Client-Server Architecture

This project implements a client-server architecture using the socket programming paradigm. The communication follows a request-response model:

Client: The client is responsible for sending commands to the server and processing the responses. It connects to the server using a TCP socket.

Server: The server listens for client connections on a specified port and processes commands such as SEND, LIST, READ, DEL, and QUIT. It interacts with a mail spool directory for storing and retrieving message data.

Used Technologies

Programming Language: C++, Makefile

Development Strategy

Design:

- Defined a simple protocol with commands like SEND, LIST, READ, DEL, and QUIT.
- Established validation rules for input fields (e.g., username and subject).
- Decided to use a directory-based storage system for messages, with each message stored in a separate file.

Implementation:

- Client:
  - Implemented functions for validating user inputs and handling commands.
  - Commands are converted to structured requests and sent to the server.
- Server:
  - Implemented handlers for each command.
  - Used file I/O to store and retrieve messages.
  - Managed user-specific directories within the mail spool.

Testing:

- Verified end-to-end communication for all commands.
- Tested edge cases for input validation and error handling (e.g., invalid usernames, non-existent message files).

Required Adaptations:

- Scalability:
  - Current implementation handles one client per process. To support multiple concurrent clients, threading or asynchronous I/O could be introduced.
- Error Handling:
  - Enhance error responses with more descriptive messages (e.g., differentiating between missing users and unreadable files).