Project Report

Submitted By:

Muhammad Wakeel 2023-CS-601

Submitted By:

Asif Hussain 2023-CS-646

Submitted To:

Mam Mariyam Manzoor

Dated:

31th December 2025

Department of Computer Science

University of Engineering and Technology Lahore, New Campus

Project Title:

Hospital Resource Scheduler – A Linux-Based OS Project in C

Table of Content:

1. Introduction

Hospitals receive many patients simultaneously, each requiring different services such as consultation, lab tests, or treatment. Due to limited resources (doctors, machines, rooms), inefficient scheduling can cause delays. This project simulates a hospital environment using Operating System concepts in C on Linux, where patient requests are treated as jobs and scheduled using CPU scheduling algorithms.

2. Objectives

- Simulate patient requests as jobs
- Implement and compare CPU scheduling algorithms
- Execute patient jobs using multithreading
- Demonstrate process creation using fork() and exec()
- Implement IPC between scheduler and logger
- Ensure synchronization using semaphores and mutexes
- Use dynamic memory allocation for patient data

3. Implementation Details

3.1 CPU Scheduling Algorithms

The scheduler determines the execution order of patient requests.

Algorithms Implemented: FCFS, SJF, Priority, Round Robin

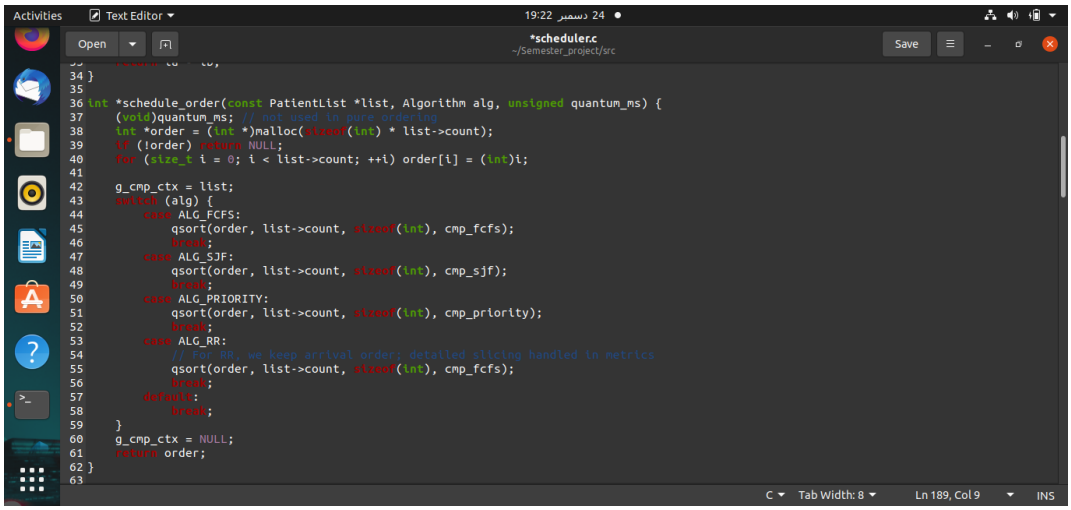

Figure 3.0: CPU scheduling logic using qsort() for multiple algorithms

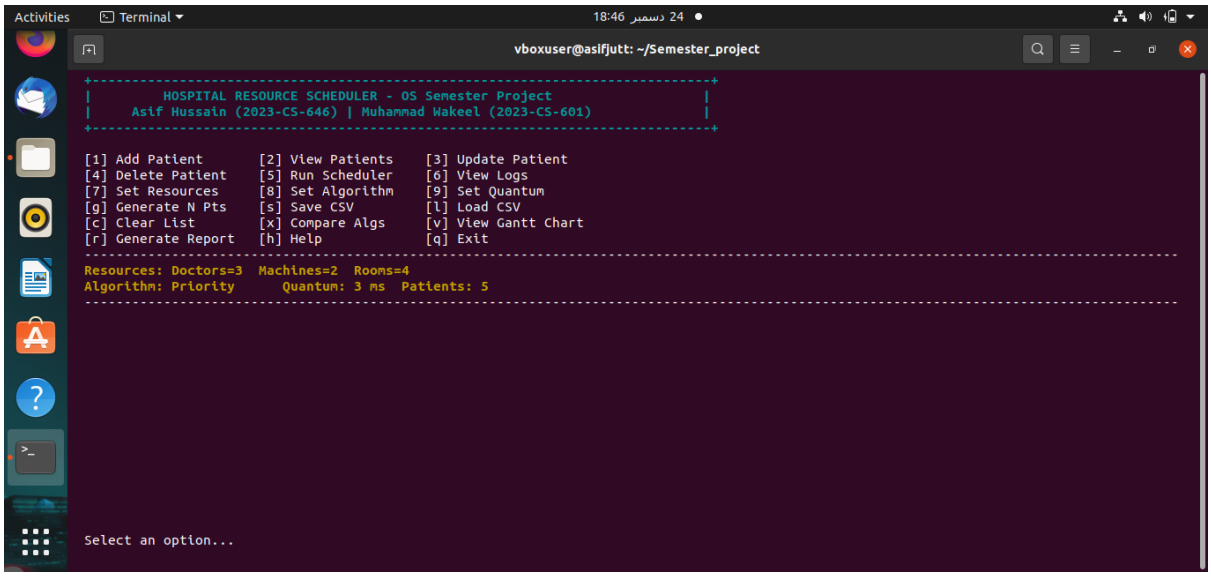

Figure 3.1: Main Manue Selections Interface

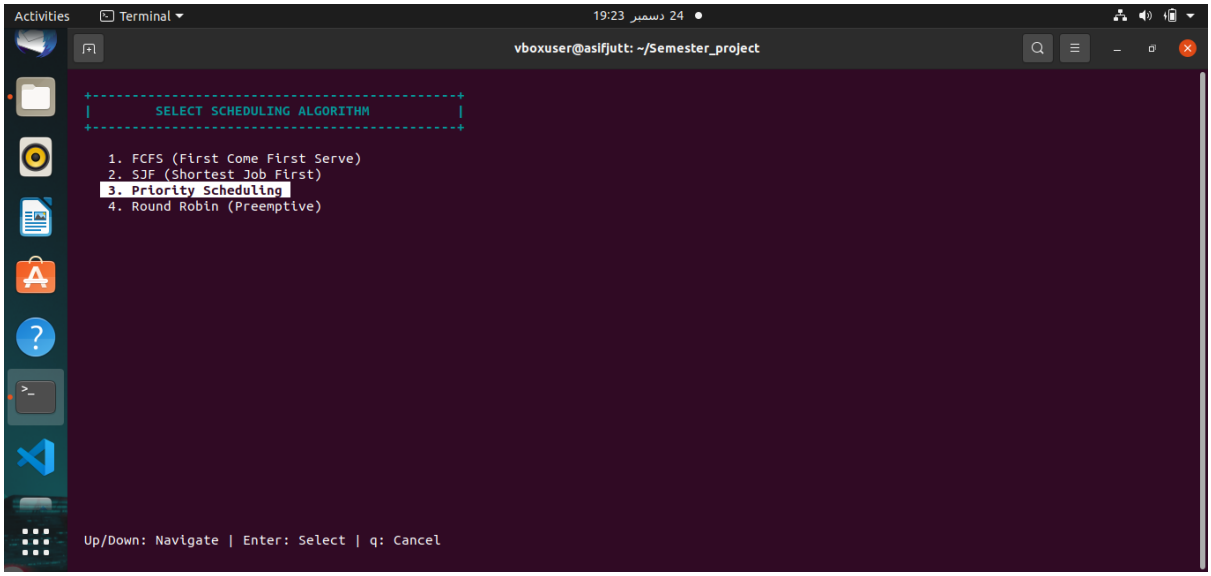

Figure 3.2: Scheduling algorithm selection interface

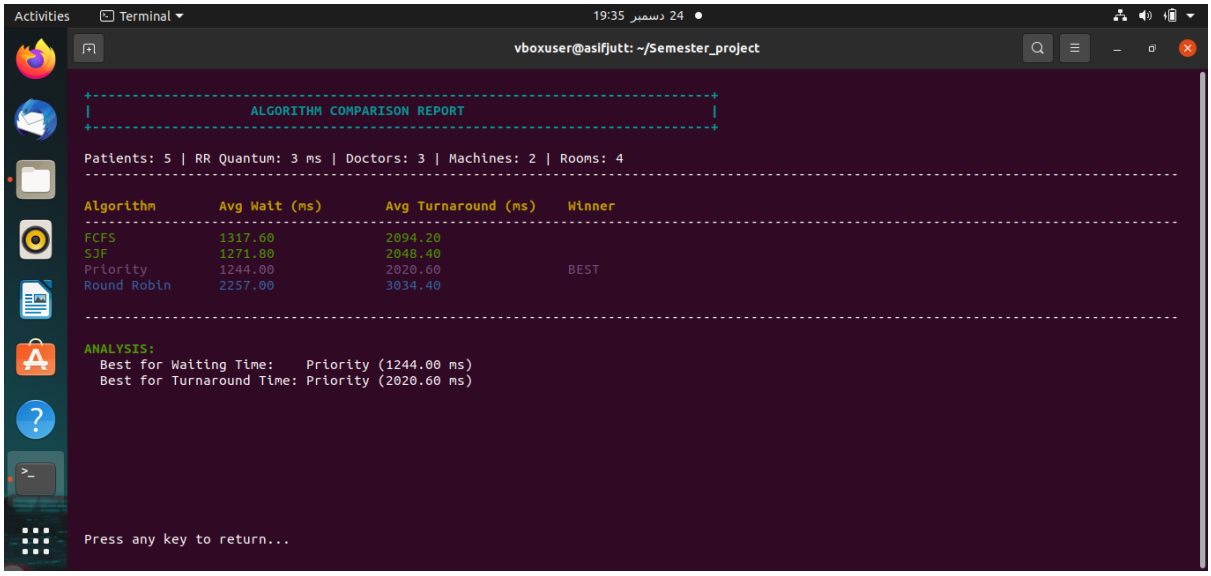

Figure 3.3: Comparison of scheduling algorithms

3.2 Multithreaded Execution

Each patient is handled by a separate thread to simulate parallel hospital services.

