**Time Management System – Project Proposal (C Programming)**

Prasun Rimal

Westcliff University

C Programming

Professor Shrestha

July 25, 2025

**Time Management System – Project (C Programming)**

**About**

This project aims to design and implement a console‑based Time Management System. The system will allow users to log their daily routine tasks, store them persistently, and receive timely reminders through the command‑line interface. The solution focuses on leveraging standard C libraries (e.g., <time.h>, <stdio.h>) and fundamental data structures to create an efficient, lightweight productivity tool that can run on any operating system with a C compiler.

**Reason behind it**

In an era of information overload and hybrid learning/working environments, individuals often struggle to manage their schedules effectively. Building a simple, offline Time Management System in C not only addresses this practical challenge but also offers learners a concrete, real‑world application to practice core programming concepts such as file I/O, dynamic memory allocation, and process timing. Developing the project strengthens problem‑solving skills while delivering a useful productivity aid.

**Target Audience**

- Beginner to intermediate programming students seeking hands‑on experience with C.  
- Individuals who prefer lightweight, command‑line productivity tools.  
- Educators looking for an applied example to teach fundamental concepts in structured programming.

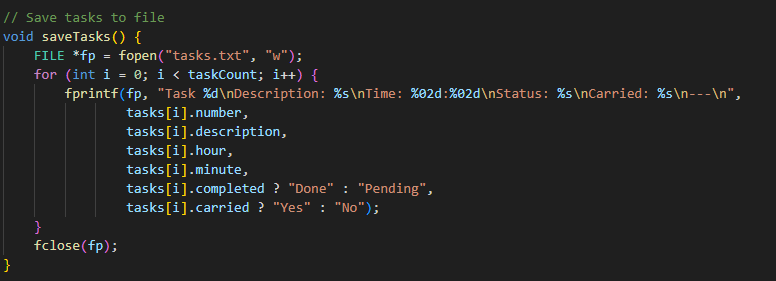
**Learning expectation**

Participants will:  
• Apply modular programming principles using functions and header files.  
• Implement data structures (structs, arrays, and/or linked lists) to manage tasks.  
• Handle persistent storage through text or binary file operations.  
• Utilize the <time.h> library for real‑time scheduling and reminders.  
• Practice debugging strategies and effective documentation.  
• Gain experience with compiling and testing cross‑platform C applications.

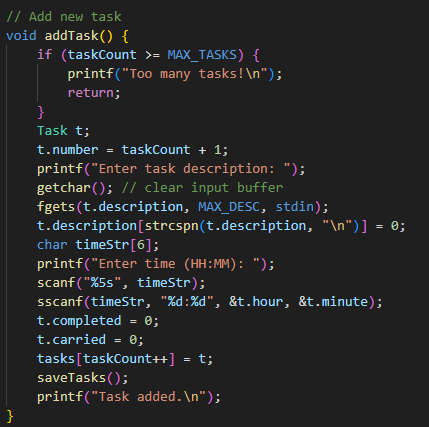
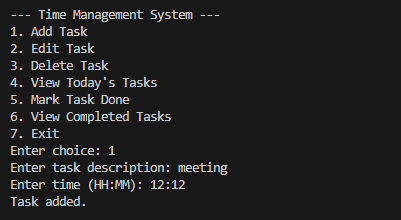
**Features**

The core function of the system is to let users update their daily routines and receive reminders at specified times. Key features include:

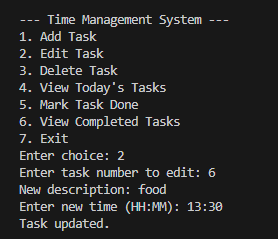
1. Add, edit, and delete routine tasks with descriptions and scheduled times.
2. If someone has remaining tasks it needs to be carried over next days
3. Persistent storage of tasks to maintain data between sessions.
4. Option to view today’s agenda and completed tasks.
5. log of tasks done till now.
6. Taks can be updated as done or deleted.



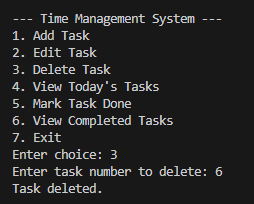
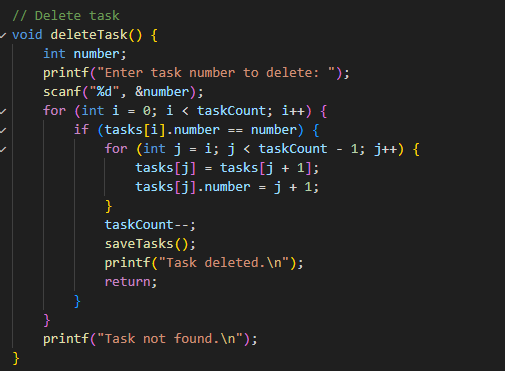
The saveTasks() is the one which saves all the current tasks to a file named as tasks.txt. It starts off by opening the file in write mode where it creates the file in case it does not exist or empties the file in case it already does. It then proceeds to cycle through all the tasks on the task list of a program writing the information of the task in a formatted way. In each task it saves the task number (as it is now titled as task), description, the number of hours spent in HH:MM format, the status is whether it is "Done" or "Pending", and whether the task is carried over to the next day or not ("Yes" or "No"). In order to have the data highly organized and easy to read each task is separated with a line which contains ---. Once all the tasks are written, the function will close the file so as to save all the data well to use it in future. This role serves as the basis to store persistent records of tasks provided in more than one running of the program.



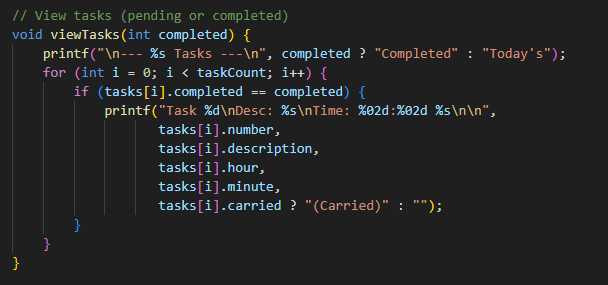
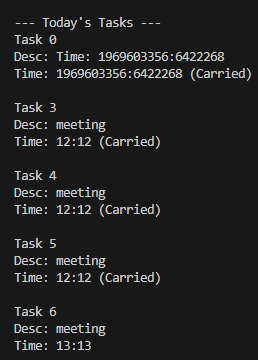
The addTask() method is called to generate a new task to be added to the system. It then finds out whether the limit of task number (MAX\_TASKS) has been exceeded; in that case, it shows a message and leaves the function. In case of room to add more work, it creates a new structure of Task. The following task is assigned a number (one bigger than the total) than the previous one. The program will then ask the user to enter the description of a task making sure that any remaining input is cleared with getchar() and reading the input with fgets(). To eliminate characters in the input containing the newline, it trims it with the use of strcspn(). Then the user will be prompted to key in the time in HH:MM format which in turn is read into a temporary string and split into hour and minute with the help of sscanf(). The task is then left at not completed and not carried over and then put into a list of tasks. The revised list is lastly stored in the file, and a confirmation message is printed.



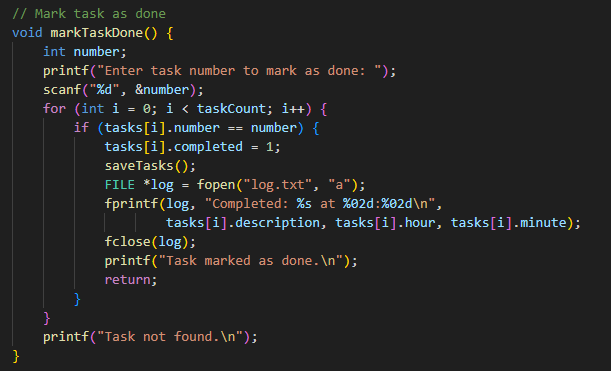
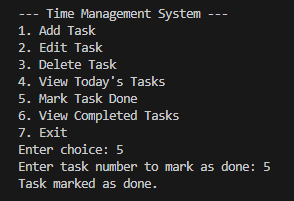
The editTask () script enables the user to edit the description as well as time to be spent on an existing task. It will first prompt the user to request the number of the task he or she wants to edit. Then is will follow through the list of tasks and find the one that will match the specified ID. In the case of successful searching, the program asks a user to input a new description, erasing any remaining input by means of a getchar() call, and then, by means of the fgets() call, reading an updated description. It removes the newline character at the end of an input to make the description clean. Then the user is prompted to enter a new time of the form HH:MM and this time is passed through sscanf() to obtain hour and minute values. After updating the task, the saveTasks() will be used to save updated changes in the file, and the success message will be posted. In the event of the lack of the task number, it will print a message that the task is not present.



