Hameed Sura & Gruber Aline

TwMailer Protocol

Client and Server Architecture:

Client:
The client is the end-user application responsible for creating and sending Messages and connects to the server using the server's IP address and port. It interacts with the server through a series of specific commands.

Server:
The server is the core of our email system. It listens for incoming connections, processes client requests, and manages storage.
It listens on a designated port, waiting for clients to establish connections.

How they connect:
The server allocates dedicated threads to handle multiple clients simultaneously. Communication between clients and the server relies on a text-based protocol that defines specific commands for sending, listing, reading, and deleting emails. The server maintains a "mail spool" directory, storing emails in separate files for different users.

Used Technologies:

C++ Programming Language:
Both the client and server are developed using C++

Sockets and Network Communication:
The system relies on low-level socket programming for network communication. Sockets allow the client and server to establish connections and exchange data over a network.

TCP/IP Protocol:
Our system uses TCP/IP to ensure that email data is sent and received without loss.

File Handling:
File handling is important for our email system, as it handels creating, reading, writing, and deleting email files. The system uses standard file I/O operations for these tasks.

Multi-Threading:
Multi-threading is employed on the server to handle multiple client connections simultaneously. Each client is assigned a dedicated thread.

Text-Based Protocol:
Communication between the client and server is carried out using a simple text-based protocol. Specific commands and responses are exchanged in plain text.

Directory Management:
The server maintains a "mail spool" directory, where email messages are stored as individual files. Basic directory operations are used for creating and managing these directories.

Signal Handling:
Signal handling is used for shutting down the server when needed.
Bsp: the system responds to a SIGINT signal (e.g., when the user presses Ctrl+C) to exit safely.

Standard Input/Output Streams:
The standard input and output streams (stdin and stdout) are used for reading user input and providing output messages to the client.

File Stream (fstream):
This class enables reading and writing operations into files.

Deployment Strategies:

We started with the client side and then moved to the server side. First up was 'send,' which we got working smoothly. Next, we handled 'list' to display messages and 'read' to view specific ones. We repurposed some 'list' code for file searching in the 'read' function.
Afterwards we moved to, 'delete'. We borrowed heavily from our 'read' code to locate and eliminate messages. Finally, we wrapped it all up with 'quit,' the simplest part of the process.

Once everything was set, we ran tests and added some error messages.