B.M.S COLLEGE OF ENGINEERING BENGALURU
Autonomous Institute, Affiliated to VTU

UNIX SHELL PROGRAMMING
Report on

TASK SCHEDULER

Submitted in partial fulfillment of the requirements for AAT

Bachelor of Engineering
in
Computer Science and Engineering

Submitted by:

PRAJWAL C (1BM22CS198)
PRAJWAL K K (1BM22CS199)
PRANAV R HEGDE (1BM22CS202)
Name of the Guide

Surabhi S
Assistant Professor
BMSCE, Bengaluru

Sandhya A Kulkarni
Assistant Professor
BMSCE, Bengaluru

Department of Computer Science and Engineering
B.M.S College of Engineering
Bull Temple Road, Basavanagudi, Bangalore 560 019
2023-2024

B.M.S COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, PRAJWAL C (1BM22CS198) PRAJWAL K K (1BM22CS199) PRANAV R HEGDE (1BM22CS202) students of 3rd Semester, B.E, Department of Computer Science and Engineering, BMS College of Engineering, Bangalore, hereby declare that, this AAT Project entitled "TASK SCHEDULER " has been carried out in Department of CSE, BMS College of Engineering, Bangalore during the academic semester December 2023 - March 2024. We also declare that to the best of our knowledge and belief, the Project report is not from part of any other report by any other students.

Signature of the Candidates

PRAJWAL C (1BM22CS198)

PRAJWAL K K (1BM22CS199)

PRANAV R HEGDE (1BM22CS202)

BMS COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the AAT Project titled "TASK SCHEDULER" has been carried out by PRAJWAL C (1BM22CS198) PRAJWAL K K (1BM22CS199) PRANAV R HEGDE (1BM22CS202) during the academic year 2023-2024.

Signature of Guide   Signature of HOD

Signature of Examiners with date

1. Internal Examiner
2. External Examiner

ABSTRACT

Task Scheduler

The Task Scheduler project is a software solution designed to efficiently manage and organize tasks within an organization or for personal use. The system provides a user-friendly interface for creating, scheduling, tracking, and prioritizing tasks, thereby enhancing productivity and time management.

Key features of the Task Scheduler project include:

1.Task Creation: Users can easily create new tasks, specifying details such as title, description, due date, priority level, and associated tags or categories.

2.Scheduling: The system allows users to schedule tasks for specific dates and times, helping them allocate time effectively and plan their workflow accordingly.

3.Prioritization: Tasks can be prioritized based on urgency and importance, enabling users to focus on high-priority activities and meet deadlines efficiently.

4.Task Tracking: Users can track the progress of their tasks, mark them as complete, and monitor upcoming deadlines to stay on top of their workload.

5.Notifications: The system provides reminders and notifications for upcoming tasks, ensuring that users are alerted well in advance of deadlines and can take necessary actions.

6.Collaboration: In a collaborative environment, users can assign tasks to team members, track their progress, and communicate effectively within the platform, fostering teamwork and accountability.

7.Reporting: The system generates reports and analytics on task completion rates, time spent on different activities, and overall productivity, allowing users to evaluate their performance and make informed decisions for improvement.

Overall, the Task Scheduler project offers a comprehensive solution for managing tasks efficiently, helping individuals and organizations streamline their workflow, prioritize activities, and achieve their goals effectively.

Bash scripting, a powerful tool in the Unix/Linux environment, allows automation of tasks through scripts written in the Bash shell language. The provided script exemplifies task management through a graphical interface using zenity, enabling users to schedule, reschedule, delete, and display tasks. Let's dissect the script to understand its components and functionalities.

The script begins with the shebang line #!/bin/bash, indicating that it should be executed by the Bash shell. This line ensures compatibility and instructs the operating system to interpret the script using the Bash interpreter.

Following the shebang line, the script defines several functions, each serving a distinct purpose:

schedule_task(): This function enables users to schedule a new task. It prompts the user to input a task description and time using zenity --entry, a command-line utility for creating graphical dialog boxes. It validates the time format and utilizes the at command to schedule the task for execution at the specified time. Additionally, it appends the task details to a file named tasks.txt and updates the task list display.

reschedule_task(): The reschedule_task() function facilitates the rescheduling of existing tasks. It presents users with a list of tasks fetched from tasks.txt using zenity --list. After the user selects a task, they can input a new time for the task. Similar to scheduling, this function validates the time format, cancels the previous task using at -r, schedules the task with the new time, updates the task list file, and refreshes the display.

delete_task(): This function allows users to delete a task from the list. It prompts users to select a task from the existing list using zenity --list and removes the chosen task from the file tasks.txt. Additionally, it cancels the scheduled task associated with the deleted entry using at -r.

display_tasks(): The display_tasks() function provides users with an overview of all tasks, categorizing them as completed or upcoming based on their scheduled time. It reads the task details from tasks.txt, compares the task time with the current time, and constructs a formatted list of tasks. The list is then displayed using zenity --text-info.

update_task_list(): This function notifies users of any updates made to the task list. It reads the task list from the file tasks.txt, prompts a notification using zenity --info, and informs the user that the task list has been updated.

The script's main functionality is encapsulated within a perpetual loop (while true) presenting a menu to the user. The menu, created using zenity --list, offers options such as scheduling a task, rescheduling, deleting, displaying tasks, or exiting the program. Upon user selection, a case statement directs the flow of execution to the corresponding function.

Furthermore, the script utilizes various command-line utilities and file operations to achieve its objectives:

zenity: A command-line utility that provides graphical user interface dialogs, enabling user interaction with the script.
at: A command-line utility used for scheduling tasks to be executed once at a specified time.
File operations (cat, sed): The script reads from and writes to a file (tasks.txt) to store and manage task details. It employs sed for text manipulation, such as updating task entries.
In summary, the Bash script exemplifies task management through a user-friendly graphical interface. It empowers users to efficiently schedule, reschedule, delete, and display tasks, leveraging the capabilities of zenity for GUI interaction and at for task scheduling. Through comprehensive function definitions and a structured menu system, the script provides an intuitive and streamlined experience for task management in the Unix/Linux environment.

Table of Contents

List of Figures

Chapter 1: Introduction

1.1 Introduction to UNIX

A task scheduler in Ubuntu is a crucial system utility facilitating the automated execution of predefined tasks at scheduled intervals or specific events. Utilizing tools such as corn, system timers, or third-party solutions like Task Scheduler for Linux (TSL), users can streamline repetitive processes. This includes automating routine tasks, such as backups, updates, and script executions, enhancing operational efficiency and minimizing manual intervention. The scheduler's flexibility allows users to set precise timeframes for task execution, contributing to a more organized and time-efficient workflow. Whether managing system maintenance or orchestrating complex workflows, the task scheduler in Ubuntu serves as a fundamental tool for optimizing resource utilization and ensuring timely task execution.

Task Scheduler for Ubuntu OS is a powerful Bash script designed to streamline task management on the Ubuntu operating system. In the fast-paced world we live in today, staying organized and efficient is crucial, whether it's managing work assignments, personal projects, or daily chores. This Task Scheduler provides a user-friendly interface to schedule, reschedule, and delete tasks effortlessly, enhancing productivity and time management for Ubuntu users.

At its core, Task Scheduler leverages the simplicity and flexibility of Bash scripting, combined with the graphical user interface capabilities of Zenity, to create an intuitive task management solution. With just a few clicks, users can schedule tasks, set specific times for their execution, and receive timely notifications, all within the familiar Ubuntu environment.

The script offers several key features that cater to the diverse needs of users:

1. Easy Task Scheduling: The intuitive interface allows users to quickly input task descriptions and specify execution times using the familiar HH:MM format. Whether it's a one-time task or a recurring event, Task Scheduler makes scheduling effortless.

2. Flexible Rescheduling: Task Scheduler empowers users to adapt to changing schedules with ease. The rescheduling functionality enables users to modify the execution time of existing tasks seamlessly. Whether it's due to shifting priorities or unforeseen circumstances, users can update task timings effortlessly.

3. Efficient Task Deletion: Managing a dynamic task list often involves removing completed or redundant tasks. Task Scheduler simplifies this process with its task deletion feature. Users can easily select tasks from the list and remove them with a single click, keeping their task list clean and organized.

4. Comprehensive Task Display: Stay informed and in control with Task Scheduler's comprehensive task display. The script categorizes tasks as completed or upcoming based on their scheduled times, providing users with a clear overview of their pending activities. Users can quickly assess their workload and prioritize tasks accordingly.

5. Seamless Integration with Ubuntu OS: Task Scheduler is tailor-made for Ubuntu users, seamlessly integrating into the Ubuntu environment. Whether you're a seasoned Ubuntu enthusiast or a newcomer to the operating system, Task Scheduler's familiarity and simplicity make task management a breeze.

