LOG_FILE="${LOG_FILE:-/var/log/syslog}"
PATTERN="${PATTERN:-"ERROR|Failed"}"
CONTINUOUS=0

usage() {
  echo "Usage: $0 -f /path/to/log -p 'pattern' [-c continuous]"
  exit 1
}

while getopts ":f:p:c" opt; do
  case $opt in
    f) LOG_FILE="$OPTARG" ;;
    p) PATTERN="$OPTARG" ;;
    c) CONTINUOUS=1 ;;
    *) usage ;;
  esac
done

ALERT_LOG="./maintenance_alerts.log"

run_scan() {
  if [ -r "$LOG_FILE" ]; then
    tail -n 1000 "$LOG_FILE" | grep -Ein "$PATTERN" --color=never | tee -a \
        "$ALERT_LOG" || true
  else
    log "No read permission for $LOG_FILE - using sudo for one-shot scan"
    sudo tail -n 1000 "$LOG_FILE" | grep -Ein "$PATTERN" --color=never | tee -a \
        "$ALERT_LOG" || true
  fi
}

run_continuous() {
  log "Running continuous monitor on $LOG_FILE for pattern: $PATTERN"

  if [ -r "$LOG_FILE" ]; then
    tail -n 0 -F "$LOG_FILE" | while read -r line; do
      if echo "$line" | grep -Eiq "$PATTERN"; then
        echo "[$(date -u +'%Y-%m-%dT%H:%M:%SZ')] ALERT: $line" | tee -a "$ALERT_LOG"
      fi
    done
  else
    log "No read permission for $LOG_FILE - using sudo for continuous monitor"
    sudo tail -n 0 -F "$LOG_FILE" | while read -r line; do
```

```
      if echo "$line" | grep -Eiq "$PATTERN"; then
        echo "[$(date -u +'%Y-%m-%dT%H:%M:%SZ')] ALERT: $line" | tee -a \
            "$ALERT_LOG" >/dev/null
      fi
    done
  fi
}

if [ "$CONTINUOUS" -eq 1 ]; then
  run_continuous
else
  run_scan
fi
```

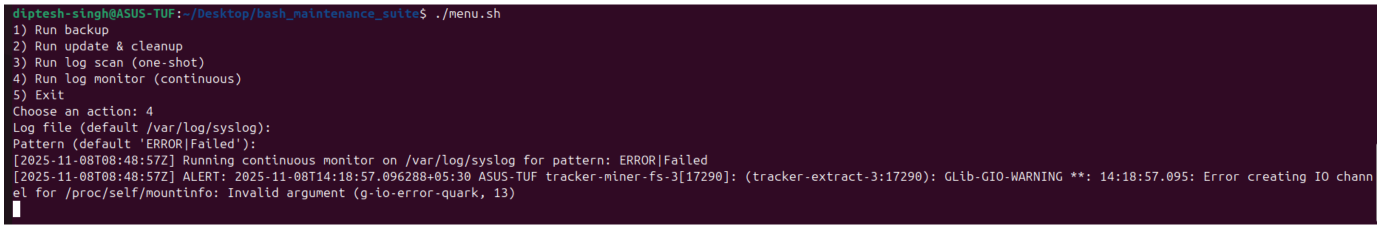
Figure 7: Log scan (one-shot)



