ISA Projekt: Klient POP3

Dokumentácia

Jakub Duda, xdudaj02

13. 11. 2021

Obsah

1. Úvod

Protokol POP3 je protokol na aplikačnej vrstve využívaný emailovými klientami na sťahovanie emailov z emailových serverov. Protokol POP3 bol v istom čase najpoužívanejším emailovým protokolom, no v súčasnej dobe je postupne vytláčaný do úzadia najmä protokolom IMAP. Základným rozdielom medzi týmito dvomi protokolmi je ich jednoduchosť, resp. robustnosť a princíp práce s emailovými správami.

Kým protokol POP je omnoho jednoduchší a hlavnou myšlienkou je, že používateľ nemá vždy stabilné pripojenie na internet, a preto je vhodné, aby boli emaily dostupné aj bez internetového pripojenia. Z toho dôvodu, hlavným princípom funkcionality protokolu POP3 je pripojenie sa na server, stiahnutie emailov, nasledované voliteľným vymazaním všetkých emailov v emailovej schránke.

[1]Na druhej strane protokol IMAP je zložitejší a poskytuje väčšie množstvo funkcií. Na rozdiel od protokolu POP3 maily nevymazáva. Funkcionalita navyše zahŕňa napríklad možnosť vytvárania súborov, označovania jednotlivých emailov rôznymi značkami alebo pripojenie z viacerých zariadení. Tento protokol je ale taktiež obťažnejší na implementáciu.

Protokol POP3 sa štandardne pripája pomocou TCP pripojenia na porte 110. Sedenie medzi klientom a serverom sa postupne môže nachádzať v troch rôznych stavoch, a to autorizácia, transakcia a aktualizácia.

Komunikácia prebieha pomocou jednoduchých správ. Klient posiela preddefinované príkazy, zvyčajne nasledované argumentom. Niektoré príkazy nemajú žiadny argument. Server na každý požiadavok odpovedá zvyčajne jednoduchou jednoriadkovou správou. Začiatok odpovede je buď „+OK“ alebo „-ERR“, čo značí úspech, resp. neúspech klientovho požiadavku. V prípade úspechu nasleduje odpoveď a v prípade neúspechu nasleduje chybová hláška. Spomínaný začiatok odpovede a tiež preddefinované príkazy musia byť napísané veľkými písmenami. Jednotlivé požiadavky a odpovede sú zakončené znakom nového riadku, ktorý je výlučne určený ako dvojica znakov CR a LF.

Na niektoré požiadavky zvyčajne server odpovedá viacriadkovou odpoveďou. Je to hlavne príkaz na vypísanie zoznamu všetkých správ a najčastejšie používaný príkaz na vrátenie obsahu konkrétnej správy. Na označenie konca danej odpovede server odošle takzvaný ukončovací oktet, ktorým je znak bodky nachádzajúci sa samotný na riadku, teda postupnosť piatich oktetov „CRLF.CRLF“. Keďže takáto kombinácia sa môže v tele emailu kľudne vyskytnúť, v prípade, že znak bodky je prvým znakom riadku, je tento znak duplikovaný.

Po pripojení odošle server „pozdravenie“, prvú odpoveď. Sedenie sa následne dostane do stavu autorizácie, kedy server overuje totožnosť klienta. Existujú dva mechanizmy. Prvým je jednoduchá autorizácia pomocou dvoch príkazov USER a PASS, ktoré sú nasledované používateľským menom, resp. heslom používateľa. Nevýhodou tohto prístupu je nebezpečenstvo jednoduchého odcudzenia prihlasovacích údajov, ktoré sú posielané cez internet bez akéhokoľvek zabezpečenia. Z toho dôvodu existuje bezpečnejšia možnosť prihlásenia pomocou príkazu APOP, ktorý ale tento program neimplementuje.