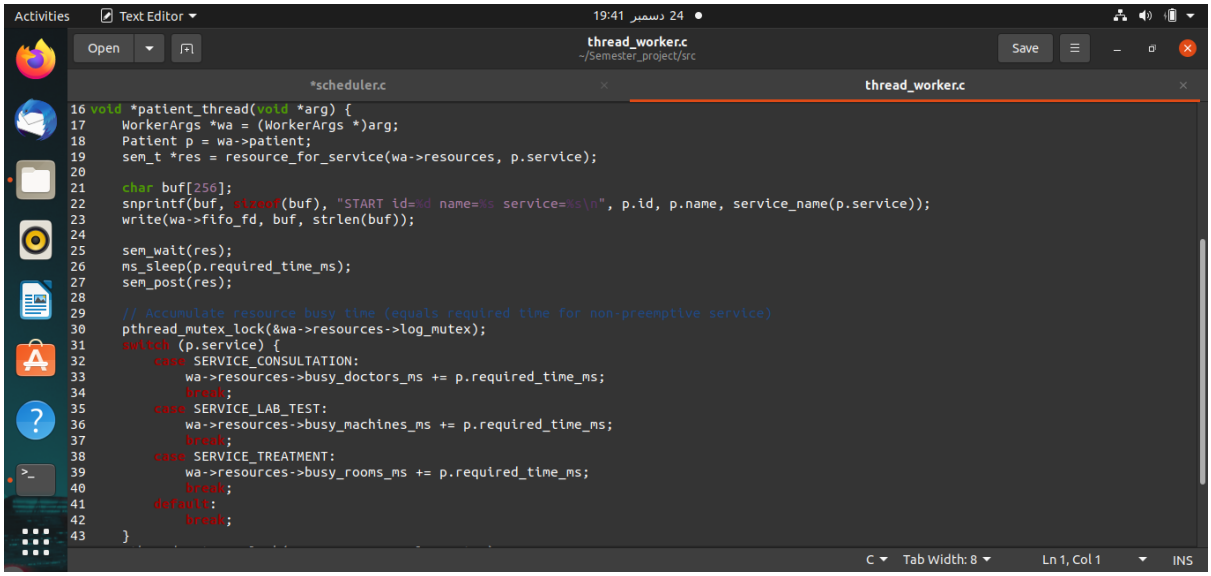

Figure 3.4: Patient handling using POSIX threads

Figure 3.5: Thread creation for patient jobs

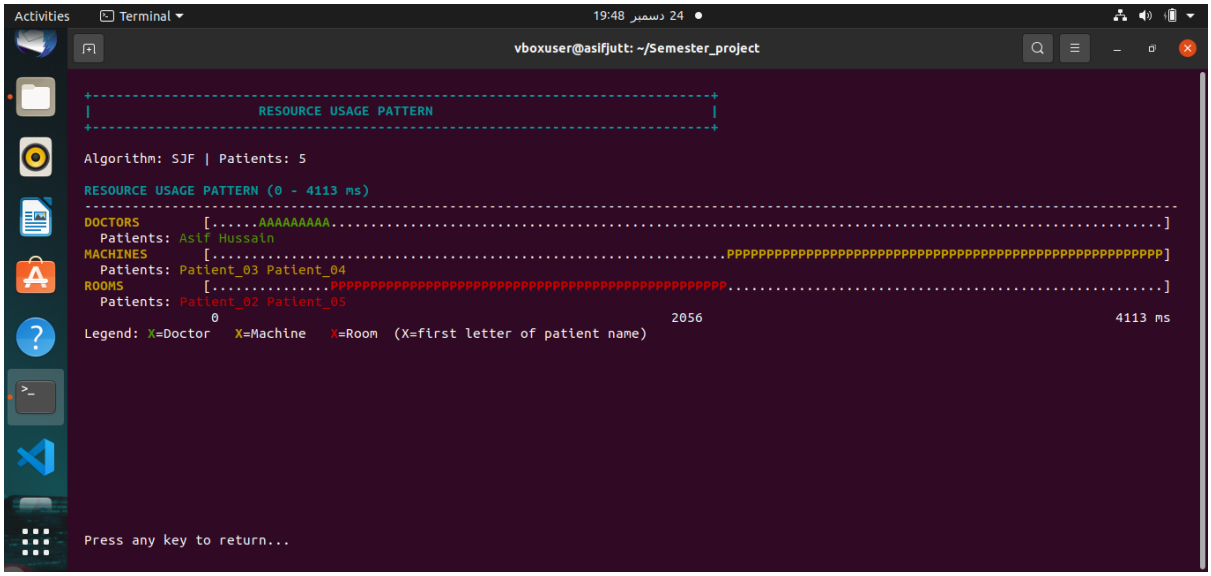

Figure 3.6: Parallel execution of patient threads

3.3 Dynamic Memory Allocation

Patient data is stored dynamically to support runtime changes.

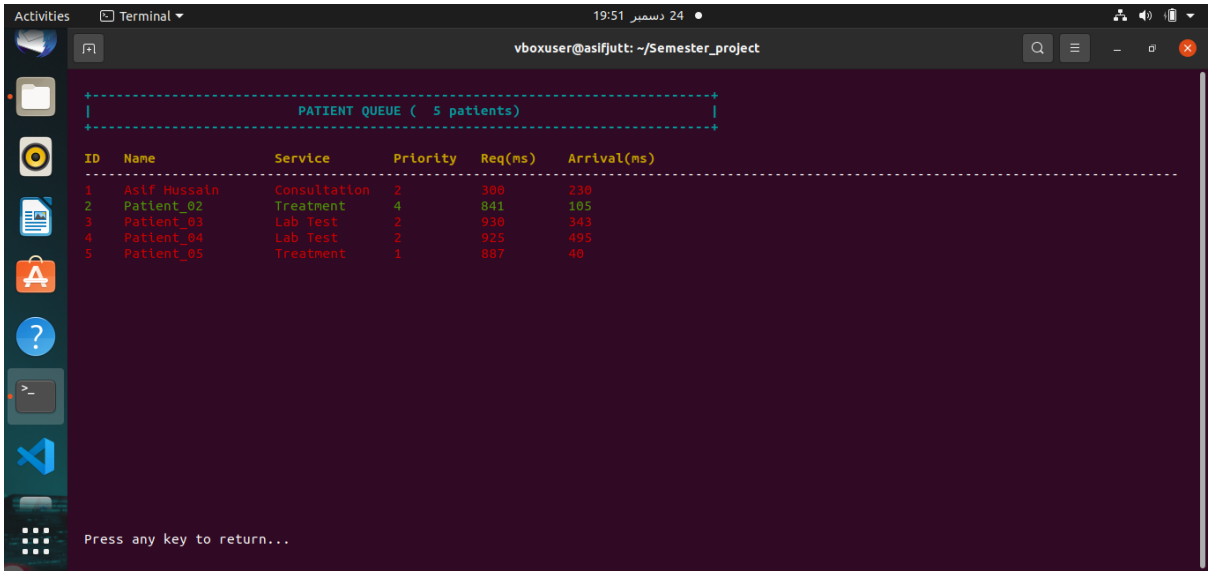

Figure 3.7: Dynamic memory allocation for patient records

3.4 Process Creation

A separate logger process is created to record scheduling events.