```
diptesh-singh@ASUS-TUF:~/Desktop/bash_maintenance_suite$ ./menu.sh
1) Run backup
2) Run update & cleanup
3) Run log scan (one-shot)
4) Run log monitor (continuous)
5) Exit
Choose an action: 3
Log file (default /var/log/syslog): /var/log/syslog
Pattern (default 'ERROR|Failed'):
running one-shot scan
645:2025-11-08T13:53:49.066945+05:30 ASUS-TUF gnome-shell[3027]: meta_window_set_stack_position_no_sync: assertion 'window->stack_position >= 0' failed
665:2025-11-08T13:54:35.704865+05:30 ASUS-TUF gnome-shell[3027]: JS ERROR: TypeError: this.actor is null#012_syncEnabled@resource:///org/gnome/shell/ui/windowManager.js:145:25#012onSto
pped@resource:///org/gnome/shell/ui/windowManager.js:157:35#012_makeEaseCallback/<@resource:///org/gnome/shell/ui/environment.js:65:22#012_easeActorProperty/<@resource:///org/gnome/she
ll/ui/environment.js:247:60#012_destroyWindowDone@resource:///org/gnome/shell/ui/windowManager.js:1607:21#012onStopped@resource:///org/gnome/shell/ui/windowManager.js:1575:39#012_makeE
aseCallback/<@resource:///org/gnome/shell/ui/environment.js:65:22#012_easeActor/<@resource:///org/gnome/shell/ui/environment.js:160:64#012@resource:///org/gnome/shell/ui/init.js:21:20
669:2025-11-08T13:54:44.041312+05:30 ASUS-TUF tracker-miner-fs-3[13779]: (tracker-extract-3:13779): GLib-GIO-WARNING **: 13:54:44.040: Error creating IO channel for /proc/self/mountinf
o: Invalid argument (g-io-error-quark, 13)
683:2025-11-08T13:55:07.915158+05:30 ASUS-TUF gnome-shell[3027]: meta_window_set_stack_position_no_sync: assertion 'window->stack_position >= 0' failed
696:2025-11-08T13:55:31.479144+05:30 ASUS-TUF tracker-miner-fs-3[13879]: (tracker-extract-3:13879): GLib-GIO-WARNING **: 13:55:31.478: Error creating IO channel for /proc/self/mountinf
o: Invalid argument (g-io-error-quark, 13)
703:2025-11-08T13:55:44.330149+05:30 ASUS-TUF gnome-shell[3027]: meta_window_set_stack_position_no_sync: assertion 'window->stack_position >= 0' failed
704:2025-11-08T13:55:45.538108+05:30 ASUS-TUF tracker-miner-fs-3[13901]: (tracker-extract-3:13901): GLib-GIO-WARNING **: 13:55:45.537: Error creating IO channel for /proc/self/mountinf
o: Invalid argument (g-io-error-quark, 13)
706:2025-11-08T13:55:49.081609+05:30 ASUS-TUF google-chrome.desktop[4189]: [4182:4182:1108/135549.081165:ERROR:ui/compositor/compositor_animation_observer.cc:65] CompositorAnimationObs
```

(Figure 7 — One-shot scan output and appended alert written to `maintenance_alerts.log`.)

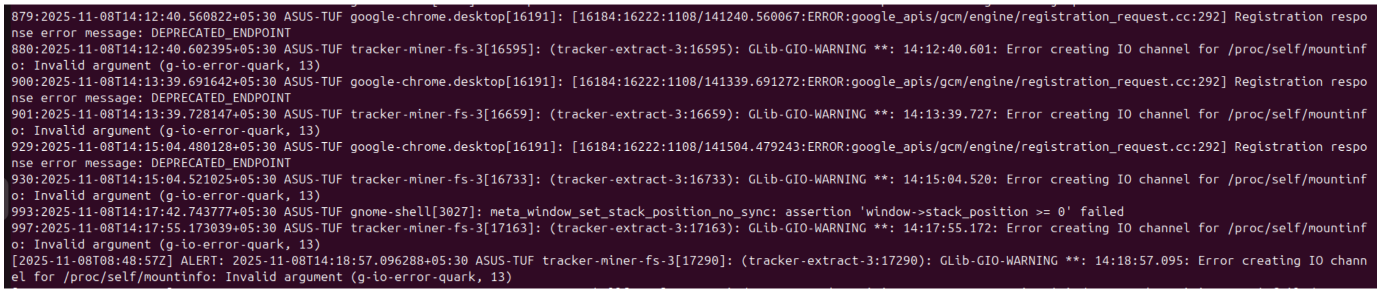
Figure 8: Continuous log monitor (running)



```
diptesh-singh@ASUS-TUF:~/Desktop/bash_maintenance_suite$ ./menu.sh
1) Run backup
2) Run update & cleanup
3) Run log scan (one-shot)
4) Run log monitor (continuous)
5) Exit
Choose an action: 4
Log file (default /var/log/syslog):
Pattern (default 'ERROR|Failed'):
[2025-11-08T08:48:57Z] Running continuous monitor on /var/log/syslog for pattern: ERROR|Failed
[2025-11-08T08:48:57Z] ALERT: 2025-11-08T14:18:57.096288+05:30 ASUS-TUF tracker-miner-fs-3[17290]: (tracker-extract-3:17290): GLib-GIO-WARNING **: 14:18:57.095: Error creating IO chann
el for /proc/self/mountinfo: Invalid argument (g-io-error-quark, 13)
█
```

(Figure 8 — `log_monitoring.sh -c` running, capturing live log lines.)

Figure 9: maintenance_alerts.log contents