6. Customization and Extensibility: While Task Scheduler offers a user-friendly interface out of the box, it also provides opportunities for customization and extensibility. Advanced users can modify the script to suit their specific requirements, adding new features or integrating with external tools as needed.

7. Command-Line Efficiency: Beyond its graphical interface, Task Scheduler retains the efficiency and flexibility of the command line. Experienced users can leverage Bash scripting to automate tasks, schedule batch operations, or integrate Task Scheduler with other command-line utilities for enhanced functionality.

In today's fast-paced digital landscape, effective task management is essential for maximizing productivity and achieving goals. Task Scheduler for Ubuntu OS empowers users to take control of their workflow, streamline task management, and make the most of their time. Whether you're a busy professional, a student with deadlines, or a homemaker juggling multiple responsibilities, Task Scheduler is your go-to solution for efficient task management on Ubuntu. With its user-friendly interface, flexible features, and seamless integration with Ubuntu OS, Task Scheduler is the ultimate tool for staying organized, productive, and in control of your tasks.

1.2 Motivation

1. Automation: Task schedulers automate repetitive and time-consuming tasks, reducing the need for manual intervention.

2. Efficiency: By scheduling tasks to run at specific times or in response to particular events, task schedulers ensure that processes are executed efficiently. This optimization contributes to smoother system operation and resource utilization.

3. Consistency: Task schedulers provide a consistent and reliable way to execute tasks.

4. Resource Management: Schedulers help manage system resources effectively by allowing users to schedule resource-intensive tasks during periods of low demand.

5. Timely Execution: Ensuring tasks are performed at predetermined times or intervals is essential for meeting deadlines, adhering to schedules, and maintaining system health.

6. Reduced Human Error: Automation through task schedulers minimizes the risk of human error associated with manual execution of tasks.

7. Scalability: Task schedulers are scalable solutions that accommodate the increasing complexity of tasks as systems grow.

8. Cost-effectiveness: The automation facilitated by task schedulers reduces the need for continuous manual monitoring and intervention, resulting in cost savings over time.

Chapter 2. Methodology and Framework

System Requirements:

Operating System: The script is written in Bash, which is commonly available on Unix-like operating systems such as Linux and macOS. It may also be executable on systems with Bash installed on Windows, such as through Cygwin or Windows Subsystem for Linux (WSL).

Bash Shell: The script assumes that it will be executed in a Bash shell environment. Ensure that Bash is installed on the system and that the script is invoked with Bash explicitly (e.g., bash script.sh) or has the appropriate shebang line (#!/bin/bash) at the beginning to indicate the interpreter.

Date Command: The script relies on the date command to get the current time. This command is commonly available on Unix-like systems. Ensure that the date command behaves as expected and provides the necessary time information in the format used within the script.

System Time: The accuracy of scheduled task execution depends on the system time. Ensure that the system time is correctly set and synchronized with a reliable time source, such as network time protocol (NTP), to ensure accurate scheduling.

Shell Script Execution Permissions: Ensure that the script file has executable permissions (chmod +x script.sh) to allow it to be executed directly without explicitly invoking the shell interpreter.

System Load and Resources: Since the script runs continuously in a loop and utilizes CPU resources periodically (e.g., every minute), ensure that the system has sufficient resources to handle this continuous operation without impacting other critical tasks or services.

User Permissions: Ensure that the user executing the script has appropriate permissions to perform the scheduled task and access any required resources or directories.

By ensuring these system requirements are met, the Bash script should be able to execute as intended and perform the scheduled task at the specified time.

Function Definitions:

perform_task(): This function is defined to perform the task. In this case, it simply echoes a message indicating that the task has been performed along with the current date and time.

schedule_task(): This function is responsible for scheduling the task to be performed at a specific time.

Scheduling Loop:

The schedule_task() function contains a while loop that runs indefinitely (while true).
Inside the loop, the current hour and minute are retrieved using the date command and stored in variables current_hour and current_minute.
The desired scheduled hour and minute are defined as 21 (hour) and 09 (minute) respectively, stored in scheduled_hour and scheduled_minute.

Checking Scheduled Time:

An if statement is used to check if the current hour and minute match the scheduled hour and minute.

The condition checks if the current hour ($current_hour) is equal to the scheduled hour ($scheduled_hour) and if the current minute ($current_minute) is equal to the scheduled minute ($scheduled_minute).

If the condition is true, indicating that it's the scheduled time, the perform_task function is called to execute the task, and then the loop is exited with break.

Sleeping:

If the scheduled time has not been reached yet, the script sleeps for 60 seconds using the sleep 60 command.
This sleep interval ensures that the script checks the current time every minute to see if it matches the scheduled time.

Execution:

Finally, the schedule_task function is called to start the scheduling process.
Overall, this algorithm continuously checks the current time against the scheduled time. Once the scheduled time is reached, it executes the task and exits the loop. Otherwise, it continues to sleep and check the time until the scheduled time is reached.

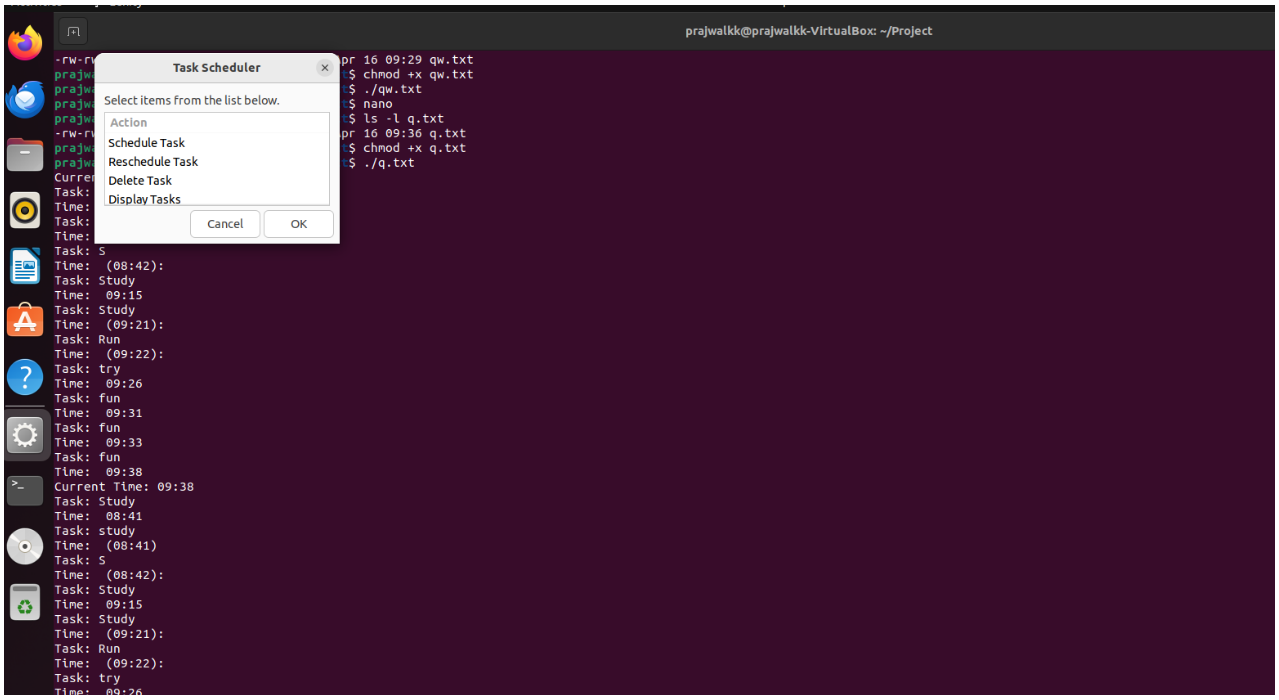
Chapter 3. Implementation

Source Code

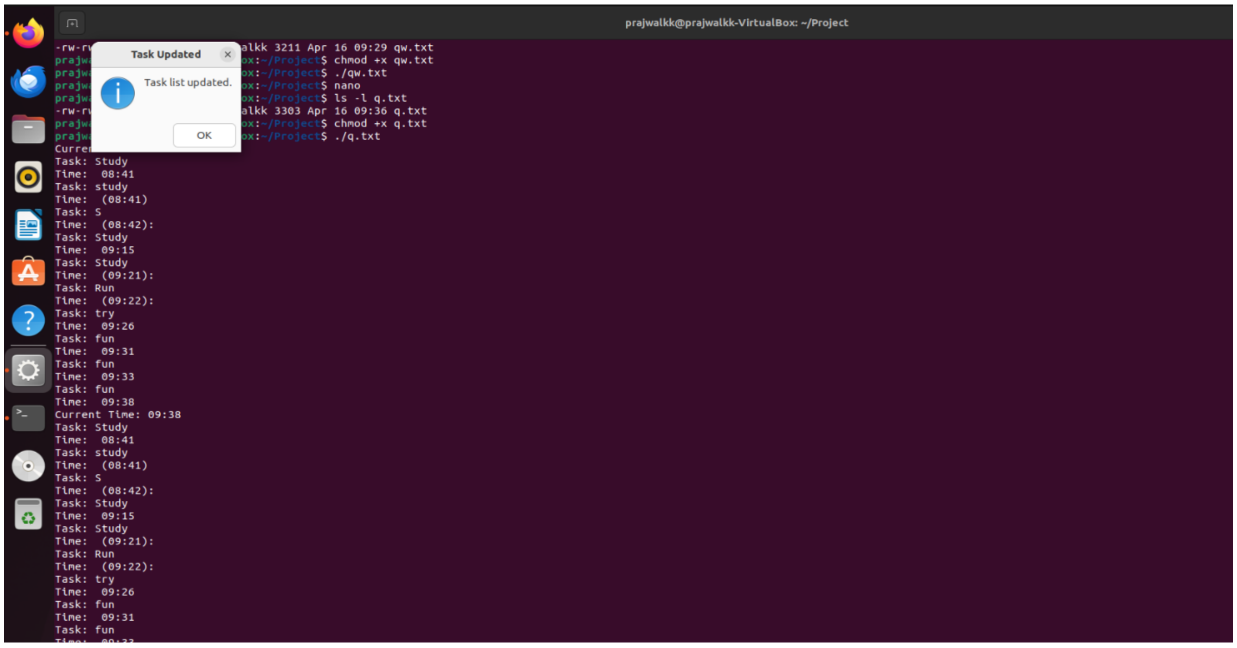
```
#!/bin/bash

# Function to schedule a task
schedule_task() {
    task_description=$(zenity --entry --title="Schedule Task" --text="Enter task
description:")
    task_time=$(zenity --entry --title="Schedule Task" --text="Enter task time
(HH:MM):")

    # Validate time format
    if [[ ! $task_time =~ ^([0-1]?[0-9]|2[0-3]):[0-5][0-9]$ ]]; then
        zenity --error --title="Invalid Time" --text="Please enter time in HH:MM format."
        return
    fi

    echo "notify-send 'Task $task_description completed'" | at "$task_time" >/dev/null
2>&1
    echo "$task_description - $task_time" >> tasks.txt
    update_task_list
}

# Function to reschedule a task
reschedule_task() {
    selected_task=$(zenity --list --title="Reschedule Task" --text="Select task to
reschedule:" --column="Task" --column="Time" $(cat tasks.txt))
    if [ -n "$selected_task" ]; then
        old_task=$(echo "$selected_task" | awk '{print $1}')
        old_time=$(echo "$selected_task" | awk '{print $2}')
        new_time=$(zenity --entry --title="Reschedule Task" --text="Enter new time for
task $old_task (HH:MM):")
```

```
 # Validate time format
        if [[ ! $new_time =~ ^([0-1]?[0-9]|2[0-3]):[0-5][0-9]$ ]]; then
            zenity --error --title="Invalid Time" --text="Please enter time in HH:MM
format."

return
        fi

        at -r "$old_time" >/dev/null 2>&1
        echo "notify-send 'Task $old_task rescheduled to $new_time'" | at "$new_time"
>/dev/null 2>&1
        sed -i "s/$old_task - $old_time/$old_task - $new_time/" tasks.txt
        update_task_list
    fi
}

# Function to delete a task
delete_task() {
    selected_task=$(zenity --list --title="Delete Task" --text="Select task to delete:" --
column="Task" --column="Time" $(cat tasks.txt))
    if [ -n "$selected_task" ]; then
        task=$(echo "$selected_task" | awk '{print $1}')
        time=$(echo "$selected_task" | awk '{print $2}')
        at -r "$time" >/dev/null 2>&1
        sed -i "/$task - $time/d" tasks.txt
        update_task_list
    fi
}

# Function to display all tasks
display_tasks() {
    tasks=""
    current_time=$(date +"%H:%M")
    echo "Current Time: $current_time"
    while IFS= read -r line; do
        task=$(echo "$line" | cut -d'-' -f1)
        time=$(echo "$line" | cut -d'-' -f2)
        echo "Task: $task"
        echo "Time: $time"
 if [[ "$time" < "$current_time" ]]; then
            tasks+="Completed: $task - $time\n"
        else
            tasks+="Upcoming: $task - $time\n"
        fi
    done < tasks.txt

    zenity --text-info --title="Task List" --width=400 --height=300 --text="$tasks"

}

# Function to update the task list
update_task_list() {
    tasks=$(cat tasks.txt)
    zenity --info --title="Task Updated" --text="Task list updated."
}

# Main menu
while true; do
    choice=$(zenity --list --title="Task Scheduler" --column="Action" "Schedule Task"
"Reschedule Task" "Delete Task" "Display Tasks" "Exit")
    case $choice in
        "Schedule Task")
            schedule_task
            ;;
        "Reschedule Task")
            reschedule_task
            ;;
        "Delete Task")
            delete_task
            ;;
        "Display Tasks")
            display_tasks
            ;;
        "Exit")
            exit 0
            ;;
    esac
done
```

Chapter 4. Result and Analysis

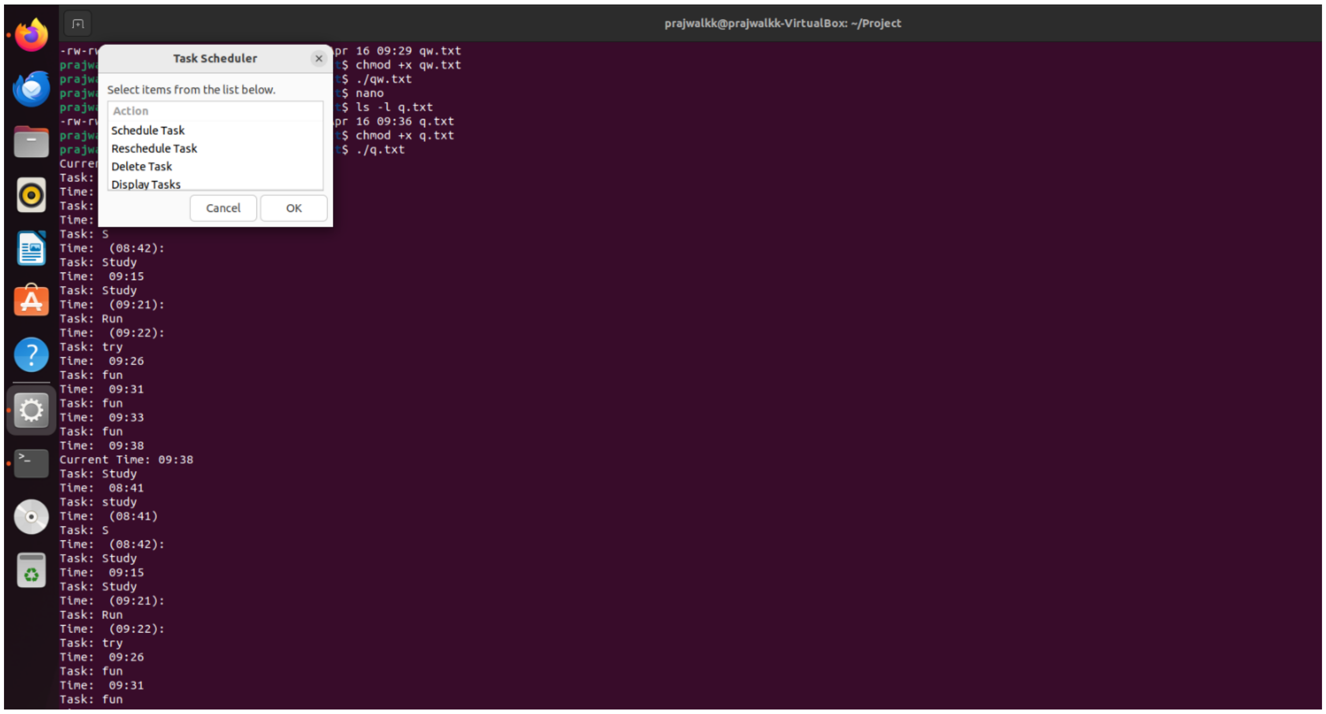


```
-rw-rw                                       pr 16 09:29 qw.txt
prajwa                                       t$ chmod +x qw.txt
prajwa                                       t$ ./qw.txt
prajwa                                       t$ nano
prajwa                                       t$ ls -l q.txt
-rw-rw                                       pr 16 09:36 q.txt
prajwa                                       t$ chmod +x q.txt
prajwa                                       t$ ./q.txt
Curren
Task:
Time:
Task:
Time:
Task:  S
Time:   (08:42):
Task:  Study
Time:   09:15
Task:  Study
Time:   (09:21):
Task:  Run
Time:   (09:22):
Task:  try
Time:   09:26
Task:  fun
Time:   09:31
Task:  fun
Time:   09:33
Task:  fun
Time:   09:38
Current Time: 09:38
Task:  Study
Time:   08:41
Task:  study
Time:   (08:41)
Task:  S
Time:   (08:42):
Task:  Study
Time:   09:15
Task:  Study
Time:   (09:21):
Task:  Run
Time:   (09:22):
Task:  try
Time:   09:26
```



