

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Síťové aplikace a správa sítí
POP3 server

Obsah

1	Úvod	2
2	Dôležité pojmy	2
2.1	POP3	2
2.2	IMF	2
2.3	Maildir	2
3	Návrh a implementácia	3
3.1	Spracovanie argumentov	3
3.2	Pripojenie klientov	3
3.3	Spracovanie príkazov	3
3.4	Ukončenie signálom SIGINT	4
3.5	Pomocné súbory, reset	4
4	Použitie programu	4
5	Záver	4

1 Úvod

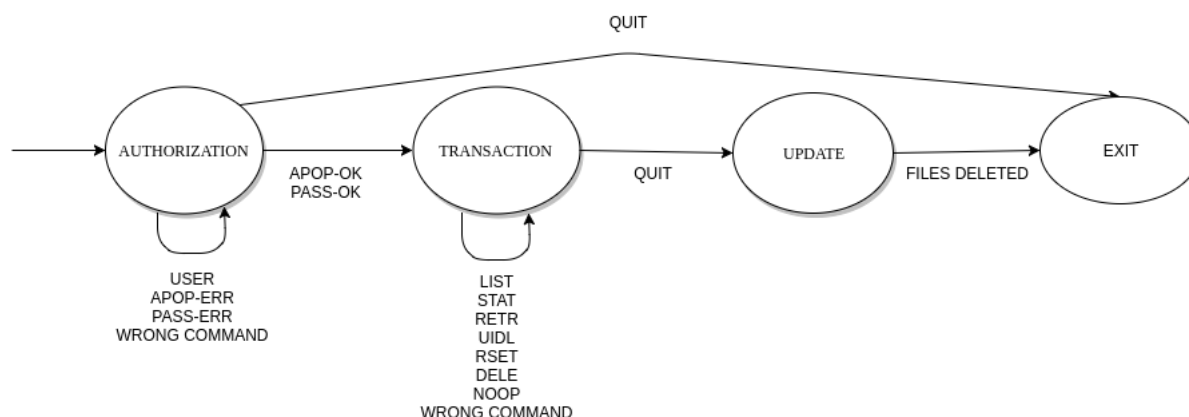
Táto dokumentácia popisuje program s názvom **popser** vytvorený ako projekt do predmetu Sít'ové aplikácie a správa sítí. Úlohou tohoto zadania bolo vytvoriť e-mailový server, ktorý pracuje nad protokolom POP3, spracováva príkazy klientov a poskytuje na ne odpovede. Program má implementovať celú špecifikáciu POP3(okrem sekcie č. 8) a má pracovať s e-mailmi vo formáte IMF. V nasledujúcich častiach sú popísané jednotlivé protokoly, popis riešenia jednotlivých problémov a ich implementácia.

2 Dôležité pojmy

Funkčnosť celej aplikácie je založená na niekoľkých protokoch a štandardoch. K pochopeniu zadania a k vypracovaniu projektu je potrebné ich najprv preštudovať a porozumieť.

2.1 POP3

Najdôležitejším protokolom v rámci celej implemetácie je protokol POP3(Post Office Protocol - Version 3) popísaný v RFC1939 [2]. Slúži na prístup k e-mailom na mailovom serveri ktorý podporuje tento protokol. Neposkytuje rozsiahle manipulačné operácie, zvyčajne klient sa pripojí k serveru, stiahne si maily a na serveri sa vymažú. Používa sa komunikácia typu klient-server. Klient po pripojení posíla príkazy serveru, server ich spracuje a poskytne odpoveď. Protokol popisuje presný formát správ a odpovedí. Odpoveď musí začať s +OK pri pozitívnej odpovedi, -ERR pri negatívnej a musí končiť s \r\n(CRLF). Pripojenie prechádza niekoľkými stavmi. Obrázok 1 popisuje stavy spojenia a prechody medzi nimi.



Obr. 1: Stavový automat pripojenia POP3.

2.2 IMF

Internet message format je popísaný v RFC5322 [3]. Udáva formát súborov s ktorými bude náš server pracovať. Každý e-mail sa má skladať z hlavičky a tela správy. Každý riadok súboru má byť ukončený s dvojicou znakov \r\n(tj. CRLF).

2.3 Maildir

Formát Maildir [1] popisuje potrebnú adresárovú štruktúru. Pričínok, ktorý bude použitý ako Maildir musí obsahovať adresáre **new**, **cur** a **tmp**, poprípade môže obašhovať nejaké ďalšie súbory alebo adresáre. V rámci tejto implementácie sa používa iba new a cur, keďže s tmp pracujú rôzne SMTP servery. Tiež popisuje že jednotlivé e-mailové súbory musia mať jedinečný názov v rámci jedného Maildiru a premenovanie súborov pri presune z new do cur.

