

# Linux Lab Midterm Project

---

## Linux Lab Midterm Project

---

### Daily User Log Archiver

**Student Name:** Hrithvik Bhardwaj

**Course:** B.Tech CSE — Linux Lab

**Project Type:** Shell Scripting

**Date:** October 2025

---

### Goal

Create a **shell script** that:

- Logs current system information (user, date, processes, disk usage)
  - Rotates and archives old logs weekly
  - Runs automatically every day using a **cron job**
- 

### Implementation Details

#### 1. Identify User

The script identifies the user executing it using:

```
whoami
```

Example:

```
echo "User: $(whoami)"
```

---

#### 2. File Management

Logs are stored in:

```
~/daily_logs/log_YYYY-MM-DD.txt
```

A new file is created every day, containing:

- Current date and time
- Logged-in user
- System uptime

- Top 5 CPU-consuming processes
  - Disk usage summary
  - Kernel/system messages (via `dmesg`)
- 

### 3. Archiving

- Logs older than **7 days** are moved to `~/daily_logs/archive`
- Every **Sunday**, the archive directory is compressed into:

```
weekly_logs_YYYY-WW.tar.gz
```

---

### 4. Loop and Conditions

The script uses a loop to check each log's age and move it if older than 7 days:

```
for file in log_*.txt; do
    if [ condition-to-check-age ]; then
        mv "$file" archive/
    fi
done
```

Optimized version uses:

```
find "${LOG_DIR}" -maxdepth 1 -type f -name 'log_*.txt' -mtime +7 -print0 | while
IFS= read -r -d $'\0' file; do
    mv "$file" "${ARCHIVE_DIR}/"
done
```

---

## Full Shell Script

```
#!/usr/bin/env bash
# daily_log.sh – Daily User Log Archiver

set -u

BASE_DIR="${HOME}/daily_logs"
LOG_DIR="${BASE_DIR}/logs"
ARCHIVE_DIR="${BASE_DIR}/archive"
LOCKDIR="${BASE_DIR}/.lock"
WEEKLY_DIR="${BASE_DIR}"
TODAY="$(date +%Y-%m-%d)"
LOGFILE="log_${TODAY}.txt"
LOGPATH="${LOG_DIR}/${LOGFILE}"
WEEK_TAG="$(date +%Y-%V)"
```

```
mkdir -p "${LOG_DIR}" "${ARCHIVE_DIR}" || { echo "Failed to create log dirs"; exit 1; }
```

```
acquire_lock() {  
    if mkdir "${LOCKDIR}" 2>/dev/null; then  
        return 0  
    else  
        echo "Another instance is running."  
        exit 1  
    fi  
}
```

```
release_lock() {  
    rm -rf "${LOCKDIR}"  
}
```

```
write_log() {  
    {  
        echo "=== Daily User Log ==="  
        echo "Date: $(date --iso-8601=seconds)"  
        echo "User: $(whoami)"  
        echo  
        echo "--- Uptime ---"  
        uptime  
        echo  
        echo "--- Top 5 CPU processes ---"  
        ps -eo pid,comm,%cpu,%mem --sort=-%cpu | head -n 6  
        echo  
        echo "--- Disk Usage ---"  
        df -h  
        echo  
        echo "--- Log Directory Size---"  
        du -sh "${LOG_DIR}" 2>/dev/null  
        echo  
        echo "--- Recent Kernel Messages ---"  
        if command -v dmesg >/dev/null 2>&1; then  
            dmesg | tail -n 50  
        else  
            echo "(dmesg not available)"  
        fi  
        echo  
        echo "=== End of Log ==="  
    } > "${LOGPATH}"  
    echo "Log written: ${LOGPATH}"  
}
```

```

}

rotate_old_logs() {
    find "${LOG_DIR}" -maxdepth 1 -type f -name 'log_*.txt' -mtime +7 -print0 | while
IFS= read -r -d $'\0' file; do
        mv -v "$file" "${ARCHIVE_DIR}/"
    done
}

weekly_archive_if_sunday() {
    if [ "$(date +%u)" -eq 7 ]; then
        if [ "$(find "${ARCHIVE_DIR}" -type f | wc -l)" -gt 0 ]; then
            TARNAME="${WEEKLY_DIR}/weekly_logs_${WEEK_TAG}.tar.gz"
            tar -czf "${TARNAME}" -C "${ARCHIVE_DIR}" .
            echo "Created weekly archive: ${TARNAME}"
            find "${ARCHIVE_DIR}" -type f -delete
            echo "Cleared archive directory."
        else
            echo "No files to archive."
        fi
    fi
}

show_menu() {
    cat <<MENU
Daily Log Archiver Menu:
1) Create today's log
2) Rotate old logs
3) Create weekly archive
4) View latest log
0) Exit
MENU

    read -r choice
    case "${choice}" in
        1) main ;;
        2) rotate_old_logs ;;
        3) weekly_archive_if_sunday ;;
        4) ls -lt "${LOG_DIR}" | head -n1 | xargs -I{} cat "${LOG_DIR}/{}" ;;
        0) release_lock; exit 0 ;;
        *) echo "Invalid choice." ;;
    esac
}

```

```

main() {
    acquire_lock
    write_log
    rotate_old_logs
    weekly_archive_if_sunday
    release_lock
}

if [ "${1:-}" = "menu" ]; then
    acquire_lock
    show_menu
    release_lock
    exit 0
fi

main
exit 0

```

## Cron Job Setup

To schedule the script to run daily at 8 PM:

```
crontab -e
```

Add the line:

```
0 20 * * * /home/user/daily_log.sh >/dev/null 2>&1
```

Confirm with:

```
crontab -l
```

## Directory Structure

```

~/daily_logs/
|
├─ logs/
|   ├─ log_2025-10-01.txt
|   ├─ log_2025-10-02.txt
|   └─ ...
|
├─ archive/
|   └─ log_2025-09-25.txt

```

```
| └─ ...  
|  
└─ weekly_logs_2025-42.tar.gz
```

## Optional Enhancements

### 1. Email Logs Automatically

```
echo "Log attached" | mail -s "Daily Log" -a "$LOGPATH" user@example.com
```

### 2. Error Handling

- Script checks if directories exist before writing.
- Uses a lock (`.lock`) to avoid concurrent runs.

### 3. Interactive Menu

- Case-based menu allows manual logging, archiving, and viewing.

## Commands Used

Feature	Command	Purpose
Identify User	<code>whoami</code>	Get current username
Date Format	<code>date +%Y-%m-%d</code>	Create timestamped filenames
Disk Usage	<code>df -h</code>	Human-readable disk usage
Process List	<code>`ps -eo pid,comm,%cpu,%mem --sort=-%cpu`</code>	<code>head -n 6`</code>
File Search	<code>find . -name "log_*.txt" -mtime +7</code>	Find logs older than 7 days
Archiving	<code>tar -czf</code>	Compress weekly logs
Scheduling	<code>crontab -e</code>	Automate daily execution

## Learning Outcomes

- Practical experience with **shell scripting**, **loops**, and **conditions**
- Understanding of **cron jobs** for automation
- Use of **system commands** (`ps`, `df`, `find`, `tar`)
- Managing **file I/O** and **directory structures**
- Basic **error handling** and **process synchronization**

## Conclusion

The *Daily User Log Archiver* efficiently:

- Automates daily system monitoring
- Reduces manual log management
- Demonstrates file handling, archiving, scheduling, and automation concepts

This project encapsulates key Linux administration skills through scripting.