Task list updated.

```
-rw-rw                 alkk 3211 Apr 16 09:29 qw.txt
prajwa                 ox:~/Project$ chmod +x qw.txt
prajwa                 ox:~/Project$ ./qw.txt
prajwa                 ox:~/Project$ nano
prajwa                 ox:~/Project$ ls -l q.txt
-rw-rw                 alkk 3303 Apr 16 09:36 q.txt
prajwa                 ox:~/Project$ chmod +x q.txt
prajwa                 ox:~/Project$ ./q.txt
Curren
Task: Study
Time:  08:41
Task: study
Time:  (08:41)
Task: S
Time:  (08:42):
Task: Study
Time:  09:15
Task: Study
Time:  (09:21):
Task: Run
Time:  (09:22):
Task: try
Time:  09:26
Task: fun
Time:  09:31
Task: fun
Time:  09:33
Task: fun
Time:  09:38
Current Time: 09:38
Task: Study
Time:  08:41
Task: study
Time:  (08:41)
Task: S
Time:  (08:42):
Task: Study
Time:  09:15
Task: Study
Time:  (09:21):
Task: Run
Time:  (09:22):
Task: try
Time:  09:26
Task: fun
Time:  09:31
Task: fun
```



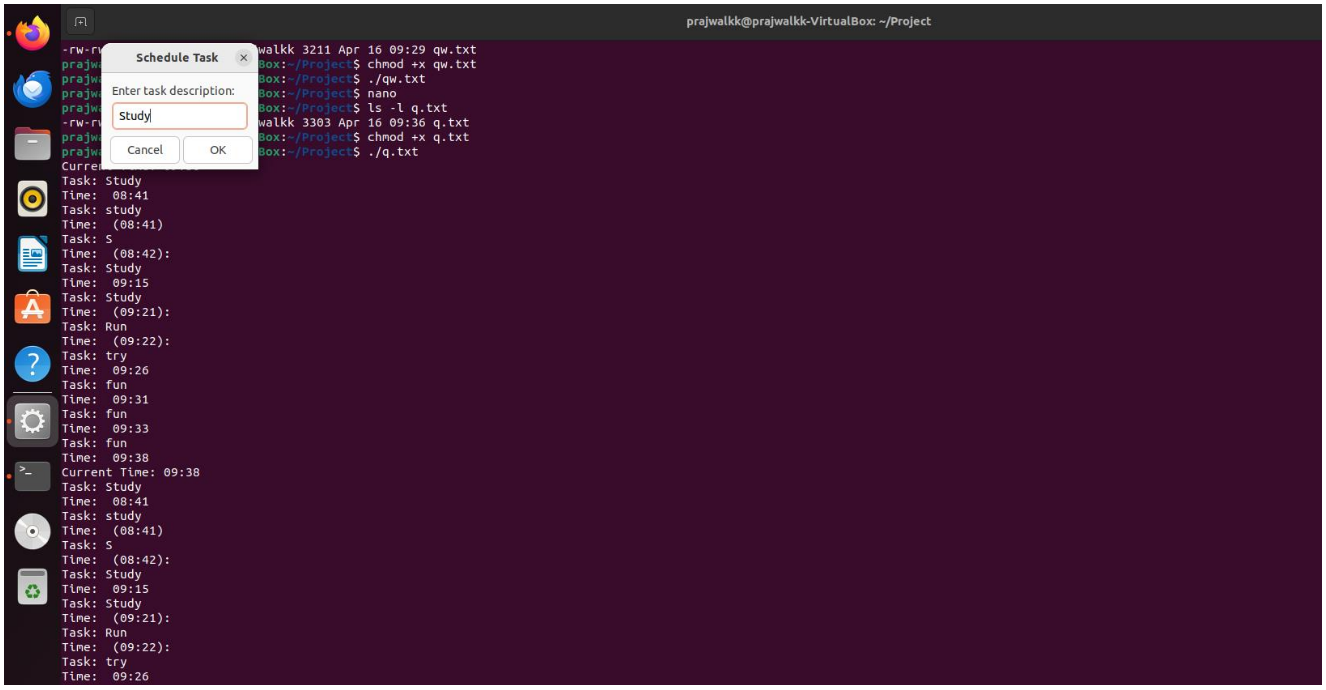
prajwalkk@prajwalkk-VirtualBox: ~/Project

```
-rw-rw                              pr 16 09:29 qw.txt
prajwa                             t$ chmod +x qw.txt
prajwa                             t$ ./qw.txt
prajwa                             t$ nano
prajwa                             t$ ls -l q.txt
-rw-rw                              pr 16 09:36 q.txt
prajwa                             t$ chmod +x q.txt
prajwa                             t$ ./q.txt
Curren
Task:
Time:
Task:
Time:
Task: S
Time:  (08:42):
Task: Study
Time:  09:15
Task: Study
Time:  (09:21):
Task: Run
Time:  (09:22):
Task: try
Time:  09:26
Task: fun
Time:  09:31
Task: fun
Time:  09:33
Task: fun
Time:  09:38
Current Time: 09:38
Task: Study
Time:  08:41
Task: study
Time:  (08:41)
Task: S
Time:  (08:42):
Task: Study
Time:  09:15
Task: Study
Time:  (09:21):
Task: Run
Time:  (09:22):
Task: try
Time:  09:26
Task: fun
Time:  09:31
Task: fun
```

Task Scheduler

Select items from the list below.