Po úspešnej identifikácii klienta, sedenie prejde do stavu transakcie. V tomto stave server uzamkne emailovú schránku a otvorí ju pre komunikáciu s týmto jedným klientom. Klient môže

[1] Post Office Protocol. (13.11.2021). https://en.wikipedia.org. https*://en.wikipedia.org/wiki/Post_Office_Protocol*

následne posielať rôzne príkazy na zobrazenie stavu emailovej schránky a na manipuláciu s jednotlivými emailovými správami. Príkaz STAT zobrazí stav schránky, t.j. počet emailov a ich veľkosť v oktetoch. Príkaz LIST vypíše základnú jednoriadkovú informáciu o emailovej správe skladajúcej sa z čísla emailu a jeho veľkosti v oktetoch, pre email špecifikovaný ako argument pomocou jeho čísla. V prípade, že nie je špecifikovaný žiadny email, server vypíše takúto informáciu pre každý email v schránke. Príkaz RETR vyžaduje jeden argument, ktorým je číslo správy. Odpoveďou je veľkosť správy v oktetoch nasledovaná celým obsahom emailu. Príkaz DELE slúži na vymazanie správy a taktiež očakáva určenie konkrétnej správy pomocou čísla. Po použití tohto príkazu, správa nie je vymazaná, ale zatiaľ len označená ako určená na vymazanie. Medzi ďalšie príkazy, ktoré ale tento program neimplementuje patria napríklad NOOP, RSET alebo rozširujúce príkazy TOP či UIDL.

Použitím bezargumentového príkazu QUIT, klient odošle podnet na prechod z transakčného stavu do stavu aktualizácie. V tomto stave server vymaže všetky správy, ktoré sú označené na vymazanie. Následne server uvoľní exkluzívne zámok na mailovej schránke a ukonči pripojenie.

Pri vysvetlenom postupe pripojenie TCP protokolom, sa ale vynára problém bezpečnosti takéhoto programu, keďže takáto komunikácia sa dá, napríklad pomocou programu Wireshark, veľmi jednoducho zachytiť a rovnako jednoducho prečítať. Preto tento program implementuje aj možnosť zabezpečenej komunikácie, ktorá je vykonaná pomocou protokolu TSL bežiacom na porte 995.

Posledným nespomenutým príkazom je príkaz STLS. Tento príkaz môže klient zaslať na server, čím žiada server o vylepšenie nezabezpečeného pripojenia na pripojenie zabezpečené.

2. Funkcionalita

Program *popcl* implementuje funkcionalitu jednoduchého emailového klienta. Pomocou protokolu POP3 sa s využitím prihlasovacích údajov uložených v súbore špecifikovanom ako argument programu pripojí na server špecifikovaný ako argument programu a uloží stiahnuté emaily do adresára špecifikovaného ako argument programu.

Program ponúka množstvo prepínačov príkazového riadku, ktorými sa dá ovplyvniť jeho funkcionalita. Program vyžaduje tri povinné argumenty a umožňuje použitie ďalších siedmych voliteľných.

Prvý povinný argument je meno alebo IP adresa servera, ktorý sa musí vyskytovať ako prvý. Ďalšie povinné argumenty sú cesta k súboru obsahujúcemu prihlasovacie údaje, ktorý je predchádzaný prepínačom `-a` a cesta k adresáru, do ktorého bude zapisovaný výstup programu, ktorý je predchádzaný prepínačom `-o`.

Voliteľný prepínač `-n` obmedzuje stiahnutie emailov na nové emaily, teda také emaily, ktoré ešte nikdy neboli stiahnuté. Použitie prepínača `-d` nastaví, že po stiahnutí správ zo servera budú na serveri všetky správy vymazané. Funkcionalita týchto dvoch prepínačov nijako neovplyvňuje ten druhý, a teda môžu byť kombinované. V prípade kombinácie teda program stiahne nové správy a na serveri vymaže všetky správy.

Voliteľným prepínačom `-p`, ktorý očakáva jeden argument, je možné zmeniť „defaultné“ číslo portu, ktorý bude použitý na komunikáciu.

Voliteľný prepínač `-T` značí, že bude použité zabezpečené spojenie s využitím TLS. Prepínač `-S` značí, že pripojenie k serveru prebehne klasickou formou a bezprostredne po pripojení bude toto spojenie vylepšené na zabezpečené. Pomocou prepínačov `-c` a `-C` očakávajúcich po jednom argumente, je možné špecifikovať cestu k súboru, z ktorého majú byť načítané certifikáty potrebné pri zabezpečenom pripojení, resp. cestu k takémuto adresáru. Prepínače `-T` a `-S` nie je možné kombinovať. Prepínače `-c` a `-C` je možné kombinovať, no môžu byť použité len v prípade, že je použitý jeden z prepínačov -T alebo `-S`.

