Proposal

Semester Project

Submitted By:

Asif Hussain 2023-CS-646

Submitted By:

Muhammad Wakeel 2023-CS-601

Submitted To:

Mam Mariyam Manzoor

Dated:

09th 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

Group Members:

Name	Roll No	Responsibilities
Asif Hussain	2023-CS-646	CPU Scheduling & Threads
Muhammad Wakeel	2023-CS-601	Process Creation, IPC & Synchronization

Introduction:

Hospitals receive many patients at the same time, and every patient needs different services. Some need to meet a doctor, some need lab tests, and some need treatment. But a hospital has limited resources like doctors, nurses, and medical machines. If resources are not used properly, patients may face delays.

This project is a Hospital Resource Scheduler, designed using Operating System concepts. In our system:

- Every patient request behaves like a job/process.
- It requires CPU time, a resource, and execution time.
- The scheduler decides which patient request should be served first.
- We will simulate a real hospital environment using C on Linux.

The project demonstrates how OS scheduling and process management can make better use of limited hospital resources. It uses threads, inter-process communication, synchronization, and CPU scheduling algorithms to handle multiple patient requests.

Objectives:

The main objectives of the project are:

1. To simulate patient requests as jobs running on a scheduler.
2. To implement multiple CPU scheduling algorithms and compare their results.
3. To use multithreading to run patient tasks in parallel.
4. To show process creation using fork() and exec().

5. To implement Inter-Process Communication (IPC) between scheduler and logger.
6. To use semaphores and mutexes to manage shared hospital resources safely.
7. To demonstrate dynamic memory allocation for patient data.

Scope of the Project:

The project will cover the following working components:

1. CPU Scheduling Algorithms

The scheduler will support multiple scheduling techniques such as:

- First Come First Serve (FCFS)
- Shortest Job First (SJF)
- Priority Scheduling
- Round Robin (RR)

We will compare how each algorithm affects waiting time and turnaround time of patients.

2. Multithreaded Execution

Each patient request will be handled by a separate thread, representing:

- Consulting a doctor
- Running a lab test
- Getting treatment

Threads will run in parallel to simulate multiple real-time requests.

3. Dynamic Memory Allocation

Patient information (ID, name, priority, service type, required time, etc.) will be stored using:

- malloc()
- calloc()
- free()

Data structures will be created at runtime based on number of patients.

4. Process Creation

We will create a separate process for log handling using:

- fork(): Creates child process

- exec(): Executes logging program

This child process will record:

- Scheduling steps
- Resource usage
- Waiting times
- Completed jobs

5. Inter-Process Communication (IPC)

Scheduler and logger will communicate using:

- Shared Memory
- Message Queues
- Pipes

The scheduler sends log messages to the logger, and the logger stores them in a text file.

6. Synchronization

Multiple patient requests will compete for resources like:

- Doctors
- Machines
- Rooms

To avoid conflicts, we will use:

- Mutex Locks → to protect critical sections
- Semaphores → to control limited resources

This ensures that two threads cannot use the same doctor at the same time.

7. Extendable Features

The project is designed in a modular way, so in future:

- A GUI can show real-time status of patient queues
- Statistical analysis can show average time, resource usage
- More resource types can be added

Tools and Technologies:

- Programming Language: C
- Operating System: Linux (Ubuntu recommended)
- Compiler: gcc
- Threading Library: pthreads
- Synchronization: POSIX Semaphores, Mutex
- IPC: shared memory/message queues/pipes
- Process Management: fork(), exec()

Expected Output:

After running the project:

- The system will generate a list of patient requests.
- CPU scheduling algorithm will decide the order of execution.
- Each patient task will run on threads.
- Real-time log will be maintained by the logger process.
- Final report will show:
  - waiting time
  - turnaround time
  - resource usage
  - comparison between scheduling algorithms

Conclusion:

This project shows how Operating System concepts can help in managing real-life problems like hospital resource allocation. It covers CPU scheduling, processes, threads, IPC, dynamic memory, and synchronization in a single integrated system. The idea can be extended further to a full hospital automation system.