```
879:2025-11-08T14:12:40.560822+05:30 ASUS-TUF google-chrome.desktop[16191]: [16184:16222:1108/141240.560067:ERROR:google_apis/gcm/engine/registration_request.cc:292] Registration respo
nse error message: DEPRECATED_ENDPOINT
880:2025-11-08T14:12:40.602395+05:30 ASUS-TUF tracker-miner-fs-3[16595]: (tracker-extract-3:16595): GLib-GIO-WARNING **: 14:12:40.601: Error creating IO channel for /proc/self/mountinf
o: Invalid argument (g-io-error-quark, 13)
900:2025-11-08T14:13:39.691642+05:30 ASUS-TUF google-chrome.desktop[16191]: [16184:16222:1108/141339.691272:ERROR:google_apis/gcm/engine/registration_request.cc:292] Registration respo
nse error message: DEPRECATED_ENDPOINT
901:2025-11-08T14:13:39.728147+05:30 ASUS-TUF tracker-miner-fs-3[16659]: (tracker-extract-3:16659): GLib-GIO-WARNING **: 14:13:39.727: Error creating IO channel for /proc/self/mountinf
o: Invalid argument (g-io-error-quark, 13)
929:2025-11-08T14:15:04.480128+05:30 ASUS-TUF google-chrome.desktop[16191]: [16184:16222:1108/141504.479243:ERROR:google_apis/gcm/engine/registration_request.cc:292] Registration respo
nse error message: DEPRECATED_ENDPOINT
930:2025-11-08T14:15:04.521025+05:30 ASUS-TUF tracker-miner-fs-3[16733]: (tracker-extract-3:16733): GLib-GIO-WARNING **: 14:15:04.520: Error creating IO channel for /proc/self/mountinf
o: Invalid argument (g-io-error-quark, 13)
993:2025-11-08T14:17:42.743777+05:30 ASUS-TUF gnome-shell[3027]: meta_window_set_stack_position_no_sync: assertion 'window->stack_position >= 0' failed
997:2025-11-08T14:17:55.173039+05:30 ASUS-TUF tracker-miner-fs-3[17163]: (tracker-extract-3:17163): GLib-GIO-WARNING **: 14:17:55.172: Error creating IO channel for /proc/self/mountinf
o: Invalid argument (g-io-error-quark, 13)
[2025-11-08T08:48:57Z] ALERT: 2025-11-08T14:18:57.096288+05:30 ASUS-TUF tracker-miner-fs-3[17290]: (tracker-extract-3:17290): GLib-GIO-WARNING **: 14:18:57.095: Error creating IO chann
el for /proc/self/mountinfo: Invalid argument (g-io-error-quark, 13)
```

(Figure 9 — Content of `maintenance_alerts.log` showing captured alerts.)

5.5 `utils.sh` — shared helpers

Purpose: shared utility functions used by all scripts:

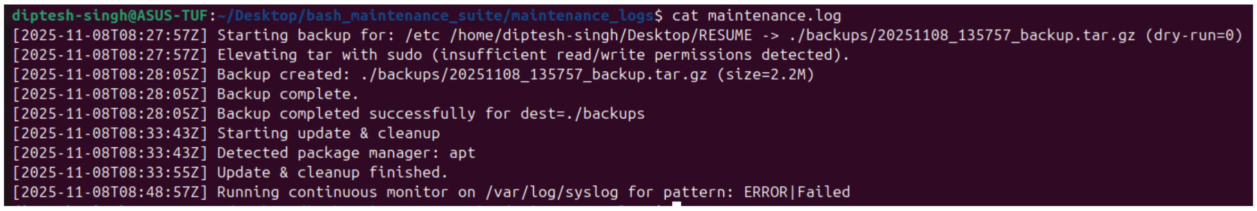
- `log()` — timestamped writing to `${LOG_DIR:-./maintenance_logs}/maintenance.log`.
- `detect_pkg_mgr()` — returns package manager name.
- `safe_sudo()` — call a command with `sudo` only if not root.

CODE — `utils.sh`

```bash
#!/usr/bin/env bash
# utils.sh - helper functions for maintenance scripts
set -o errexit
set -o nounset
set -o pipefail

LOG_DIR="${LOG_DIR:-./maintenance_logs}"
mkdir -p "$LOG_DIR"

log() {
    # usage: log "message"
    local ts
    ts=$(date -u +"%Y-%m-%dT%H:%M:%SZ")
    echo "[$ts] $*" | tee -a "$LOG_DIR/maintenance.log"
}

detect_pkg_mgr() {
    if command -v apt-get >/dev/null 2>&1; then
        echo "apt"
    elif command -v dnf >/dev/null 2>&1; then
        echo "dnf"
    elif command -v yum >/dev/null 2>&1; then
        echo "yum"
    elif command -v pacman >/dev/null 2>&1; then
        echo "pacman"
    else
```