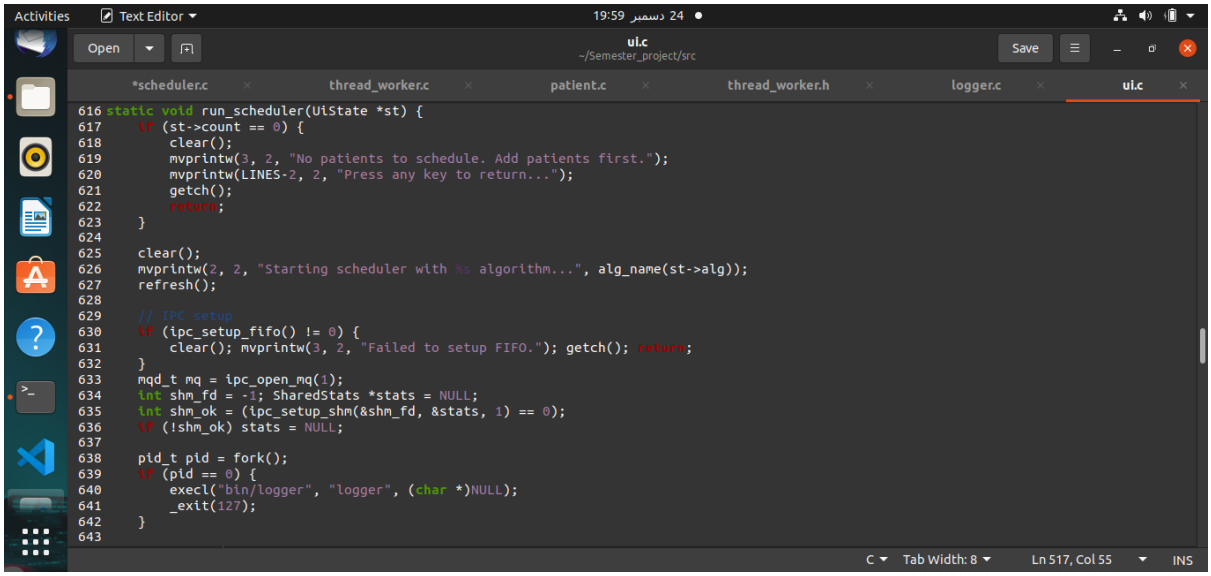

Figure 3.9: Logger process creation using fork() and exec()

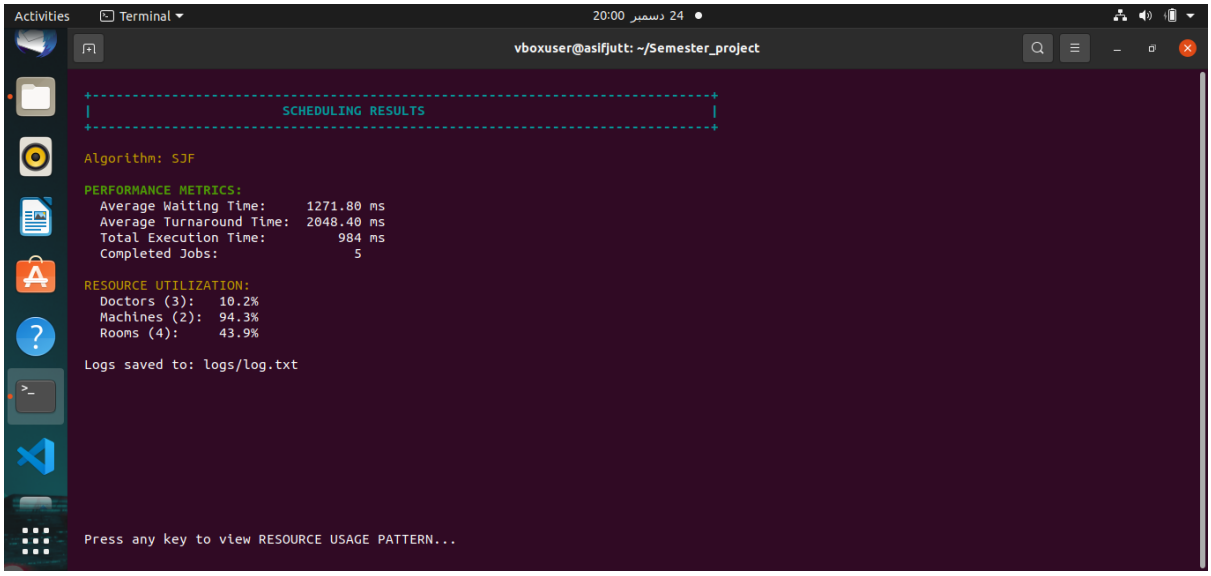

Figure 3.10: Parent (scheduler) and child (logger) processes

3.5 Inter-Process Communication (IPC)

Scheduler and logger communicate using FIFO, message queues, and shared memory.

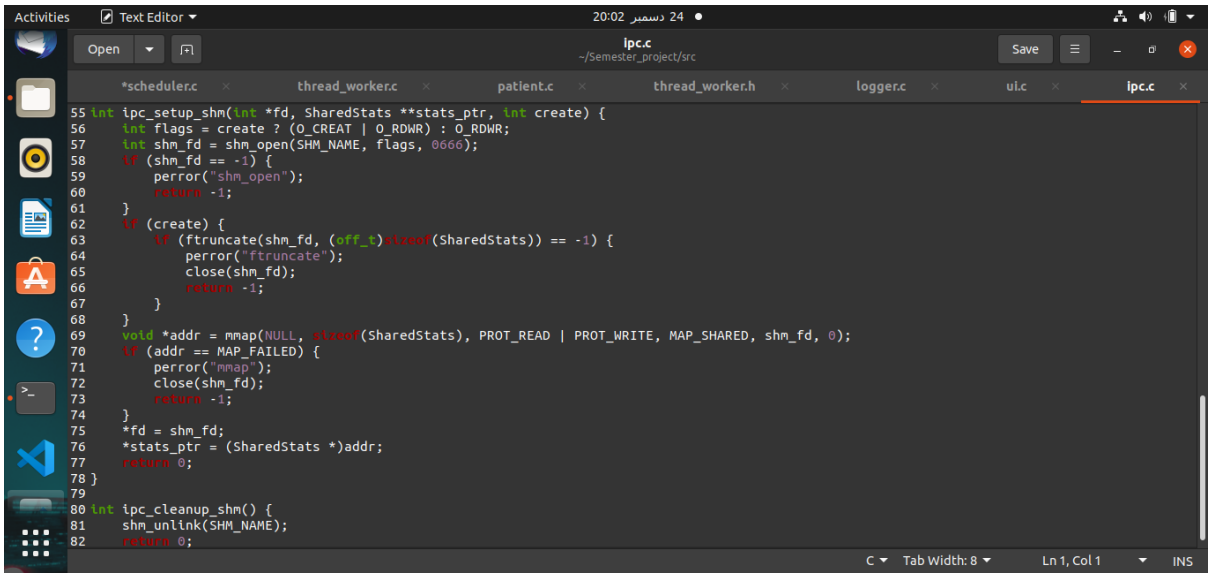

Figure 3.11: IPC implementation using FIFO and shared memory

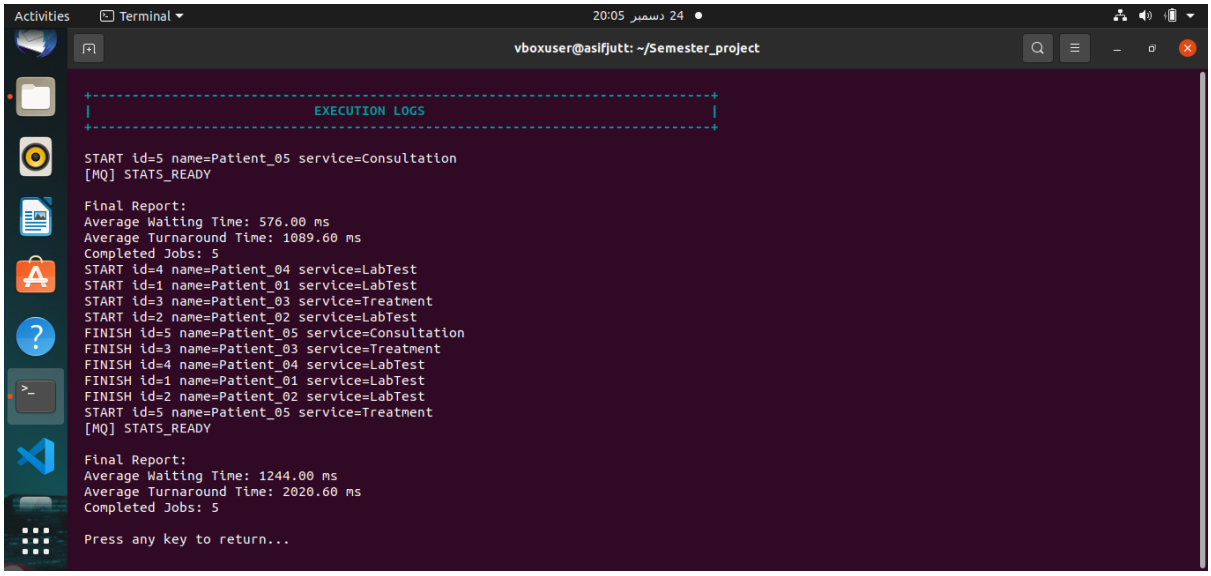

Figure 3.12: Inter-process communication in action

3.6 Synchronization

Semaphores and mutexes control access to limited resources.