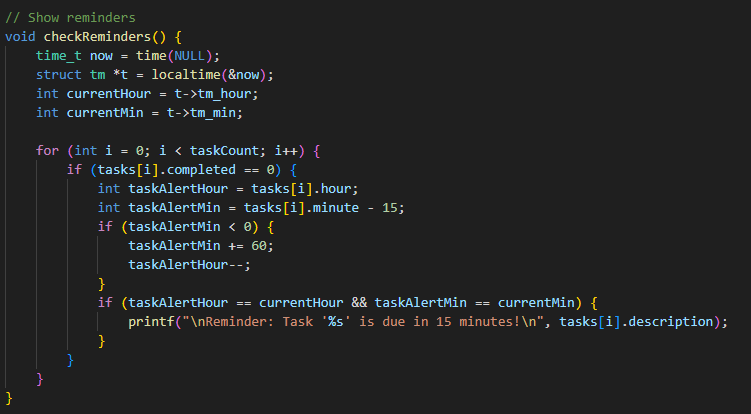
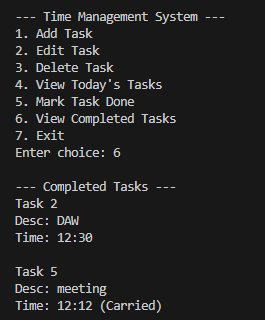
The deleteTask() method is going to be created to delete a definite task out of the list of tasks of a user according to the number of this task. It initiates by asking the user to feed in the number of the task he wants to delete and saves it. It then goes over the list of saved tasks and compares it with a match. After the task requested is located, the function moves all tasks after that location to the right one spot in the array to fill the hole that was left behind by the deleted task. This also involves the updatings of the numbers of the tasks to ensure a proper order. Upon an effective removal, it reduces the overall number of tasks and stores the modified version of a task list using the saveTasks() method, which makes sure that the alterations remain in place and do not disappear. In case, no comparable task is identified, then it advises the user by showing him or her a message saying, task not found. On the whole, this feature allows keeping a clean and current list of tasks due to the ability to remove undesirable or unnecessary elements of this list.



The viewTasks(int completed) method in this Time Management System is created in such a way that it is able to show either the completed or pending (today) tasks depending on the settings of the completed parameter. It starts by printing a header that has dynamic value as per the status of tasks that are being viewed, that is, is it completed or not? This iterative loop is then executed by going through all the tasks and comparing the status of each task as to whether it was completed or not to the requested type that has been put. When it comes to matching tasks, it will print the task number, description, scheduled time (hour and minute), as well as whether the task got over into another day or not. This utility will improve on the productivity of the user as they will find it easy to get a summary of all the works, they have ahead of them or have been completed through the command line.



The viewTasks(int completed) method in this Time Management System is created in such a way that it is able to show either the completed or pending (today) tasks depending on the settings of the completed parameter. It starts by printing a header that has dynamic value as per the status of tasks that are being viewed, that is, is it completed or not? This iterative loop is then executed by going through all the tasks and comparing the status of each task as to whether it was completed or not to the requested type that has been put. When it comes to matching tasks, it will print the task number, description, scheduled time (hour and minute), as well as whether the task got over into another day or not. This utility will improve on the productivity of the user as they will find it easy to get a summary of all the work they have ahead of them or have been completed through the command line.



The checkReminders() function is designed to notify users when a pending task is approaching its scheduled time. It starts by getting the current system time using time() and converting it to a local time structure with localtime(). It then extracts the current hour and minute. The function loops through all tasks in the tasks array and checks for tasks that are not yet completed (completed == 0). For each of these, it calculates a reminder time—15 minutes before the scheduled task time. If subtracting 15 minutes results in a negative value, it adjusts the time correctly by subtracting one hour and adding 60 minutes. When the current time matches this reminder time, the function prints a reminder message alerting the user that the task is due in 15 minutes. This feature helps users stay on track and is directly aligned with your project goal of providing timely command-line reminders for upcoming tasks.

**Conclusion**

The Time Management System project combines practical utility with pedagogical value. By completing this project, learners will deepen their understanding of C programming while delivering a tool that promotes effective daily planning. The project’s modular design also leaves room for future expansions, such as graphical interfaces or networked synchronization.