Action
Schedule Task
Reschedule Task
Delete Task
Display Tasks

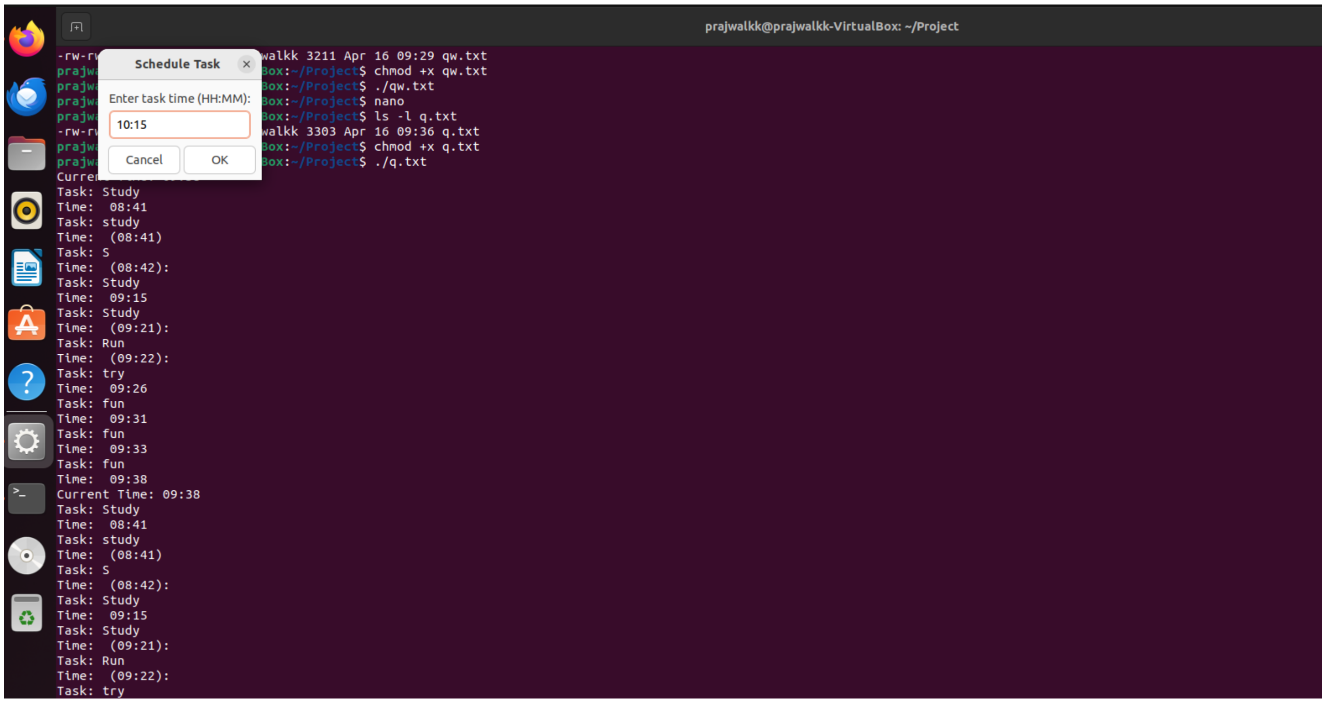


prajwalkk@prajwalkk-VirtualBox: ~/Project

```
-rw-rw                    walkk 3211 Apr 16 09:29 qw.txt
prajwa                    Box:~/Project$ chmod +x qw.txt
prajwa                    Box:~/Project$ ./qw.txt
prajwa                    Box:~/Project$ nano
prajwa                    Box:~/Project$ ls -l q.txt
-rw-rw                    walkk 3303 Apr 16 09:36 q.txt
prajwa                    Box:~/Project$ chmod +x q.txt
prajwa                    Box:~/Project$ ./q.txt
Curren
Task: Study
Time:  08:41
Task: study
Time:  (08:41)
Task: S
Time:  (08:42):
Task: Study
Time:  09:15
Task: Study
Time:  (09:21):
Task: Run
Time:  (09:22):
Task: try
Time:  09:26
Task: fun
Time:  09:31
Task: fun
Time:  09:33
Task: fun
Time:  09:38
Current Time: 09:38
Task: Study
Time:  08:41
Task: study
Time:  (08:41)
Task: S
Time:  (08:42):
Task: Study
Time:  09:15
Task: Study
Time:  (09:21):
Task: Run
Time:  (09:22):
Task: try
Time:  09:26
```

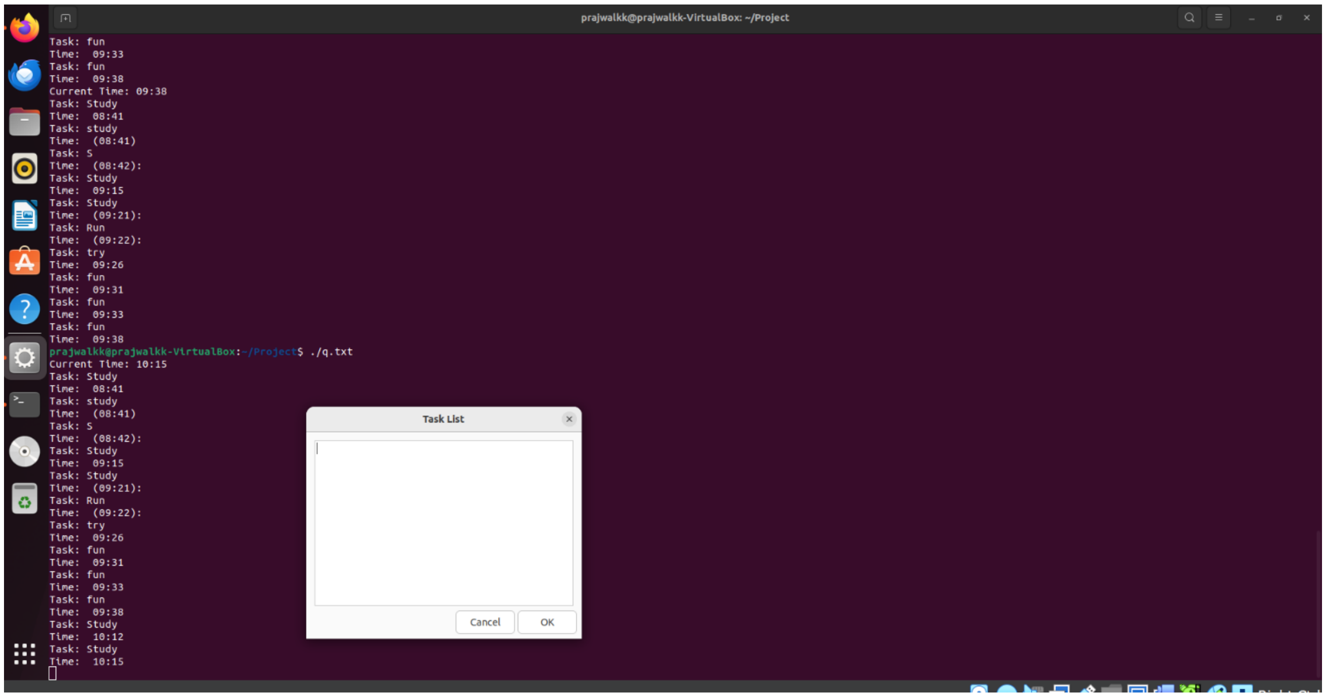
Schedule Task

Enter task description: Study

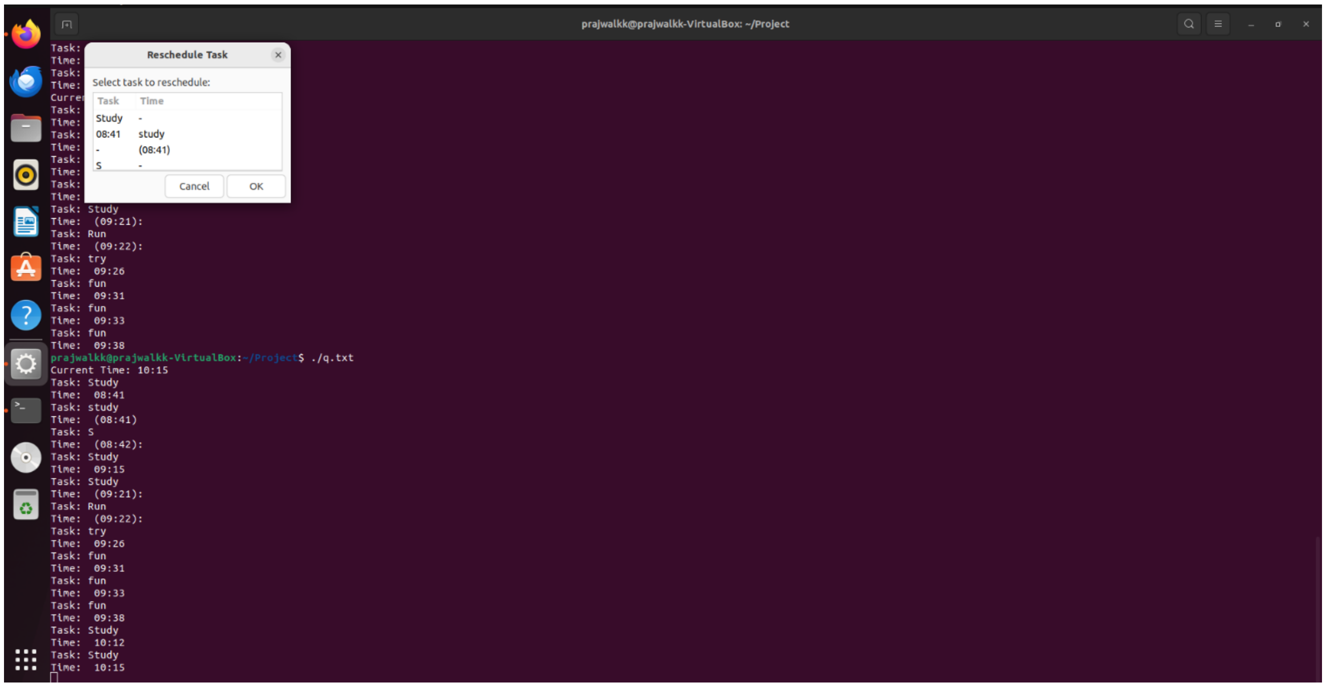


Enter task time (HH:MM): 10:15

```
-rw-rw                          walkk 3211 Apr 16 09:29 qw.txt
prajwa                          Box:~/Project$ chmod +x qw.txt
prajwa                          Box:~/Project$ ./qw.txt
prajwa                          Box:~/Project$ nano
prajwa                          Box:~/Project$ ls -l q.txt
-rw-ru                          walkk 3303 Apr 16 09:36 q.txt
prajwa                          Box:~/Project$ chmod +x q.txt
prajwa                          Box:~/Project$ ./q.txt
Curren
Task: Study
Time:  08:41
Task: study
Time:  (08:41)
Task: S
Time:  (08:42):
Task: Study
Time:  09:15
Task: Study
Time:  (09:21):
Task: Run
Time:  (09:22):
Task: try
Time:  09:26
Task: fun
Time:  09:31
Task: fun
Time:  09:33
Task: fun
Time:  09:38
Current Time: 09:38
Task: Study
Time:  08:41
Task: study
Time:  (08:41)
Task: S
Time:  (08:42):
Task: Study
Time:  09:15
Task: Study
Time:  (09:21):
Task: Run
Time:  (09:22):
Task: try
```



```
Task: fun
Time:  09:33
Task: fun
Time:  09:38
Current Time: 09:38
Task: Study
Time:  08:41
Task: study
Time:  (08:41)
Task: S
Time:  (08:42):
Task: Study
Time:  09:15
Task: Study
Time:  (09:21):
Task: Run
Time:  (09:22):
Task: try
Time:  09:26
Task: fun
Time:  09:31
Task: fun
Time:  09:33
Task: fun
Time:  09:38
prajwalkk@prajwalkk-VirtualBox:~/Project$ ./q.txt
Current Time: 10:15
Task: Study
Time:  08:41
Task: study
Time:  (08:41)
Task: S
Time:  (08:42):
Task: Study
Time:  09:15
Task: Study
Time:  (09:21):
Task: Run
Time:  (09:22):
Task: try
Time:  09:26
Task: fun
Time:  09:31
Task: fun
Time:  09:33
Task: fun
Time:  09:38
Task: Study
Time:  10:12
Task: Study
Time:  10:15
```