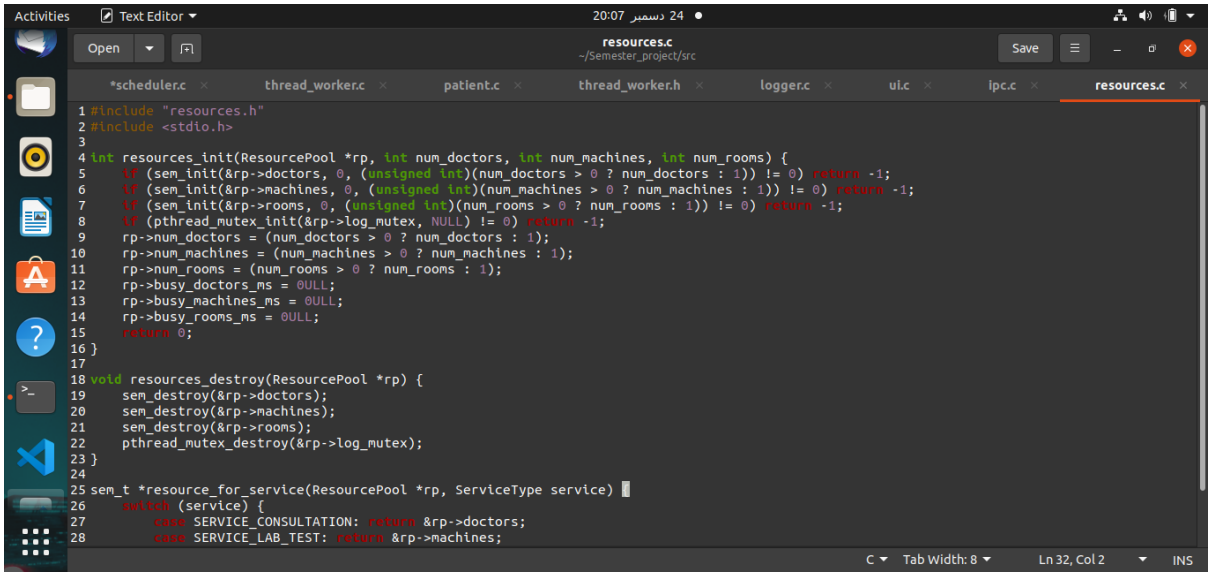

Figure 3.13: Resource initialization using semaphores

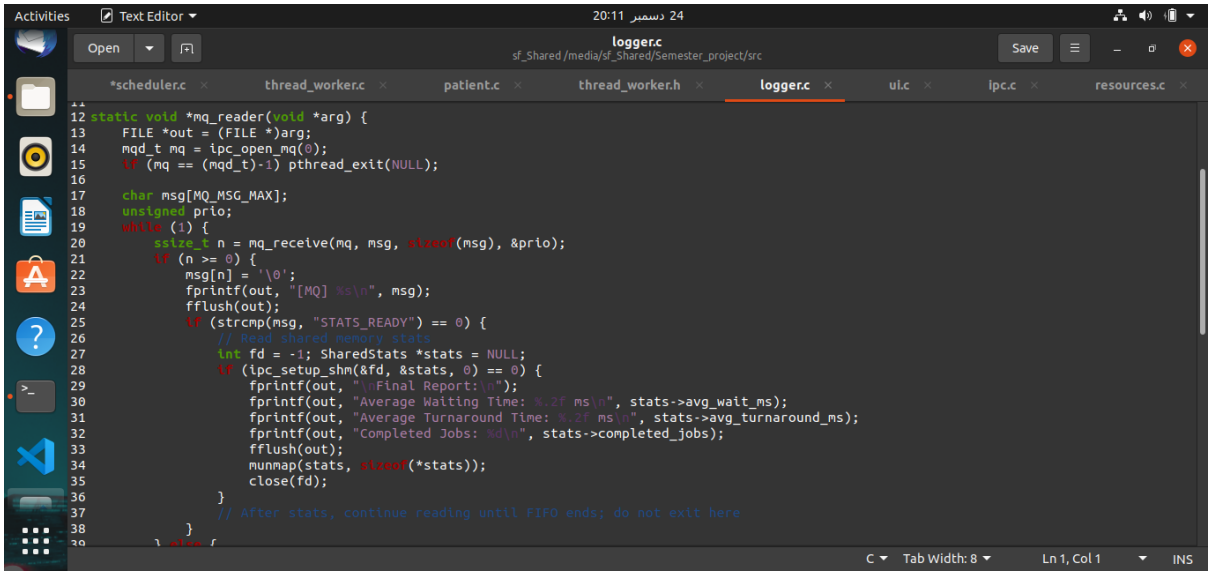

Figure 3.14: Controlled access to shared resources

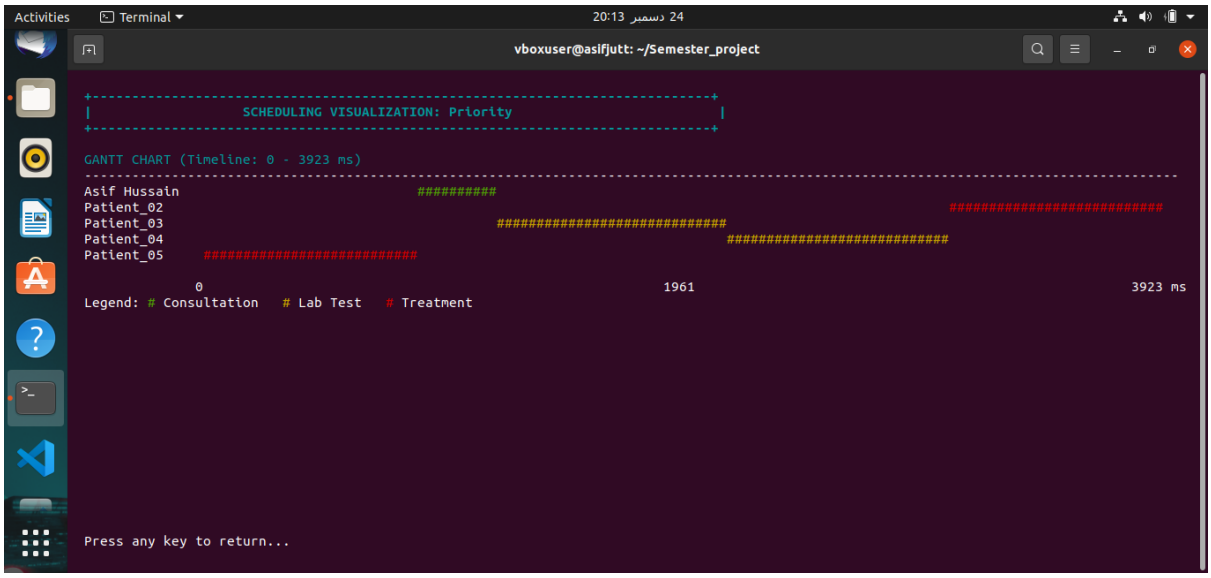

Figure 3.15: Gantt chart visualization of patient execution

5. Conclusion

This project demonstrates the practical application of Operating System concepts such as CPU scheduling, multithreading, IPC, synchronization, and dynamic memory management in a real-world hospital scenario. The system efficiently schedules patient requests while ensuring safe and synchronized resource usage.