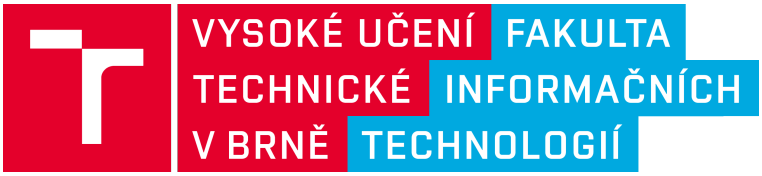
ISA - Zadanie POP3 klient

Richard Seipel - xseipe00

October 2021



Obsah

1 Základy POP3

Post Office Protocol verzie 3 je štandardný internetový protokol na aplikačnej vrstve. Používa sa na sťahovanie správ z emailového servera do lokálneho zariadenia prostredníctvom TCP/IP spojenia [4]. V dnešnej dobe má stále využitie pre jeho jednoduchosť.

1.1 Fázy komunikácie

- **Fáza spojenia** - prebieha pri pripájaní klienta na server, štandartne na port 110 alebo 995
- **Fáza autorizácie** - počas tejto fázy klient zadáva príkazy s prihlasovacími údajmi
- **Fáza transakcií** - nastáva po úspešnom prihlásení, klient môže na server zasielať príkazy na prácu so správami, až kým nie je komunikácia ukončená
- **Fáza aktualizácie** - nastáva po zadaní príkazu pre ukončenie komunikácie [4]

1.2 Príkazy

1.2.1 Validné v autorizačnej fáze

- `USER <meno>` - zadanie mena
- `PASS <heslo>` - zadanie hesla
- `APOP <meno> <hash_hesla>` - volitelný príkaz - obchádza odoslanie hesla v nechránenej podobe
- `QUIT` - ukončí komunikáciu

1.2.2 Validné v transakčnej fáze

- `STAT` - vráti počet správ na serveri a ich celkovú veľkosť
- `LIST [<číslo>]` - vráti zoznam správ na serveri a veľkosť každej z nich (parameter špecifikuje jednu správu, ktorá sa má zobraziť)
- `RETR <číslo_správy>` - vráti vybranú správu
- `DELE <číslo_správy>` - pripraví vybranú správu na vymazanie
- `NOOP` - nerobí nič, ak je všetko v poriadku server vráti pozitívnu odozvu
- `RSET` - resetuje zoznam správ určených na vymazanie
- `QUIT` - ukončí komunikáciu a nastane fáza aktualizácie (prebieha mazanie označených správ)

- `TOP <číslo_správy> <počet_riadkov>` - volitelný príkaz - vráti hlavičku a niekoľko riadkov vybranej správy
- `UIDL [<číslo>]` - volitelný príkaz - vráti zoznam správ na serveri a pre každú jej unikátne id na serveri (parameter špecifikuje jednu správu, ktorá sa má zobraziť)

*Volitelný príkaz nemusí byť implementovaný serverom [4].

1.3 Odpoveď servera

Odpovede servera začínajú `+OK`, ak je všetko v poriadku, alebo `-ERR`, ak nastala chyba. Jednoriadkové odpovede Končia znakmi `CRLF` (`\r\n`) a viacriadkové postupnosťou `CRLF.CRLF` (`\r\n.\r\n`) [4].

2 Práca s klientom

2.1 Spustenie

Ak si používateľ chce stiahnuť všetky správy zo servera s nezabezpečeným pripojením, spustí program príkazom:

```
./popcl <server> -a <autentifikačný_súbor> -o <výstupný_priečinok>
```

Povinné parametre:

`<server>` - ip adresa servera alebo jeho doménové meno

`-a <autentifikačný_súbor>` - cesta k súboru s menom a heslom pre prihlásenie na server

`-o <výstupný_priečinok>` - cesta k priečinku pre uloženie stiahnutých správ

Volitelné parametre:

`-p <port>` - pre výber čísla portu

`-n` - pre stiahnutie len nových správ

`-d` - pre vymazanie všetkých správ na serveri

`-T` - spustí šifrovanie celej komunikácie

`-S` - zaháji šifrované spojenie po pripojení k serveru pomocou príkazu STLS

`-c <cesta_k_súboru>` - pre pridanie cesty k súboru s certifikátom

`-C <cesta_k_priečinku>` - pre pridanie cesty k priečinku s cerifikátmi

Parameter `<server>` sa zadáva ako prvý. Všetky ostatné parametre môžu byť použité v ľubovoľnom poradí. Bez zadaného parameta `-p` je predvolené číslo portu `110` pre nezabezpečenú komunikáciu a `995` pre zabezpečenú, podľa registra čísel portov organizácie IANA [2].

2.2 Autentifikačný súbor

Autentifikačný súbor slúži na uloženie prihlasovacích údajov k serveru. Jeho štruktúra je následovná:

```
username = <prihlasovacie_meno>
password = <heslo>
```

Prihlasovacie meno môže byť celá emailová adresa alebo len jej časť pred zavináčom.

2.3 Výstupný priečinok

Správy sú ukladané ako samostatné súbory do priečinka zvoleného pri spustení programu. Názov každého súboru sa skladá z id správy, ktorú obsahuje a prípony `txt`. Spolu so správami je uložený aj súbor `last_message_marker`, obsahujúci ID najnovšej z ukladaných správ.

Používateľ si môže správy ľubovoľne otvárať, presúvať, či mazať a pri ďalšom ukladaní do rovnakého priečinka s parametrom `-n` budú zo servera stiahnuté len správy novšie, ako správa ktorej id je uložené v pomocnom súbore. Používateľ tak môže mať napríklad jeden priečinok pre každú emailovú schránku a v ňom si ukladať, čítať a triediť poštu. V prípade potreby môžu byť všetky správy na serveri vymazané použitím parametra `-d` a ostanú prístupné len lokálne.

3 Implementácia

3.1 Spracovanie argumentov

Spracovanie argumentov je implementované v triede `Arguments`. Ich načítanie sa spustí metódou `loadArguments`, ktorá s pomocou ďalších privátnych metód argumenty uloží, skontroluje a prípadne nastaví predvolené hodnoty volitelným parametrom, ktoré neboli zadané. Prístup k ich hodnotám a tiež k informácii, či daný argument bol použitý, je potom možný cez radu atribútov.

3.2 Autentifikácia

Trieda `Auth` obsahuje prihlasovacie údaje a metódu `loadAuth` pre ich načítanie.

3.3 Pripojenie a základná komunikácia so serverom

Pre implemetáciu pripojenia na server a jeho zabezpečenia som využil knižnicu openssl. Informácie pre prácu s ňou som čerpal z tutorálu na portáli IBM Developer [3]. Samotná implementácia pripojenia sa nachádza v triede `Connection`. Trieda obsahuje odkazy na openssl objekty bio, ctx a ssl, ktoré sú potrebné na vytvorenie pripojenia, kontrolu certifikátov, čítanie a zápis.

Pripojenie je možné nadviazať jednou z metód `establishConnection`, `establishSecureConnection` alebo `establishSTLSConnection`, podľa vybraného spôsobu zabezpečenia. Zatiaľ čo metódy `read` a `write` slúžia ako všeobecný základ komunikácie, `readLine` a `readMessage` zabezpečujú čítanie jednoriadkovej alebo viacriadkovej odpovede, podľa vyššie popísaných štandardov POP3.

3.4 Práca s reťazcami a súbormi

Namespace `Response` združuje funkcie pre spracovanie textových vstupov a výstupov, vyhľadávanie informácií v reťazcoch či súboroch, prípadne aj ich úpravu pre ďalšie použitie. Napríklad odstránenie prvého a posledného riadku viacriadkovej odpovede. Ďalej obsahuje funkciu `checkIfStartsWithOk`, pre kontrolu prvých znakov odpovede servera a taktiež funkcou `checkDirectory` pre kontrolu priečinka, s ktorého súbormi sa má ďalej pracovať.

3.5 Hlavný tok programu

Po spustení programu sú vytvorené objekty tried `Connection`, `Arguments` a `Auth`. Následne sú objekty naplnené dátami a podľa prijatých argumentov sa vyberie spôsob pripojenia.

Po úspešnom pripojení sa odošlú serveru príkazy na prihlásenie používateľa zadanými prihlasovacími údajmi a ak je prihlásenie úspešné, program ďalším príkazom zistí počet správ na serveri. Ďalej dochádza k samotnému ukladaniu správ. Program si od servera postupne vyžaduje správy a ukladá ich do vybraného priečinka. Postupuje od najnovších k starším a ak je zvolený parameter `-n` kontroluje, či sa id čítanej správy nezhoduje s id poslednej vyzdvihnutej správy z predchádzajúceho spustenia.

Po dokončení sťahovania prípadne program vymaže všetky správy na serveri, ak bol zadaný parameter `-d` a následne ukončí spojenie so serverom príkazom `QUIT`. Ak počas behu programu nastane chyba, je vypísaný jej krátky popis na chybový výstup, uvoľní sa alokovaná pamäť a program skončí s návratovou hodnotou -1.

4 Špecifické riešenia

4.1 Zabezpečenie po pripojení

Zaujímavou časťou implementácie je nadviazanie STLS spojenia, kde bolo potrebné upraviť existujúci `BIO` objekt na zabezpečený. K tomu mi pomohla zverejnená implementácia funkcie `BIO_new_ssl_connect` na fóre stackoverflow, ktorej kód som použil na vytvorenie zabezpečenia, no vynechal som časť, kde bol vytváraný `BIO` objekt a nahradil som ho mojim, už vytvoreným objektom.

4.2 Sťahovanie nových správ

Dalšia zaujímavá časť bola implementácia sťahovania nových správ, pri zadaní prepínača -n. Na začiaku som túto funkcionalitu implementoval ukladaním času a dátumu najnovšej uloženej správy, konvertovaného do formátu unix timestamp a následným sťahovaním len správ s novšou hodnotou. No hneď na to som zistil, že takto nemám ako rozoznávať správy odoslané v rovnakej sekunde a prešiel som na ukladanie id správy.

Nové správy sú tak sťahované dovtedy, kým sa id sťahovanej správy nerovná tomu uloženému. Nevýhodou je, že ak by bola posledná uložená správa zo serveru vymazaná, program sa pri ďalšom sťahovaní nezastaví a stiahne zo serveru aj všetky správy doručené pred ňou. Nakoľko však program dokáže vymazávať zo serveru len všetky správy naraz, nenastáva žiadny problém. Ak by to bolo potrebné mohol by som kontrolovať aj čas aj id naraz. Lokálne s vymazávaním správ nie je problém, keďže id je uložené v samostatnom súbore.

5 Testovanie

Na testovanie som od začiatku používal lokálny emailový server, ktorý som vytvoril a nakonfiguroval pomocou programu hMailServer. Do vytvorených účtov som sa prihlásil cez emailový kilent Microsoft Outlook, aby som si mohol odosielať testovacie správy. Na tomto serveri som otestoval všetky tri druhy pripojenia, aj použitie vlastného súboru a priečinka s certifikátom, ktorý som si spolu so súkromným a verejným kľúčom pre server vytvoril.

Taktiež som otestoval ostatné parametre, chybové výpisy a korektné uvoľňovanie pamäte, aj pri ukončení programu vyvolaním výnimky. Neskôr som si pre účely testovania na serveri eva.fit.vutbr.cz vytvoril konto na serveroch centrum.sk a seznam.cz. Funkcionalitu som porovnával s programom `telnet` pre testovanie nešifrovanej komunikácie a s programom `openssl s_client` pre testovanie šifrovanej komunikácie, vrátane módu `starttls` [1].

Literatúra

[1] *OpenSSL s_client*. openssl.org. Dostupné z: `https://www.openssl.org/docs/man1.1.1/man1/openssl-s_client.html`.

[2] *Service Name and Transport Protocol Port Number Registry*. IANA. Dostupné z: `https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml`.

[3] Ballard, K. *Secure programming with the OpenSSL API*. IBM Developer, 2017. Dostupné z: `https://developer.ibm.com/tutorials/l-openssl/`.

[4] Myers, J. a Rose, M. *RFC1939: Post office protocol-Version 3*. RFC Editor, 1996. Dostupné z: `https://datatracker.ietf.org/doc/html/rfc1939`.