Program striktne vyžaduje aby súbor s prihlasovacími údajmi začínal nasledovne:

> username: používateľské meno
> password: heslo používateľa

a tiež, aby využíval linuxové ukončenie riadku.

Príkladné spustenie programu je možné nájsť v kapitole číslo 4.

3. Implementácia

3.1. Všeobecné informácie

Program *popcl* je implementovaný v jazyku C++. Program ale nevyužíva veľa konštrukcií objektového programovania a väčšina problémov je riešených tvorbou funkcií.

Program využíva viacero importovaných knižníc. Okrem základných knižníc ako `iostream`, `fstream`, `sstream`, `cstring` a `sys/stat.h` program využíva knižnice z rodiny OPENSSL: `openssl/bio.h`, `openssl/ssl.h` a `openssl/err.h`.

Volania všetkých funkcií z týchto knižníc sú ošetrené konštrukciou `if-else`, ktorá sleduje možné chyby a následne volá adekvátnu funkciu na ukončenie programu, čo je v nasledujúcom texte naznačené použitím slova „pokúsiť sa“ vždy, keď je takéto volanie spomenuté.

Program si po prvom spustení s konkrétnym používateľom na konkrétnom serveri pre každého vytvorí špeciálny súbor v adresári *.config*, ktorý je vytvorený pri preklade programu. Do tohto súboru si následne program ukladá identifikačné reťazce stiahnutých správ, aby mohol pri opakovanom spustení kontrolovať novosť správ na serveri. Súbor má názov emailovej adresy daného používateľa na danom serveri. Odstránenie alebo presunutie tohto adresára alebo existujúcich súborov v ňom znemožní alebo obmedzí správnu funkcionalitu možnosti `-n`.

3.2. Globálne definície

3.2.1. Štruktúry

Program obsahuje definovanú štruktúru `data` obsahujúcu dáta využívané programom pri tvorbe pripojenia k serveru a následne pri komunikácií. Väčšina z týchto dát sú tie špecifikované pomocou možností príkazového riadku.

3.2.2. Ďalšie globálne definície

V programe je definovaných aj niekoľko globálnych konštánt. Týmto spôsobom je definovaná napríklad veľkosť bufferu používaného pri výmene správ so serverom, či čísla štandardných portov alebo hodnoty výstupných kódov programu. 0 značí úspešné vykonanie, 1 značí chybu súvisiacu s argumentami programu, 2 značí chybu súvisiacu s *OPENSSL* knižnicami a 3 značí chybu vzniknutú pri komunikácií so serverom.

3.2.3. Funkcie

Program obsahuje niekoľko funkcií, ktoré sú používané mnoho krát. Jednou takouto funkcie je funkcia `error_exit()`, ktorá je volaná pri výskyte kritickej chyby. Táto funkcia uvoľní zabratú pamäť, vypíše oznámenie o chybe, ktorého znenie dostáva ako parameter a ukončí program s chybovým kódom, ktorý tiež dostane ako parameter.

Ďalšou často volanou fukciou je funkcia `pop3_read()`, ktorá zaobaľuje čítanie odpovede zo servera. Najprv táto funkcia vyčistí používaný buffer, potom sa pomocou funkcie `BIO_read()` pokúsi

prečítať odpoveď od serveru. Jedným z parametrov je pravdivostná hodnota, ktorá značí, či v odpovedi očakávame informáciu o úspechu požiadavku v tvare +OK alebo -ERR. Pri čítaní dlhšej správy je totiž funkcia volaná opakovane na prečítanie jednej odpovede od serveru. V prípade kladného nastavenia tohto parametru, prebehne kontrola a funkcia vracia kladnú pravdivostnú hodnotu v prípade odpovede +OK a zápornú v prípade -ERR.

Protipólom k funkcií `pop3_read()` je funkcia pop3_write(), ktorá posiela požiadavky na server, zapisuje do daného sedenia. Ako parameter dostane táto funkcia znenie požiadavku, ku ktorému pridá povinné ukončenie jednoriadkovej správy, a teda pár CRLF a využitím funkcie `BIO_write()` sa pokúsi poslať tento požiadavok na server.

3.3. Činnosť programu

3.3.1. Spracovanie argumentov

Činnosť programu sa začína načítaním a skontrolovaním použitých možností príkazového riadku. Program vo vnútri bloku `try` volá funkciu `parse_args()`. Táto funkcia vykonáva kontrolu použitia správnych prepínačov, správnych kombinácií prepínačov, správneho použitia hodnôt pri prepínačoch a v niektorých prípadoch aj kontrolu správnosti týchto hodnôt. Príkladom je napríklad číslo portu alebo výstupný adresár. Kontrolou existencie výstupného adresára sa program snaží predísť zbytočnému vykonaniu pripojenia, v prípade že by následne nemohol zapisovať výstup. Načítané hodnoty prepínačov a použitie prepínačov bez hodnôt je uložené do adekvátnej štruktúry. V prípade objavenia chyby, funkcia vyvolá výnimku `arg_exception`, ktorá dedí od výnimky `runtime_error`.

Načítanie prihlasovacích údajov zo súboru prebieha vo funkcií `get_auth_data()`.

3.3.2. Inicializácia spojenia

Program pokračuje inicializáciou premenných a knižníc využívaných pri spojení so serverom. V prípade nastavenia zabezpečeného spojenia alebo bežného spojenia s vylepšením, program inicializuje objekt pre certifikáty a pokúsi sa ich načítať. Ak sú použité prepínače na definovanie umiestnenia certifikátov pokúsi sa načítať tieto. V prípade chyby, vypíše hlásenie a pokúsi sa načítať certifikáty zo štandardného systémového umiestnenia.

Následne je pomocou funkcie `BIO_new_connect()`, resp. `BIO_new_ssl_connect()` vytvorené spojenie so serverom. Program vykoná kontrolu vytvorenia spojenia a kontrolu návratovej hodnoty tejto funkcie. Následne sa program pomocou vlastnej funkcie `pop3_read()` pokúsi prečítať inicializačný pozdrav od serveru.

V prípade nastavenia vylepšenia pripojenia, program sa o toto pokúsi. Najprv zašle na server príkaz *STLS*, čím zažiada o vylepšenie spojenia. Potom ešte musí vylepšiť aj objekt pomocou, ktorého je toto spojenie v programe realizované. Toto sa vykoná vytvorením nového objektu typu BIO a „pridaním“ tohto objektu k existujúcemu objektu pomocou funkcie `BIO_push()`. Nakoniec je ešte v prípade zabezpečeného pripojenia potrebné vykonať kontrolu certifikátov.

3.3.3. Autentizácia spojenia

Pred prístupom do schránky, musí klient najprv preukázať svoju identitu. Vykonanie tohto procesu prebieha pomocou zaslania príkazu *USER* nasledovanom používateľským menom, prečítaním odpovede a v prípade kladnej odpovede, zaslaním príkazu *PASS* nasledovanom heslom používateľa. Ak sú zadané údaje správne a server ich overí, server by mal vytvoriť zámok na schránke a umožniť klientovi prístup. Zasielanie správ a prijímanie správ prebieha pomocou vlastnej funkcie `pop3_write_and_read()`, ktorá následne volá vlastné funkcie `pop3_write()`, resp. `pop3_write()`.

V prípade nesprávnych údajov alebo inej chybe vyskytnutej počas autentizácie sa program ukončí s adekvátnym hlásením a chybovým kódom.

3.3.4. Zistenie obsahu schránky

V prípade, že klient bol overený, môže začať vykonávať operácie povolené v transakčnom stave sedenia. Program najprv využije príkaz *STAT*. Z odpovede sa potom pokúsi získať počet správ v schránke pomocou vlastnej funkcie `pop3_get_number()`, ktorá sa pokúsi vrátiť číslo na druhej pozícii odpovede serveru. V prípade neúspechu funkcia vracia konštantu `INVALID_INT`.

Ďalej si program pripraví premenné na ukladanie počtov načítaných, stiahnutých a vymazaných správ, ktoré sú využité pri informatívnom výpise o činnosti programu pri správnom behu programu a môže začať so sťahovaním jednotlivých správ.

3.3.5. Načítanie a stiahnutie správ

Pomocou konštrukcie `for` s počtom opakovaní rovnom získanému počtu správ v schránke program vykoná nasledujúce príkazy pre každú správu. Program využije pre danú správu príkaz *LIST* s poradovým číslom danej správy v schránke. Ďalej program využije vlastnú funkciu `pop3_get_length()`, ktorá sa pokúsi prečítať číslo na tretej pozícií odpovede serveru, čo odpovedá veľkosti správy v oktetoch (bytoch).

Program pokračuje príkazom *RETR* s poradovým číslom danej správy v schránke a začne čítať odpoveď serveru, ktorý by mal poslať celú správu. Keďže správa je často väčšia ako použitý buffer, program pomocou konštrukcie `while` opakuje čítanie, dokým neprečíta očakávaný počet bytov.

Prečítané dáta si ukladá do predpripravenej premennej typu reťazec. S touto premennou následné vykoná niekoľko operácií, resp. kontrol. Najprv skontroluje, či je správa ukončená ukončujúcim oktetom, resp. sekvenciou oktetov. V prípade kladného výsledku tento oktet zo správy odstráni, inak ukončuje svoje činnosť s adekvátnou hláškou a výstupným kódom. Podobne program zo správy odstráni tzv. „byte-stuffed“ ukončovací oktet, ktorý vznikol duplikáciou náhodného výskytu ukončovacieho oktetu, ktorý ale nie je skutočným ukončovacím oktetom.

Následne program využije vlastnú funkciu `get_msg_id()`, ktorá prehľadá reťazec obsahujúci správu a pokúsi sa vrátiť identifikačný reťazec tejto správy. Pomocou vlastnej funkcie `get_recipient_addr()` sa program podobne pokúsi získať mailovú adresu klienta. Táto funkcia je dôležitá v prípade špecifikovania serveru pomocou IP adresy. Adresa používateľa je neskôr využitá pri ukladaní výstupu programu. Pomocou získaného identifikačného reťazca v prípade nastavenia stiahnutia nových správ skontroluje, či je daná správa nová a iba v kladnom prípade prejde na uloženie

správy. V prípade, že nastavenie stiahnutia nových správ nebolo použité program taktiež prejde na uloženie správy.

Program vytvorí nový súbor a uloží do neho obsah správy. Čo sa týka názvu súboru, ten sa skladá z tzv. „timestampu“, mailovej adresy používateľa a súčasného čísla správy na serveri spojených znakom -. Zložitosť názvu pramení z myšlienky vyhnutia sa prepisovaniu už stiahnutých správ novými. Z tohto dôvodu nie je možné použiť ako názov súboru predmet správy, aj keď toto riešenie by bolo pre používateľa asi najpríjemnejšie. Keďže program nie je obmedzený na jeden účet na jednom serveri, považoval sa za dôležité toto v názve súboru zahrnúť. Možno prijateľnejším riešením by bolo vytváranie stromovej štruktúry, resp. jednoduchých súborov pre rôznych používateľov na rôznych serveroch, no toto by nespĺňalo požiadavky zadania. Číslo správy na serveri nie je dostatočným identifikátorom, keďže zohľadňuje len súčasný stav schránky a pri vymazaní správ, by spôsobil prepis pôvodných správ. Tento identifikátor je však využitý na rozlíšenie správ stiahnutých pri jednom behu programu. Tzv. „timestamp“ je využitý na rozlíšenie správ stiahnutých v rôznych behoch programu, pričom počíta s tým, že program nebude spustený viackrát za jednu sekundu, je totiž vo formáte sekúnd. Nachádza sa na prvom mieste, aby mohli byť správy jednoducho zoradené podľa času stiahnutia.

Príklad názvu súboru:

```
timestamp-username@server-number
1636661028-isa.enjoyer123@seznam.cz-1
```

V prípade, že je použité nastavenie vymazania správ, program ešte pomocou príkazu *DELE* správu označí, že je určená na vymazanie.

3.3.6. Ukončenie činnosti programu

Na záver program pošle na server príkaz *QUIT*, čím by sa mali vymazať označené správy a ukončiť spojenie so serverom. Ako poslednú vec program vypíše informáciu o svojej činnosti, uvoľní pamäťové zdroje a ukončí svoj beh s výstupovým kódom `POPCL_OK`.

3.3.7. Nedostatky

Program neprevádza úplnú kontrolu formátu mailov a spolieha sa, že maily sú v správnom formáte IMF, požadovanom protokolom SMTP slúžiacom na odosielanie mailov.

Program tiež neprevádza úplnú kontrolu všetkých odpovedí serveru. Program väčšinou kontroluje, to s čím potrebuje pracovať, no spolieha sa, že server dodržiava formát správ požadovaný protokolom POP3, a teda na správnych miestach sú správne hodnoty. Nedodržanie tejto skutočnosti by pravdepodobne bolo kritické pri čítaní zoznamu mailov a čítaní samotného mailu.

4. Spustenie programu

Niekoľko príkladov spustení programu s ich výstupmi na príkazovom riadku:

```
$ ./popcl exampleserver.com -o maildir -a cred
popcl - OK: 10 messages downloaded.

$ ./popcl exampleserver.com -o maildir -a cred -n
popcl - OK: 4 new messages downloaded.

$ ./popcl exampleserver.com -o maildir -a cred -n
popcl - OK: 4 new messages downloaded.

$ ./popcl exampleserver.com -o maildir -a cred -T
popcl - OK: 10 messages downloaded.

$ ./popcl exampleserver.com -o maildir -a cred -S -c cert -n -d
popcl - OK: 4 new messages downloaded. 10/10 succesfully deleted.

$ ./popcl exampleserver.com -o maildir -a cred -S -C cert_dir
popcl - OK: 0 messages downloaded.

$ ./popcl exampleserver.com -o maildir
popcl - ARGUMENT ERROR: mandatory argument '-a' missing

*no internet connection
$ ./popcl exampleserver.com -o maildir -a cred
popcl - OPENSSL ERROR: cannot connect to server
```

5. Testovanie

Program bol dostatočne otestovaný. Na testovanie boli použité mailové servery *centrum.sk* a *seznam.cz*. Na testovanie zabezpečeného pripojenia bol použitý mailový server *seznam.cz*. Z dôvodu časovej tiesne nebolo prevedené dostatočné otestovanie načítavania certifikátov z užívateľom špecifikovaných súborov alebo adresárov.

6. Bibliografia

Základom môjho poznania o emailových protokoloch bola dokumentácia RFC.

1. Post Office Protocol - Version 3. (13.11.2021). www. https://datatracker.ietf.org/. *https://datatracker.ietf.org/doc/html/rfc1939*
2. Using TLS with IMAP, POP3 and ACAP. (13.11.2021). www. https://datatracker.ietf.org/. *https://datatracker.ietf.org/doc/html/rfc2595*
3. Post Office Protocol. (13.11.2021). https://en.wikipedia.org. https*://en.wikipedia.org/wiki/Post_Office_Protocol*

Základné použitie funkcií knižnice OpenSSL.

4. Secure programming with the OpenSSL API. (13.11.2021). https://developer.ibm.com. *https://developer.ibm.com/tutorials/l-openssl/*

Kontrola existencie adresára

5. Portable way to check if directory exists [Windows/Linux, C]. ivy, *https://stackoverflow.com/users/2656473/ivy*; Ingo Leonhardt, *https://stackoverflow.com/users/2470782/ingo-leonhardt.* (11.11.2021). Stack Overflow. *https://stackoverflow.com/questions/18100097/portable-way-to-check-if-directory-exists-windows-linux-c*

„Upgrade“ BIO objektu používaného na pripojenie k serveru.

6. OpenSSL: Promote insecure BIO to secure one. eko, *https://stackoverflow.com/users/4620235/eko*; Martin Prikryl, *https://stackoverflow.com/users/850848/martin-prikryl.* (11.11.2021). Stack Overflow. *https://stackoverflow.com/questions/49132242/openssl-promote-insecure-bio-to-secure-one*