“Operating Systems”

[Practical]

Project

**Submitted by: Ali Jamshed**

**Roll #: 2k23-BsCs-251**

**Section: Green (G)**

Department

**“Computer Science”**

Shell-Based Graphical Alarm Clock

**1. Introduction**

An alarm clock is a utility application designed to notify or alert the user at a specified time. In this project, we have developed an **Alarm Clock using Shell scripting** integrated with a **graphical user interface (GUI)**. The script allows users to set a specific time at which an audio alarm is triggered. This project showcases the capabilities of shell scripting in handling real-time events, integrating multimedia, and providing user interaction through GUI components.

**2. Objectives**

* To create a fully functional alarm clock using Bash shell scripting.
* To provide a graphical interface for user-friendly input and interaction.
* To implement real-time alarm triggering with precise timing.
* To play a custom MP3 sound file as an alert.
* To allow manual stopping of the alarm using GUI controls.

**3. Tools & Technologies Used:**

| Component | Description |
| --- | --- |
| Bash | Primary scripting language |
| YAD (Yet Another Dialog) | Provides GUI dialogs for input and buttons |
| mpg123 | Command-line tool to play MP3 audio files |
| notify-send | Displays desktop notifications |
| Linux Commands | date, sleep, kill, ps for time and process control |

**4. Installation Requirements:**

****

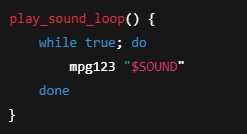
**5. Script Structure & Component Explanation:**

**1. Sound Path Configuration**

****

* Specifies the path of the custom MP3 sound file to be used as the alarm tone.

**2. Alarm Playback Function**

****

* This function plays the selected sound in an infinite loop until the user stops it.
* mpg123 ensures compatibility with MP3 audio formats.

**3. User Input for Alarm Time**

****

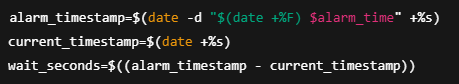
* Opens a **YAD dialog box** for the user to enter the alarm time in HH:MM format.
* If the input is empty, the script exits.

**4. Confirmation Dialog**

****

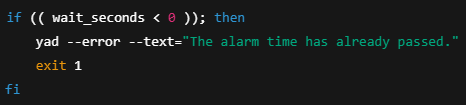
* Notifies the user that the alarm is set using a non-intrusive dialog.

**5. Time Comparison Logic**

****

* Converts the current time and alarm time into seconds since the epoch.
* Computes the exact delay in seconds.

**6. Error Handling**

****

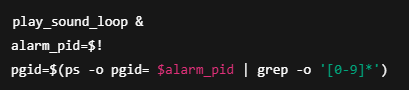
* Checks whether the user set an alarm time that is already in the past and alerts them if so.

**7. Delay Execution Until Alarm**

****

* Waits precisely the computed number of seconds to trigger the alarm.

**8. Trigger Alarm & Capture Process Group**

****

* Starts the alarm sound loop in the background and captures its process group ID to terminate later.

**9. User Interaction to Stop Alarm**

****

* Displays a **YAD window with a stop button**.
* Waits for the user to stop the alarm manually.

**10. Stopping the Alarm Sound**

****

* Kills all processes in the alarm's process group to completely stop the sound.
* Sends a system notification confirming the alarm is turned off.

**5. Features**

* Accepts precise user-defined alarm times
* Plays MP3 sound in loop
* Allows manual termination of alarm
* Accurate to the second
* Uses GUI for user-friendly interaction

**6. Script Breakdown**

**Key Functional Elements:**

| **Functionality** | **Description** |
| --- | --- |
| yad --entry | Takes alarm time input via GUI |
| date -d | Converts user input to Unix timestamp |
| sleep | Delays the script execution until alarm time |
| mpg123 | Plays the alarm MP3 in a continuous loop |
| kill | Terminates all background alarm sound processes |
| notify-send | Displays notification upon alarm dismissal |

**7. Advantages of the Script**

* **Lightweight**: Does not require a full application or framework.
* **Graphical UI**: Enhances usability over traditional command-line-only scripts.
* **Precise Timing**: Utilizes timestamp calculations for accuracy.
* **Customizable Sound**: User can specify any MP3 as the alarm tone.
* **Extensible**: Can be expanded with snooze, countdown, or recurring alarms.

**10. Conclusion**

This project demonstrates the versatility of shell scripting beyond traditional automation tasks. By integrating GUI dialogs, audio playback, and precise time control, we have successfully created a simple, interactive, and effective alarm clock application. This solution reflects how even basic shell tools can be used to develop practical, user-friendly applications when combined creatively.

=========================================================================================