TWMailer – Protokoll

1. Anwendung des Programmes

Zuerst startet man im Terminal mit „./twmailer-server 6543 maildir“ den Server und dann in einem ANDERN Terminal mit „./twmailer-client 127.0.0.1 6543“ den Client. Dann kann man die einzelnen Befehle im Clientterminal ausführen.

Mit SEND <SENDER> <RECIVER> <SUBJECT> <MESSAGE> . wird im „maildir“ ein Ordner erstellt (falls noch nicht vorhanden) mit dem namen des <RECIVER> und ein Textfile mit dem Namen message_ID (die ID wird mittels Zeitstempel generiert).

Mit LIST <USER> werden alle <Subjects> von diesem User ausgeben und ein Counter wieviele es sind.

Mit READ <USER> <MESSAGENUMBER> kann man die komplette Nachricht auslesen, wobei z.B. Messagenumber 1 die oberste im Ordner ist.

DELETE funktioniert genauso wie READ nur löscht es die Nachricht und lest sie nicht aus.

Und QUIT loggt quasi den Client aus.

2. Architektur von Client und Server

Der TWMailer basiert auf einer klassischen Client-Server-Architektur, bei der der Client Anfragen stellt und der Server diese verarbeitet. Der Client nimmt Benutzereingaben entgegen und sendet Befehle wie das Versenden, Abrufen oder Löschen von Nachrichten an den Server. Der Server verwaltet die Nachrichten der Benutzer, verarbeitet die empfangenen Befehle und sendet die entsprechenden Antworten zurück an den Client.

Die Kommunikation erfolgt über TCP-Sockets, die eine zuverlässige Verbindung zwischen Client und Server gewährleisten. Das Textprotokoll, das für die Kommunikation verwendet wird, basiert auf klar strukturierten Befehlen wie LIST, READ, DEL und SEND, die der Server ausführt.

3. Verwendete Technologien

- **Programmiersprache:** Die Implementierung wurde in C vorgenommen, was uns direkte Kontrolle über die Netzwerkkommunikation und den Speicher gibt.
- **Vorbereitete Dateien:** Zu Beginn des Projekts hatten wir bereits eine grundlegende client.c und server.c Datei zur Verfügung. Diese Dateien enthielten die grundlegende Architektur für die Verbindung über Sockets. Wir haben diese Dateien angepasst und um Funktionen wie das Abrufen und Löschen von Nachrichten erweitert.
- **TCP-Sockets:** Die Kommunikation zwischen Client und Server erfolgt über TCP-Sockets. Jeder Client baut eine Verbindung zum Server auf, um Anfragen zu senden und Antworten zu erhalten.
- **Inkludierte Bibliotheken:** In beiden Programmen werden Standard-C-Bibliotheken verwendet, um grundlegende Funktionen wie Ein-/Ausgabe, Speicherverwaltung und Netzwerkkommunikation bereitzustellen. Zu den wichtigen Includes gehören:

- <stdio.h> für Ein- und Ausgabe
- <stdlib.h> für Speicherverwaltung
- <string.h> für Zeichenkettenmanipulation
- <unistd.h> für Systemaufrufe wie read und write
- <sys/socket.h> und <arpa/inet.h> für Socket-Programmierung
- <ctype.h> für Funktionen wie isdigit(), um Zeichen zu überprüfen

4. Entwicklungsstrategie und notwendige Anpassungen

Die Entwicklung folgte einem inkrementellen Ansatz, wobei wir auf den gegebenen client.c und server.c Dateien aufbauten:

1. **Initiale Struktur:** Die bestehende Socket-Verbindung zwischen Client und Server war bereits vorhanden. Wir haben die grundlegende Struktur verfeinert, um die Funktionalität des Mailers zu erweitern.
2. **Anpassung der Funktionen:** Basierend auf den gegebenen Grundfunktionen haben wir neue Funktionalitäten wie das Abrufen (LIST), Lesen (READ) und Löschen (DEL) von Nachrichten implementiert. Dabei wurde das Protokoll so erweitert, dass zusätzlich zu den Befehlen auch Benutzernamen und Nachrichtennummern übermittelt werden.
3. **Fehlerbehandlung**: Ein wesentlicher Aspekt der Anpassung war die Verbesserung der Fehlerbehandlung. So haben wir sichergestellt, dass ungültige Eingaben oder Verbindungsabbrüche korrekt verarbeitet werden, ohne das System zu beeinträchtigen.

5. Während der Entwicklung mussten wir einige Anpassungen vornehmen:

- **Puffergrößen:** Es war wichtig, die Puffer für das Empfangen und Senden von Daten korrekt zu dimensionieren, um sicherzustellen, dass sowohl kurze als auch längere Nachrichten korrekt verarbeitet werden.
- **Erweiterung des Protokolls:** Einige Befehle, wie das Lesen einer spezifischen Nachricht (READ), erforderten zusätzliche Parameter wie die Nachrichtennummer. Dies musste bei der Anpassung der Kommunikation zwischen Client und Server berücksichtigt werden.
- **Robustheit:** Um die Robustheit des Systems zu erhöhen, wurden verschiedene Mechanismen zur Fehlerbehandlung und zur Stabilisierung bei Verbindungsabbrüchen oder falschen Eingaben integriert.