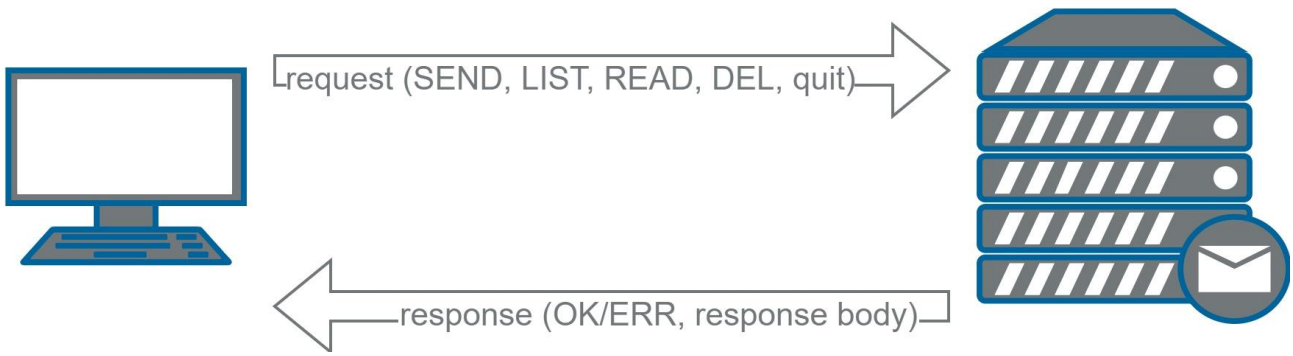
FH University of Applied Sciences TECHNIKUM WIEN

TWMailer Pro

Grundlagen verteilter Systeme WS22

Jasmin Duvivié, Si Si Sun



www.technikum-wien.at

client server architecture

Der server wird über die command line als superuser gestartet, damit dieser ins directory var/spool/mail schreiben kann. Dann wartet er auf dem port 6543 auf client Anfragen. Der TWMailer Pro ist mithilfe von forking concurrent. Wenn eine Verbindung zu einem client aufgebaut wird, schließt der parent process den connection socket und wartet auf neue Verbindungen sowie auf die Kindprozesse. Der child process schließt den listening socket, handelt die Kommunikation mit dem Client ab und schließt zum Schluss den connection socket, wenn die Verbindung zum Client nicht mehr benötigt wird. Die Synchronisation mithilfe von file description locks ist sich aus Zeitgründen nicht mehr ausgegangen.

Der client stellt zum localhost auf dem port 6543 eine Verbindung her und kommuniziert über das in der Angabe definierte Protokoll (SEND, LIST, READ, DEL, quit), nachdem er sich eingelogged hat. Hierbei sendet der Client seinen username und sein password and den server. Dieser kontrolliert in einem ldap searchquery an den FH-LDAP-Server, ob der user existiert und ob dieser user mit diesem Passwort zu einem query berechtigt ist. Wenn ja, stimmt das Passwort und der user kann requests an den server senden. Wenn nein, wird der user zur erneuten Eingabe aufgefordert. Nach drei falschen Versuchen, muss der user eine Minute warten (sleep(60)).

Der server parsed den request und sendet entsprechend der Angabe bei Erfolg eine response mit OK und, falls anwendbar, einem response body (z.B. Nachrichteninhalt) bzw. bei Fehlern eine response mit ERR und einem Hinweis auf die Fehlerursache (switch(errno)).

used technologies

Der sourcecode für den TWMailer Pro ist in Visual Studio Code in C und C++ geschrieben und baut auf den samples aus dem moodle Kurs der Lehrveranstaltung auf.

Zur Netzwerkkommunikation werden die C-Libraries sys/types.h, sys/socket.h, sys/stat.h, netinet/in.h, arpa/inet.h und unistd.h sowie errno.h zum errorhandling, string.h für c-string operations, dirent.h für directory operations und stdlib.h sowie stdio.h für Standard-C-Funktionen verwendet. Für LDAP wird ldap.h du lber.h verwendet.

Es gibt ein Makefile, dass zum Komplieren verwendet werden kann. Dieses verwendet zum Kompilieren gcc und g++.

development strategies and adaptions

Zunächst haben wir die Funktionen des servers (parsen des requests, ausführen der directory operations) sowie das Einlesen des requests am client sowie den ldap query jeweils isoliert implementiert. Dazu haben wir uns einige Male entweder vor Ort an der FH oder online in Discord getroffen und über die Live Share Funktion von Visual Studio Code über den code gesprochen und ihn gemeinsam geschrieben.

Nachdem die einzelnen requirements funktioniert haben, haben wir die code Teile in das ClientServerSample Beispiel aus dem moodle Kurs eingebaut. Zunächst hat an dieser Stelle alles eher halb funktioniert, da wir die Kommunikation noch ein wenig an unser Protokoll anpassen mussten. Dazu haben wir vor allem mithilfe von debug-messages genau verfolgt, wann wie viele bytes gesendet/empfangen wurden, um Fehler zu finden.