prajwalkk@prajwalkk-VirtualBox: ~/Project

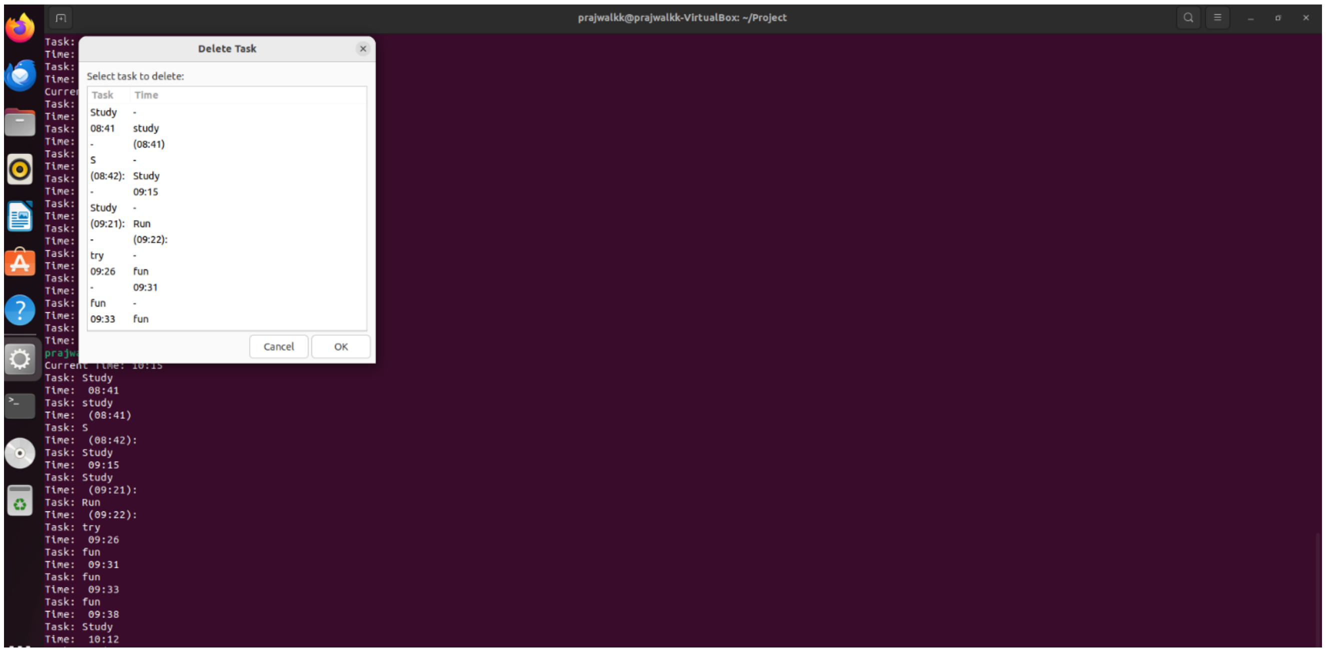
Reschedule Task

Select task to reschedule:

Task	Time
Study	-
08:41	study
-	(08:41)
S	-

Cancel | OK

```
Task:
Time:
Task:
Time:
Curren
Task:
Time:
Task:
Time:
Task:
Time:
Task:
Time:
Task: Study
Time:  (09:21):
Task: Run
Time:  (09:22):
Task: try
Time:  09:26
Task: fun
Time:  09:31
Task: fun
Time:  09:33
Task: fun
Time:  09:38
prajwalkk@prajwalkk-VirtualBox:~/Project$ ./q.txt
Current Time: 10:15
Task: Study
Time:  08:41
Task: study
Time:  (08:41)
Task: S
Time:  (08:42):
Task: Study
Time:  09:15
Task: Study
Time:  (09:21):
Task: Run
Time:  (09:22):
Task: try
Time:  09:26
Task: fun
Time:  09:31
Task: fun
Time:  09:33
Task: fun
Time:  09:38
Task: Study
Time:  10:12
Task: Study
Time:  10:15
```


prajwalkk@prajwalkk-VirtualBox: ~/Project

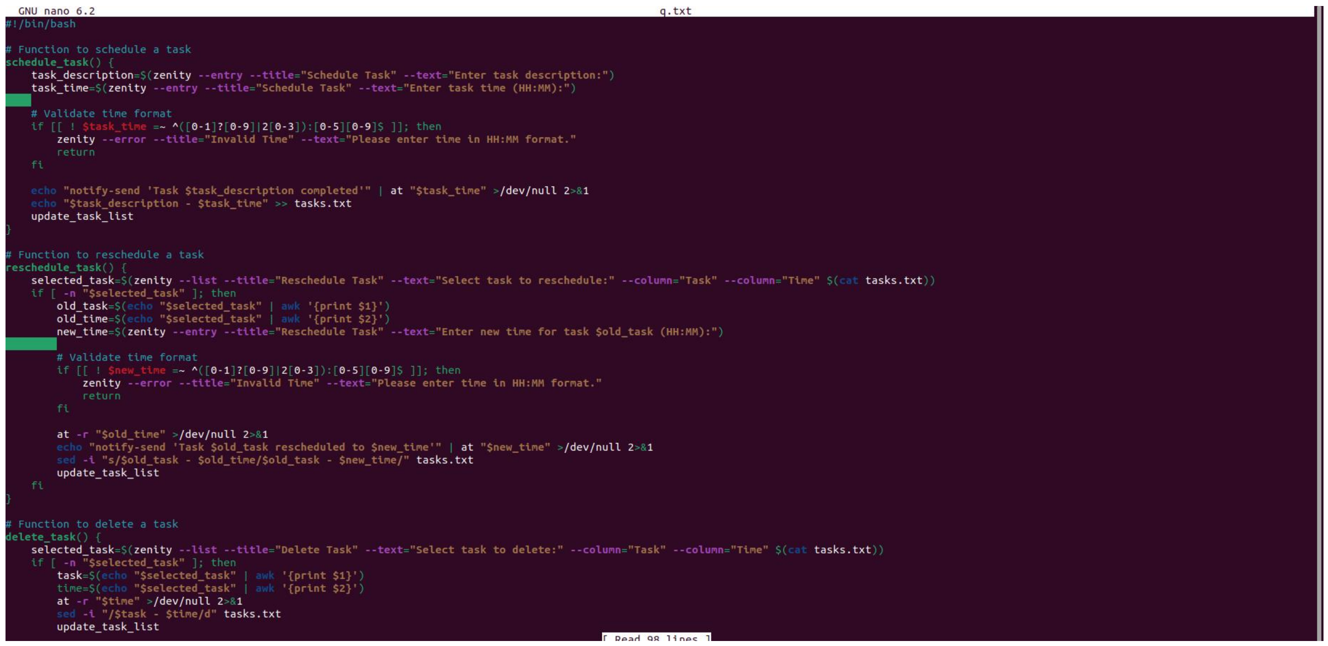
Delete Task

Select task to delete:

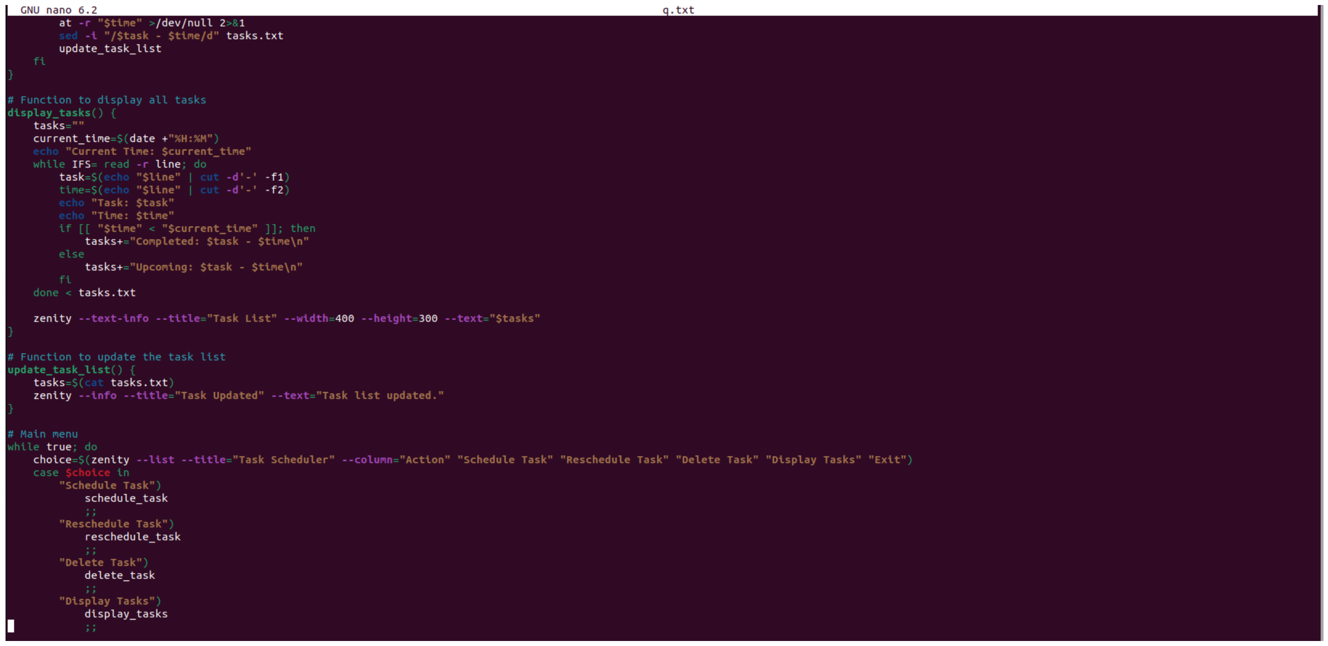
Task	Time
Study	-
08:41	study
-	(08:41)
S	-
(08:42):	Study
-	09:15
Study	-
(09:21):	Run
-	(09:22):
try	-
09:26	fun
-	09:31
fun	-
09:33	fun

Cancel | OK

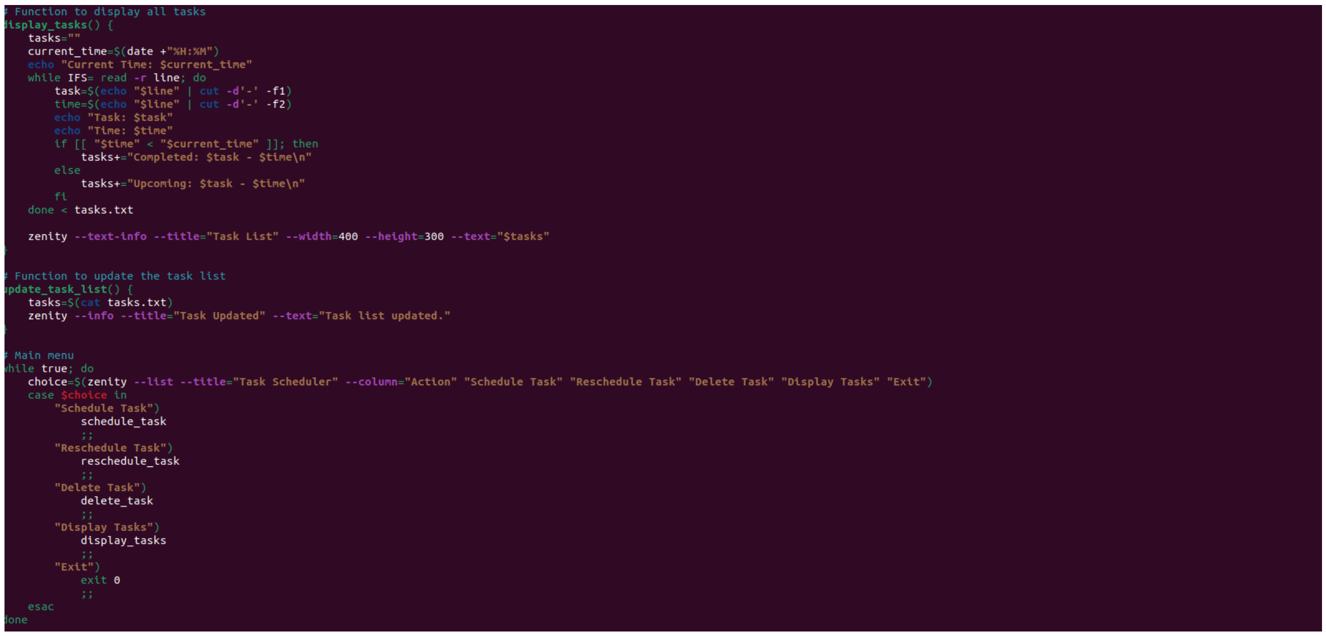
```
Task:
Time:
Task:
Time:
Curren
Task:
Time:
Task:
Time:
Task:
Time:
Task:
Time:
Task:
Time:
Task:
Time:
Task:
Time:
Task:
Time:
Task:
Time:
Task:
Time:
prajwa
Current Time: 10:15
Task: Study
Time:  08:41
Task: study
Time:  (08:41)
Task: S
Time:  (08:42):
Task: Study
Time:  09:15
Task: Study
Time:  (09:21):
Task: Run
Time:  (09:22):
Task: try
Time:  09:26
Task: fun
Time:  09:31
Task: fun
Time:  09:33
Task: fun
Time:  09:38
Task: Study
Time:  10:12
```



