Client-Server Based Course Registration System
(Academia Portal)

Problem Statement

To build a client-server based academic course registration portal, where users can act as Admin, Faculty, or Student. The server handles authentication, user management, course management, and enrollment operations, while clients interact over a network to perform various tasks based on their roles.

Implementation Details

Sockets Usage

Sockets are used to establish connections between the client and server. The server initializes a socket, binds to port `8080`, listens, and accepts multiple client connections. Each client is handled in a separate thread. Clients communicate by sending and receiving messages over the socket.

Locks (`pthread_mutex_t`) Usage

Three mutexes are used to ensure data consistency during concurrent access:

- **Student Mutex**: Ensures mutual exclusion while accessing or modifying `students.dat`.
- **Faculty Mutex**: Protects `faculty.dat` during read/write operations.
- **Course Mutex**: Synchronizes access during enrollment, unenrollment, or updating course data.

Semaphores Usage

Semaphores are included through `<semaphore.h>` but are not actively used in the current implementation. They may be reserved for future use (e.g., access control or managing concurrency limits).

Threading

Each client is handled in a separate thread using `pthread`. Threads are detached so that they clean up automatically upon termination, enabling efficient handling of multiple concurrent clients.

File Operations

Data is stored and manipulated using:

- `admin.dat` — Admin credentials.
- `students.dat` — Student information.
- `faculty.dat` — Faculty and offered course info.

Files are accessed using system calls such as `open`, `read`, `write`, and `lseek`. Mutexes simulate file-level locking for consistency.

Signal Handling

A signal handler is set up to clean up mutex resources on server shutdown (`SIGINT`), ensuring graceful termination and preventing resource leakage.

User Roles

Admin

- Add New Student
- Add Faculty
- Activate Student
- Deactivate Student
- Update Student / Faculty Details
- Exit

Student

- Enroll in New Course
- Unenroll from Courses
- View Enrolled Courses
- Change Password
- Exit

Faculty

- Add New Courses
- Remove Offered Courses
- View Enrollments
- Change Password
- Exit

Structures Used

Three main structures are defined in the system:

1. Student

- ID
- Username
- Password
- Active Status
- List of Enrolled Courses
- Course Count

2. Faculty

- ID
- Username
- Password
- List of Offered Courses
- Seats per Course
- Course Count

3. Admin

- Username
- Password

Server Code Functionality

The server sets up a socket, binds it to a port, and listens for incoming client connections. For each connection, a new thread is created to handle client interaction independently.

Clients go through a login interface to select their role (Admin, Student, Faculty), authenticate using credentials, and access role-specific features. The server handles all logic related to file access, validation, updates, and concurrency using mutexes.

Client Code Functionality

The client connects to the server and displays role-based menus through a text interface. It accepts user input, sends requests to the server, and displays responses such as success messages, errors, and data.

Output Screenshots

1. Server terminal showing connections.

```
jvs_0509@Jitin:/mnt/c/Users/jitin/OneDrive/Desktop/Mini Project$ gcc -o server server.c
jvs_0509@Jitin:/mnt/c/Users/jitin/OneDrive/Desktop/Mini Project$ ./server
Admin file created and initialized
Server started on port 8080
New client connected
user id: 0
New client connected
user id: 0
New client connected
user id: 0
New client connected
user id: 1
New client connected
user id: 0
```

Figure 1: 1

2. Admin menu interface.

```
jvs_0509@Jitin:/mnt/c/Users/jitin/OneDrive/Desktop/Mini Project$ ./client
Connected to Academia Portal Server

Welcome to Academia Portal
1. Admin
2. Faculty
3. Student
Enter your choice: 1
Enter username: admin
Enter password: admin123
Login successful

===== ADMIN MENU =====
1. Add Student
2. Add Faculty
3. Activate/Deactivate Student
4. Update Student/Faculty details
5. Exit
Enter your choice: 1
Enter student username: jitin
Enter student password: jitin
Student added successfully with ID: 0

===== ADMIN MENU =====
1. Add Student
2. Add Faculty
3. Activate/Deactivate Student
4. Update Student/Faculty details
5. Exit
Enter your choice: 2
Enter faculty username: badri
Enter faculty password: badri
Faculty added successfully with ID: 0

===== ADMIN MENU =====
1. Add Student
2. Add Faculty
3. Activate/Deactivate Student
4. Update Student/Faculty details
5. Exit
Enter your choice: 1
Enter student username: taral
Enter student password: taral
Student added successfully with ID: 1

===== ADMIN MENU =====
1. Add Student
2. Add Faculty
3. Activate/Deactivate Student
4. Update Student/Faculty details
5. Exit
Enter your choice: 2
Enter faculty username: Thangaraju
Enter faculty password: thangaraju
Faculty added successfully with ID: 1
```

Figure 2:

3. Faculty adding a course.

```
jvs_0509@Jitin:/mnt/c/Users/jitin/OneDrive/Desktop/Mini Project$ ./client
Connected to Academia Portal Server

Welcome to Academia Portal
1. Admin
2. Faculty
3. Student
Enter your choice: 2
Enter username: badri
Enter password: badri
Login successful

===== FACULTY MENU =====
1. Add new Course
2. Remove offered Course
3. View enrollments in Courses
4. Password Change
5. Exit
Enter your choice: 1
Enter course name: OS
Enter number of seats: 50
Course added successfully

===== FACULTY MENU =====
1. Add new Course
2. Remove offered Course
3. View enrollments in Courses
4. Password Change
5. Exit
Enter your choice: 1
Enter course name: C++
Enter number of seats: 50
Course added successfully
```

Figure 3:

4. Student Enrolling in a Course.

```
jvs_0509@Jitin:/mnt/c/Users/jitin/OneDrive/Desktop/Mini Project$ ./client
Connected to Academia Portal Server

Welcome to Academia Portal
1. Admin
2. Faculty
3. Student
Enter your choice: 3
Enter username: jitin
Enter password: jitin
Login successful

===== STUDENT MENU =====
1. Enroll to new Courses
2. Unenroll from already enrolled Courses
3. View enrolled Courses
4. Password Change
5. Exit
Enter your choice: 1
Available Courses:
- OS (Available seats: 50)
- C++ (Available seats: 50)
Enter course name to enroll: OS
Successfully enrolled in course

===== STUDENT MENU =====
1. Enroll to new Courses
2. Unenroll from already enrolled Courses
3. View enrolled Courses
4. Password Change
5. Exit
Enter your choice: 1
Available Courses:
- OS (Available seats: 49)
- C++ (Available seats: 50)
Enter course name to enroll: C++
Successfully enrolled in course
```

Figure 4:

5. Student viewing enrolled courses.

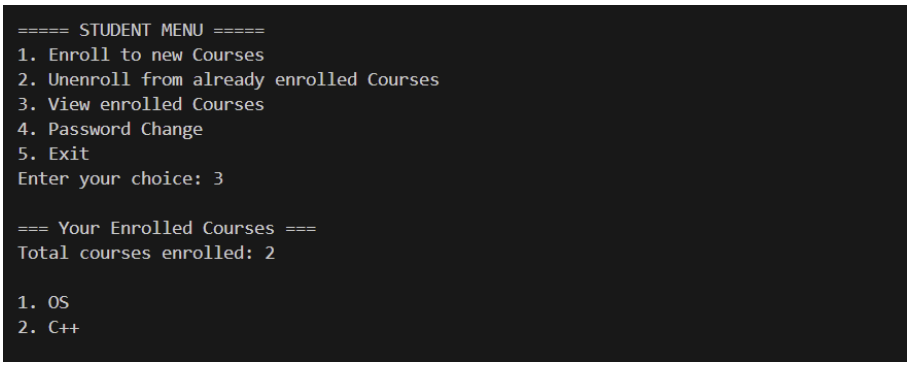

Figure 5:

6. Faculty viewing course enrollments.

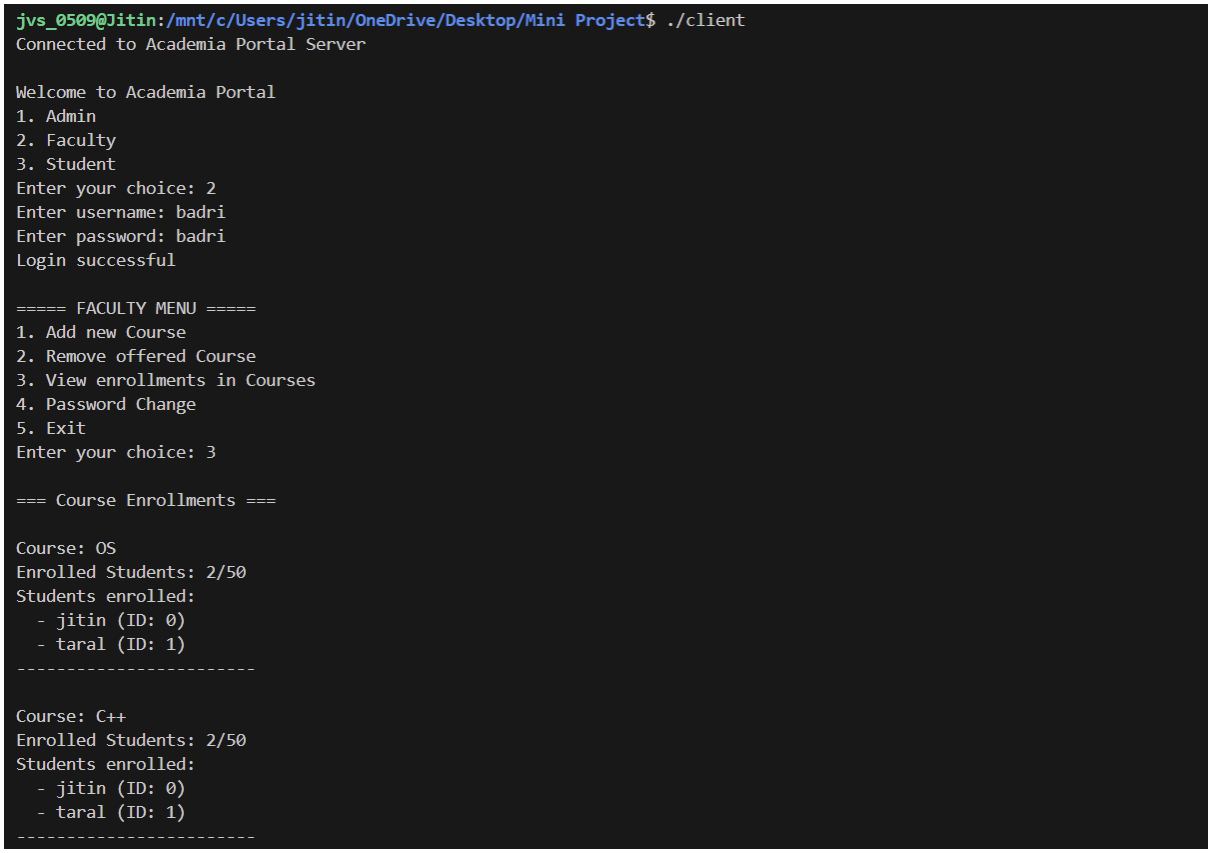

Figure 6: