TW-Mailer Pro Version: Project Description

Client and Server Architecture

The TW-Mailer Pro Version follows a client-server architecture where the client sends commands to the server, and the server processes these commands and responds accordingly. The server is designed to handle multiple clients concurrently using threads. This ensures that the server can manage multiple client connections simultaneously without blocking.

Client

- The client application connects to the server and sends commands such as LOGIN, SEND, LIST, READ, and DEL.
- The client waits for the server's response after each command and processes the response accordingly.

Server

- The server listens for incoming client connections on a specified port.
- Upon receiving a connection, the server creates a new thread or process to handle the client's requests.
- The server processes commands from the client, interacts with the LDAP server for authentication, and manages message storage and retrieval.

Used Technologies and Libraries

- **Programming Languages**: C++ and C using UNIX Sockets
- **Libraries**:
  - `nlohmann/json` for JSON parsing and serialization
  - `std::filesystem` for file and directory operations
  - `UUIDv4` for generating unique identifiers for emails
  - OpenLDAP C-API (`libldap2-dev`) for LDAP authentication
- **Concurrency**: C++ Threads

Development Strategy and Needed Protocol Adaptions

Development Strategy

1. **Modular Design**: The project is divided into multiple classes and files to ensure modularity and separation of concerns. For example, `mailHandler` is responsible for handling email-related operations.
2. **UUID Generation**: Unique identifiers for emails are generated using the `UUIDv4` library to ensure each email has a unique filename.
3. **JSON Serialization**: Emails are stored in JSON format to facilitate easy serialization and deserialization.
4. **Filesystem Operations**: The project uses `std::filesystem` to manage directories and files.

Needed Protocol Adaptions

1. **Concurrency**:
    - Refactor the server to handle multiple clients concurrently using threads.
    - Identify critical sections and use synchronization mechanisms to prevent race conditions.
2. **Authentication**:
    - Implement the LOGIN command to authenticate users using the FH LDAP server.
    - Allow only authenticated users to execute commands other than QUIT.
    - Limit login attempts to 3 per user and IP, and blacklist the IP for 1 minute after 3 failed attempts.
3. **Command Adaptions**:
    - Remove the ability to manually set the sender in the SEND command. Use session information instead.
    - Automatically set the username in LIST, READ, and DEL commands based on session information.
4. **Email Storage**:
    - Implement email storage as JSON files in the mail directory.
    - Use UUIDv4 to generate unique filenames for each email.
5. **Client Functions**:
    - Add functions to receive and display emails on the client side.
    - Ensure the client can handle large messages by receiving data in chunks.

Used Synchronization Methods

- **Mutexes**: Mutex Lock Guards are used to protect critical sections of code from concurrent access. This ensures that shared resources are accessed in a thread-safe manner, preventing race conditions and data corruption.

Handling of Large Messages

- **Message Storage**: Messages are stored as JSON in the mail directory, with each message in a separate file. This allows for efficient serialization and deserialization of message data.
- **Chunked Data Transfer**: The client and server handle large messages by transferring data in chunks. This ensures that the system can manage large messages without running into memory limitations or performance issues.

Conclusion

The TW-Mailer Pro Version enhances the basic TW-Mailer project by introducing concurrency, authentication, and improved command handling. By using threads, the server can handle multiple clients simultaneously. LDAP authentication ensures secure access, and JSON-based email storage provides a flexible and efficient way to manage email data.