Project Description: Multithreaded Client-Server Application

You will design and implement a multithreaded client-server application using message passing between processes. The server will accept connections from multiple clients and maintain a list of connected clients. Each client can send requests to the server, which will process them and send back responses.

Basic Requirements

The server has the following tasks:

1. Initialize a message queue for accepting connections from clients.
2. Listen for connections from clients and add them to a list of connected clients.
3. For each connected client, spawn a new thread that listens for messages from that client and processes them.
4. Each message from a client should contain a request type and data payload. The server should process each request and generate a response, which should be sent back to the client using its message queue.
5. When a client disconnects, its message queue should be removed from the list of connected clients and the corresponding thread should be terminated.

The client has the following tasks:

1. Initialize a message queue for communicating with the server.
2. Connect to the server by sending a connection request to its message queue.
3. Send requests to the server by sending messages to its message queue.
4. Wait for responses from the server by listening to its message queue.
5. Gracefully disconnect from the server by sending a disconnect request to its message queue.

Bonus Requirements

1. The server should enforce proper access control to prevent clients from accessing messages they should not have access to.
2. The server should use shared memory and semaphores to coordinate access to shared resources.

Tips

1. Use a message queue to communicate between the server and clients.
2. Use a data structure to maintain a list of connected clients and their corresponding message queues.
3. Use pthreads to create a new thread for each connected client.
4. Use a loop to continuously listen for messages from clients.
5. Use a switch statement to process different types of requests.
6. Use the pthread_mutex_lock and pthread_mutex_unlock functions to protect shared resources, such as the list of connected clients.

7. Handle errors and invalid input gracefully by printing error messages and returning appropriate error codes.

Deliverables

1. The server program source code.
2. A README file that includes instructions for building and running the program, as well as a brief overview of the implementation.
3. Any necessary client program source code.
4. Any configuration files necessary to run the program.
5. Documented word file with the description of all your work (the document should be formated and in the proper form i.e table of content, list of figures, references, etc.)