Dokumentace k projektu z předmětu ISA

Programování síťové služby:
Klient POP3 s podporou TLS

Eliška Kadlecová, xkadle34@stud.fit.vutbr.cz
Fakulta informačních technologií, Vysoké učení technické v Brně

Obsah

1. Úvod

Tento dokument je dokumentace k projektu Programování síťové služby: *Klient POP3 s podporou TLS* do předmětu ISA – Síťové aplikace a správa sítí na VUT FIT. Popíši zde klíčové části programu a mé řešení.

2. POP3 protokol

Internetový protokol POP3 se využívá k přístupu k emailové schránce, která je na vzdáleném serveru. Zkratka POP je z anglického *Post Office Protocol*, číslo 3 značí třetí verzi tohoto protokolu. Hlavní výhodou tohoto protokolu je, že umožňuje přístup k e-mailům i off-line. Protokol funguje tak, že se připojí k serveru, stáhne zprávy (a většinou je rovnou i smaže) a uloží je na lokální počítač. Zde je možné je prohlížet i bez připojení k internetu a zároveň se tímto způsobem předchází zaplnění INBOXu.

3. Návrh programu

V této části se zaměřím na několik klíčových částí projektu a popíšu můj způsob řešení.

3.1. Zpracování parametrů

Jakmile se program spustí, zavolá se funkce pro zpracování parametrů, která vrátí strukturu, ve které jsou tyto parametry uloženy. Z této struktury se pak jednotlivé parametry „vybírají“ a dávají do argumentů dalším volaným funkcím. Samotné zpracování probíhá ve funkci *char* parseArg(char ** begin, char ** end, const std::string & opt, bool value),* která prochází všechny parametry a hledá chtěný parametr. Pokud je navíc value zadáno jako true, funkce uloží to, co je za hledaným argumentem (například argument *-p 115* zpracuje tak, že vrátí 115, která se uloží do zmíněné struktury).

3.2. Připojení k serveru

Pokud je zadán parametr *-T,* vybere se zabezpečené připojení k serveru, pokud není, vybere se připojení „obyčejné“. Při nezabezpečeném připojení využívám knihovnu *socket.h*, při připojení zabezpečeném využívám knihovnu *openssl/bio.h.*

3.3. Autorizace

Po úspěšném připojení se v hlavní funkci (v souboru *main.cpp*) otevře autorizační soubor, ze kterého se načtou údaje k přihlášení. Autorizační soubor je popsán níže v podkapitole 5.1.

3.4. Komunikace se serverem

Při komunikaci se serverem klient posílá požadavky serveru, který na ně odpovídá buď *+OK <odpověď serveru>* pokud se zadařilo, nebo *-ERR <odpověď serveru>*, pokud došlo k nějaké chybě, která je většinou popsána v odpovědi serveru. Proto se vždy, když přijde od serveru nějaká odpověď, musí odpověď testovat, zda byl příkaz proveden úspěšně. Příkazy jsem studovala z RFC 1939.

3.5. Ukládání zpráv

Na příkaz *RETR n* (*n* je číslo zprávy, určuje server) server (pokud je příkaz úspěšný) odpovídá nejprve jedním řádkem, kde vypíše pozitivní odpověď, následně pošle celou zprávu a nakonec pošle *„.\r\n“*, čímž dá najevo, že je zpráva ukončena. Jelikož náš klient vyžaduje stahování vždy všech zpráv, je nutné nejprve zjistit, kolik jich ve schránce aktuálně je. To zjišťuji příkazem *STAT*. Následně v cyklu volám příkaz *RETR* pro stahování zpráv a ty po lehké úpravě (pouze odstranění prvního řádku, což je pozitivní odpověď serveru na příkaz *RETR* a odstranění zmíněné tečky s CRLF) ukládám do

dané složky jako samostatně pojmenované *.txt* soubory. Pokud složka neexistuje, klient se ukončí s chybou.

3.6. Mazání zpráv

Jak jsem zmiňovala výše, protokol POP3 funguje tak, že stažené zprávy většinou přímo maže ze serveru. Proto pokud je zadán parametr *-d* posílám na server příkaz *DELE n* a smažu všechny v něm obsažené zprávy (které ale předtím stáhnu do určené lokální složky). V lokální složce zprávy nechávám.

4. Implementace

Klient je naimplementován v jazyce C++, především kvůli podpoře řetězců a možnosti objektového návrhu. Hlavní metody se nachází v souboru *pop3.cpp* s hlavičkovým souborem *pop3.hpp*. Nyní zde uvedu zásadní metody se stručným popisem:

- *bool connect_server(std::string server, int port);*
- *bool connect_server_sec(std::string server, int port, std::string cert_dir, std::string cert_file);*
- *bool login(std::string username, std::string password);*
- *bool quit();*
- *bool send_command(std::string command)* – metoda pro posílání příkazu na server (pro příkazy bez identifikátoru), obdobná funkce je pro zabezpečenou verzi
- *bool send_command(std::string command, int num)* – metoda pro posílání příkazu na server (pro příkazy s identifikátorem), obdobná funkce je pro zabezpečenou verzi
- *bool get_response()* – získává odpověď serveru, v nekonečném cyklu volá *read_from_server()* a načítá odpoveď do bufferu pro odpovědi
- *std::string read_from_server()* – čte odpovědi ze serveru, vrací string s odpovědí (obdobná je metoda *read_from_server_sec()* pro zabezpečenou verzi)
- *size_t is_end_of_message(std::string msg)* – testuje, zda odpověď server již ukončil (buď hledá *CRLF.CRLF* nebo jen *CRLF* podle typu příkazu
- *bool downloadMsg(std::string out_dir)* – funkce pro stahování zpráv, popsáno výše v 3.5
- *bool dele()* – posílá příkaz *DELE n* tolikrát, kolik je zpráv na serveru
- *bool retr (int a)* – pošle příkaz *RETR n*
- *bool stat ()* – pošle příkaz *STAT*, zjištěný počet zpráv uloží do proměnné *numMsg*, ke které mají přístup všechny metody

5. Použití programu

Aby se vytvořil spustitelný soubor *popcl*, je nutné soubory přeložit příkazem *make*. Použití je následující:

popcl <server> [-p <port>] [-T|-S [-c <certfile>] [-C <certaddr>]] [-d][-n] -a <auth_file> -o <out_dir>

nebo

popcl --help pro vypsání nápovědy.

Parametry mohou být v libovolném pořadí.
Popis parametrů:

- Povinně je uveden název <*server*> (IP adresa, nebo doménové jméno) požadovaného zdroje.
- Volitelný parametr *-p* specifikuje číslo portu <port> na serveru.
- Parametr *-T* zapíná šifrování celé komunikace (pop3s), pokud není parametr uveden použije se nešifrovaná varianta protokolu.

- Parametr *-S* naváže nešifrované spojení se serverem a pomocí příkazu STLS (RFC 2595) přejde na šifrovanou variantu protokolu.
- Volitelný parametr *-c* definuje soubor <*certfile*> s certifikáty, který se použije pro ověření platnosti certifikátu SSL/TLS předloženého serverem (použití jen s parametrem *-T*, nebo *-S*).
- Volitelný parametr *-C* určuje adresář <*certaddr*>, ve kterém se mají vyhledávat certifikáty, které se použijí pro ověření platnosti certifikátu SSL/TLS předloženého serverem. (Použití jen s parametrem *-T*, nebo *-S*.)
- Pokud není uveden parametr *-c* ani *-C*, pak použijte úložiště certifikátů získané funkcí *SSL_CTX_set_default_verify_paths()*.
- Při použití parametru parametru *-d* se zašle serveru příkaz pro smazání zpráv.
- Při použití parametru parametru *-n* se bude pracovat (číst) pouze s novými zprávami.
- Povinný parametr *-a* <*auth_file*> vynucuje autentizaci (příkaz *USER*), obsah konfiguračního souboru <*auth_file*> je zobrazený níže.
- Povinný parametr *-o* <*out_dir*> specifikuje výstupní adresář <*out_dir*>, do kterého má program stažené zprávy uložit.

5.1. Konfigurační soubor s autentizačními údaji:

Tento soubor obsahuje jméno a heslo k přihlášení do určité schránky na serveru. Má přesně danou strukturu:

username = jmeno

password = heslo

Přičemž za *jmeno* a *heslo* se dosadí konkrétní hodnoty.

5.2. Omezení aplikace

Klient neumí pracovat s parametry *-S* a *-n*. Rozpozná je, ale dále s nimi nepracuje.

Zdroje

- Zadání projektu ve WIS
- RFC 1939: https://tools.ietf.org/html/rfc1939
- Kapitola 4 z knihy Síťové aplikace a jejich architektura (Matoušek, Petr): https://wis.fit.vutbr.cz/FIT/st/course-files-st.php?file=%2Fcourse%2FISA-IT%2Ftexts%2Fkapitola4.pdf&cid=12191
- Tutoriál k OpenSSL: https://www.ibm.com/developerworks/library/l-openssl/