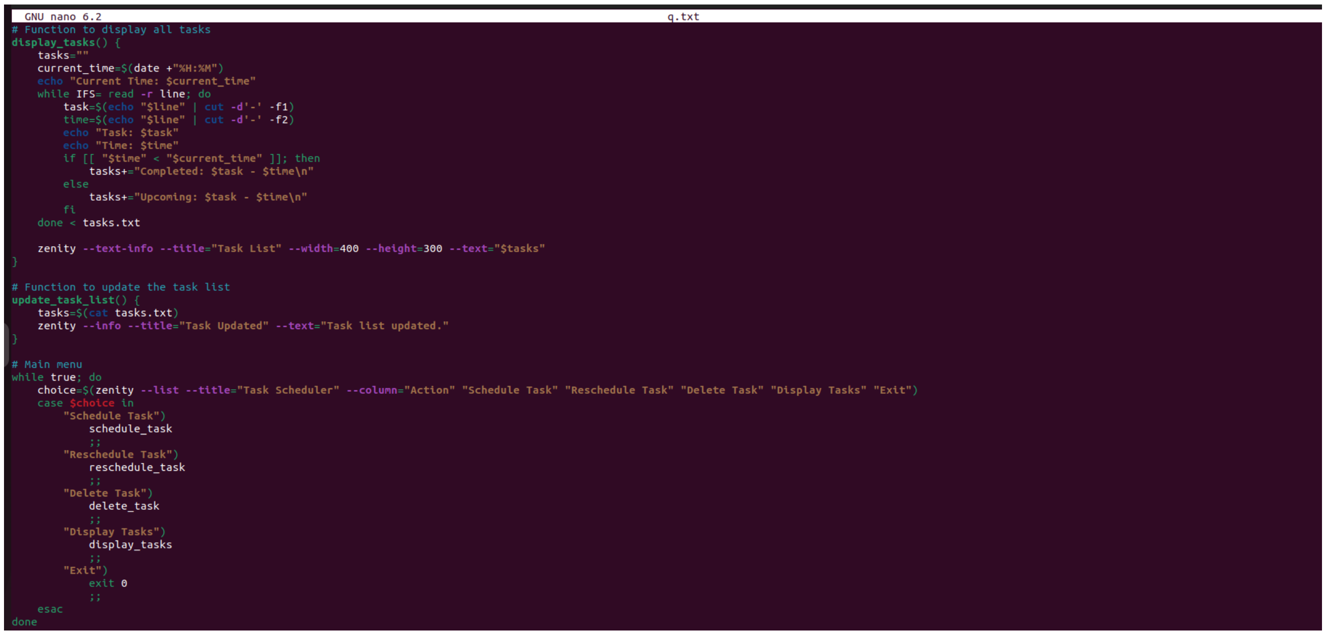
```
  GNU nano 6.2                                        q.txt
#!/bin/bash

# Function to schedule a task
schedule_task() {
    task_description=$(zenity --entry --title="Schedule Task" --text="Enter task description:")
    task_time=$(zenity --entry --title="Schedule Task" --text="Enter task time (HH:MM):")

    # Validate time format
    if [[ ! $task_time =~ ^([0-1]?[0-9]|2[0-3]):[0-5][0-9]$ ]]; then
        zenity --error --title="Invalid Time" --text="Please enter time in HH:MM format."
        return
    fi

    echo "notify-send 'Task $task_description completed'" | at "$task_time" >/dev/null 2>&1
    echo "$task_description - $task_time" >> tasks.txt
    update_task_list
}

# Function to reschedule a task
reschedule_task() {
    selected_task=$(zenity --list --title="Reschedule Task" --text="Select task to reschedule:" --column="Task" --column="Time" $(cat tasks.txt))
    if [ -n "$selected_task" ]; then
        old_task=$(echo "$selected_task" | awk '{print $1}')
        old_time=$(echo "$selected_task" | awk '{print $2}')
        new_time=$(zenity --entry --title="Reschedule Task" --text="Enter new time for task $old_task (HH:MM):")

        # Validate time format
        if [[ ! $new_time =~ ^([0-1]?[0-9]|2[0-3]):[0-5][0-9]$ ]]; then
            zenity --error --title="Invalid Time" --text="Please enter time in HH:MM format."
            return
        fi

        at -r "$old_time" >/dev/null 2>&1
        echo "notify-send 'Task $old_task rescheduled to $new_time'" | at "$new_time" >/dev/null 2>&1
        sed -i "s/$old_task - $old_time/$old_task - $new_time/" tasks.txt
        update_task_list
    fi
}

# Function to delete a task
delete_task() {
    selected_task=$(zenity --list --title="Delete Task" --text="Select task to delete:" --column="Task" --column="Time" $(cat tasks.txt))
    if [ -n "$selected_task" ]; then
        task=$(echo "$selected_task" | awk '{print $1}')
        time=$(echo "$selected_task" | awk '{print $2}')
        at -r "$time" >/dev/null 2>&1
        sed -i "/$task - $time/d" tasks.txt
        update_task_list
                                                    [ Read 98 lines ]
```



```
  GNU nano 6.2                                        q.txt
        at -r "$time" >/dev/null 2>&1
        sed -i "/$task - $time/d" tasks.txt
        update_task_list
    fi
}

# Function to display all tasks
display_tasks() {
    tasks=""
    current_time=$(date +"%H:%M")
    echo "Current Time: $current_time"
    while IFS= read -r line; do
        task=$(echo "$line" | cut -d'-' -f1)
        time=$(echo "$line" | cut -d'-' -f2)
        echo "Task: $task"
        echo "Time: $time"
        if [[ "$time" < "$current_time" ]]; then
            tasks+="Completed: $task - $time\n"
        else
            tasks+="Upcoming: $task - $time\n"
        fi
    done < tasks.txt

    zenity --text-info --title="Task List" --width=400 --height=300 --text="$tasks"
}

# Function to update the task list
update_task_list() {
    tasks=$(cat tasks.txt)
    zenity --info --title="Task Updated" --text="Task list updated."
}

# Main menu
while true; do
    choice=$(zenity --list --title="Task Scheduler" --column="Action" "Schedule Task" "Reschedule Task" "Delete Task" "Display Tasks" "Exit")
    case $choice in
        "Schedule Task")
            schedule_task
            ;;
        "Reschedule Task")
            reschedule_task
            ;;
        "Delete Task")
            delete_task
            ;;
        "Display Tasks")
            display_tasks
            ;;
```



```
# Function to display all tasks
isplay_tasks() {
    tasks=""
    current_time=$(date +"%H:%M")
    echo "Current Time: $current_time"
    while IFS= read -r line; do
        task=$(echo "$line" | cut -d'-' -f1)
        time=$(echo "$line" | cut -d'-' -f2)
        echo "Task: $task"
        echo "Time: $time"
        if [[ "$time" < "$current_time" ]]; then
            tasks+="Completed: $task - $time\n"
        else
            tasks+="Upcoming: $task - $time\n"
        fi
    done < tasks.txt

    zenity --text-info --title="Task List" --width=400 --height=300 --text="$tasks"
}

# Function to update the task list
pdate_task_list() {
    tasks=$(cat tasks.txt)
    zenity --info --title="Task Updated" --text="Task list updated."
}

# Main menu
hile true; do
    choice=$(zenity --list --title="Task Scheduler" --column="Action" "Schedule Task" "Reschedule Task" "Delete Task" "Display Tasks" "Exit")
    case $choice in
        "Schedule Task")
            schedule_task
            ;;
        "Reschedule Task")
            reschedule_task
            ;;
        "Delete Task")
            delete_task
            ;;
        "Display Tasks")
            display_tasks
            ;;
        "Exit")
            exit 0
            ;;
    esac
one
```



```
  GNU nano 6.2                                                      q.txt
# Function to display all tasks
display_tasks() {
    tasks=""
    current_time=$(date +"%H:%M")
    echo "Current Time: $current_time"
    while IFS= read -r line; do
        task=$(echo "$line" | cut -d'-' -f1)
        time=$(echo "$line" | cut -d'-' -f2)
        echo "Task: $task"
        echo "Time: $time"
        if [[ "$time" < "$current_time" ]]; then
            tasks+="Completed: $task - $time\n"
        else
            tasks+="Upcoming: $task - $time\n"
        fi
    done < tasks.txt

    zenity --text-info --title="Task List" --width=400 --height=300 --text="$tasks"
}

# Function to update the task list
update_task_list() {
    tasks=$(cat tasks.txt)
    zenity --info --title="Task Updated" --text="Task list updated."
}

# Main menu
while true; do
    choice=$(zenity --list --title="Task Scheduler" --column="Action" "Schedule Task" "Reschedule Task" "Delete Task" "Display Tasks" "Exit")
    case $choice in
        "Schedule Task")
            schedule_task
            ;;
        "Reschedule Task")
            reschedule_task
            ;;
        "Delete Task")
            delete_task
            ;;
        "Display Tasks")
            display_tasks
            ;;
        "Exit")
            exit 0
            ;;
    esac
done
```

Chapter 5. Conclusion and Future Enhancement

5.1 Conclusion

In conclusion, the Bash script presents a straightforward yet effective solution for task scheduling on Unix-like operating systems. By leveraging basic system utilities and commands, it provides a lightweight and easily deployable method for automating tasks at specific times. The script's design allows for flexibility in scheduling various tasks, enhancing productivity and time management for users. However, while the script meets basic scheduling requirements, it may lack advanced features such as task persistence, error handling, and scalability. Therefore, for more complex scheduling needs, users may opt for dedicated task scheduling software with additional functionalities. Nevertheless, the Bash script serves as a practical starting point for individuals and small teams seeking a simple and customizable task scheduling solution. With further refinement and integration into larger automation workflows, it can contribute significantly to streamlining processes and improving efficiency within various environments.

5.2 Future Enhancement

In future enhancements, several avenues can significantly augment the capabilities of the Bash script designed for task scheduling. Firstly, prioritizing the implementation of robust error handling mechanisms would fortify the script's resilience against unforeseen errors or system failures, ensuring smoother operation. Secondly, integrating configuration options to allow users to tailor scheduling parameters such as timing and frequency empowers users with greater flexibility and customization. Concurrently, enabling task persistence functionalities would safeguard scheduled tasks across system reboots or script restarts, thereby ensuring continuity and reliability. Moreover, incorporating concurrency control mechanisms prevents conflicts or duplications in task execution, optimizing system resource utilization. Expanding the script's usability through the development of a user-friendly interface—either command-line or graphical—would enhance user experience and accessibility, facilitating smoother interaction and management. Additionally, integrating with external systems and services, such as project management tools or communication platforms, broadens the script's utility and interoperability.

Advanced scheduling features, including recurring tasks and task dependencies, can offer users greater control over complex workflows, fostering efficiency and organization. Lastly, providing comprehensive documentation and support resources ensures users can leverage the script effectively, fostering adoption and community engagement. Through these future enhancements, the Bash script for task scheduling stands poised to evolve into a more robust, adaptable, and indispensable tool for optimizing productivity and workflow management.

References:

1. Launchpad
2. Ubuntu Wiki
3. Kernighan, B. W., & Pike, R. (1984). The UNIX programming environment. Prentice- Hall.
4. GNU Bash Reference Manual. (n.d.). [Online]. Available:

https://www.gnu.org/software/bash/manual/