```
        echo "unknown"
    fi
}

safe_sudo() {
    # wrap commands that may need sudo; call like: safe_sudo apt-get update
    if [ "$EUID" -ne 0 ]; then
        sudo "$@"
    else
        "$@"
    fi
}
```

Figure 10: maintenance.log sample entries



```
diptesh-singh@ASUS-TUF:~/Desktop/bash_maintenance_suite/maintenance_logs$ cat maintenance.log
[2025-11-08T08:27:57Z] Starting backup for: /etc /home/diptesh-singh/Desktop/RESUME -> ./backups/20251108_135757_backup.tar.gz (dry-run=0)
[2025-11-08T08:27:57Z] Elevating tar with sudo (insufficient read/write permissions detected).
[2025-11-08T08:28:05Z] Backup created: ./backups/20251108_135757_backup.tar.gz (size=2.2M)
[2025-11-08T08:28:05Z] Backup complete.
[2025-11-08T08:28:05Z] Backup completed successfully for dest=./backups
[2025-11-08T08:33:43Z] Starting update & cleanup
[2025-11-08T08:33:43Z] Detected package manager: apt
[2025-11-08T08:33:55Z] Update & cleanup finished.
[2025-11-08T08:48:57Z] Running continuous monitor on /var/log/syslog for pattern: ERROR|Failed
```

(Figure 10 — Sample entries from the `maintenance_logs/maintenance.log` showing logged actions.)

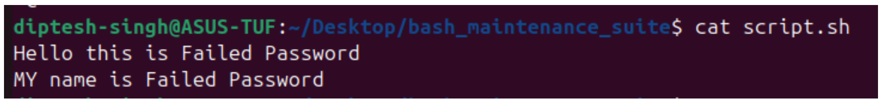
5.6 `script.sh` — sample test log file

Purpose: a developer/test file used for simulating log entries during testing and demo. Useful for demonstrating the log monitor without needing to modify system logs.

Sample content (`script.sh`):

```
Hello this is Failed Password
MY name is Failed Password
```

Figure 11: script.sh sample (for testing)



```
diptesh-singh@ASUS-TUF:~/Desktop/bash_maintenance_suite$ cat script.sh
Hello this is Failed Password
MY name is Failed Password
```

Figure 4: script.sh sample

(Figure 11 — The `script.sh` file used for testing log monitoring.)

---

6. Automating with Cron

This maintenance suite can be automated using **cron jobs**, allowing Linux systems to run backups, system cleanup, and log monitoring on a schedule without user intervention. Cron ensures that essential maintenance tasks occur regularly, improving system stability and reducing manual effort.

Key Principles for Using Cron with This Suite

- Always use **absolute paths** when scheduling scripts via cron.
- Ensure execution permission is set:

  ```
  chmod +x /absolute/path/to/*.sh
  ```

- Direct logs are already handled inside the scripts, so cron does not need extra redirection unless desired.

Example Cron Jobs

1. Daily System Backup (Runs every day at 1:30 AM)

```
30 01 * * * /home/user/maintenance_suite/system_backup.sh -s \
    /home/user/data -d /home/user/backups -r 14
```

This performs a daily backup of `/home/user/data`, keeps backups for **14 days**, and stores logs in `maintenance_logs/`.

---

2. Weekly System Update & Cleanup (Runs every Sunday at 8:00 AM)

```
00 08 * * 0 /home/user/maintenance_suite/system_update_cleanup.sh
```

This keeps your system up to date and removes unnecessary packages weekly.

---

3. Continuous Log Monitoring at Startup (Optional)

```
@reboot /home/user/maintenance_suite/log_monitoring.sh -f \
    /var/log/syslog -p "ERROR|Failed" -c
```

This ensures log alerts are captured from the time the system starts.

Tip

If a cron-triggered job requires elevated privileges, use `sudo crontab -e` instead of standard `crontab -e`.

Conclusion

The **Bash Maintenance Suite** successfully automates key Linux maintenance tasks including system backups, updates, and log monitoring. It provides a simple, modular, and user-friendly solution that reduces manual effort and improves system reliability. With minimal privilege usage, clear logging, and optional cron automation, this tool is practical for regular system maintenance and can be easily extended with additional features if required.