3 Návrh a implementácia

Program je implementovaný v jazyku C/C++ s použitím povolených knižníc potrebných pre vypracovanie zadania. Je rozdelený do dvoch modulov. Nie je použitý objektový návrh ale je vytvorená jedna trieda ktorá reprezentuje jeden modul. Druhý modul obsahuje hlavný program s funkciou `main()` a s ostatnými potrebnými funkciami.

3.1 Spracovanie argumentov

Pre spracovanie argumentov bola vytvorená trieda `Arguments`, ktorá obsahuje privátne premenné pre každý prepínač a jednu metódu na kontrolu argumentov. Na začiatku sa kontroluje či bol zadaný prepínač `-h`. Ďalej sa kontroluje či bol program spustený iba so samotným prepínačom `-r`. Na koniec sa povinné a voliteľné parametre kontrolujú pomocou funkcie `getopt` ktorá vyhodnotí správnosť parametrov a ich hodnôt. Na začiatku funkcie `main` sa vytvorí objekt typu `Arguments` a zavolá sa metóda na spracovanie argumentov.

3.2 Pripojenie klientov

Sieťová komunikácia a pripojenie klientov je riešená pomocou BSD socketov. Keďže sa jedná o server ktorý je paralelný, čiže paralelne vybavuje každé pripojenie boli použité vlákna z knižnice `pthread.h`. Keď sa jeden klient pripojí akceptuje sa pripojenie, vytvorí sa socket cez ktorý komunikuje a ďalej jeho požiadavky sú spracované v rámci vlákna. Túto časť popisuje ďalšia kapitola. Sockety sú nastavené ako neblokujúce.

3.3 Spracovanie príkazov

Táto časť implementácie je najrozsiahlejšia. Pre spracovanie príkazov pre každého klienta je vytvorené vlákno kde sa daný klient obsluhuje. Hlavnú časť vlákna tvorí smyčka kvôli použitiu nebolkojúcich socketov v ktorej je volaná funkcia `recv`, kontrolujú sa jej návratové hodnoty, prípadne či by blokovala. Po úspešnom načítaní dát do bufferu nasleduje spracovanie príkazov. Stavový automat ktorý je vidno na obrázku 1 je implementovaný pomocou jazykovej konštrukcie `switch`, kde každý `case` reprezentuje jeden stav. C++ neumožňuje mať reťazce vo vetvách `case`, preto bola vytvorená enumerácia a pomocná funkcia, ktorá mapuje reťazce na odpovedajúce hodnoty enumerácie. Podobné riešenie bolo použité pre spracovanie príkazov, ktoré sú spracované vo vstavoch. Výsledkom je dvojúrovňový `switch`, kde 1. úroveň udáva stav a 2. príkaz.

Ak sa príkaz spracoval a bol podporovaný nasleduje spracovanie prípadných argumentov. Pri príkazoch ktoré nemajú argument je možné zadať argument ale ten sa ignoruje. V prípadoch kde sú požadované argumenty tak sú povinné, ak nejaký chýba alebo je ich viac tak to server vyhodnotí ako chybu. Argumenty sa rozdeľujú podľa medzier, výnimkou je príkaz `PASS` kde heslo môže obsahovať medzeru takže vlastne vždy bude mať iba jeden argument. Ak sa nepošle žiadny príkaz alebo poslaná správa nekončí dvojicou `\r\n` tak server pošle zápornú odpoveď klientovi.

Pre generovanie hashu pri príkaze `APOP` bola použitá funkcia `MD5` z knižnice `openssl/md5.h`. Táto funkcia bola tiež použitá pre vytvorenie unikátneho identifikátoru pre súbory kde sa hashuje konkaténácia aktuálneho času, názvu súboru a globálna číselná premenná ktorá sa inkrementuje pri každom generovaní tohto identifikátoru.

Po úspešnej autentifikácii klienta je prevedená kontrola správnosti adresárovej štruktúry, ak je chybná tak server odpojí aktuálneho klienta. Tiež sa vykoná pokus o uzamknutie `Maildiru`. Pri tejto implementácii ak sa nepodarí pričínok uzamknúť tak klient je odpojený. Na uzamknutie sa používa globálna premenná typu `pthread_mutex_t`.

Veľkosť súborov je vypočítaná pri presune z `new` do `cur`. Kvôli rôznej reprezentácii ukončovania riadku na odlišných systémoch bola potreba kontrolovať tento znak. V prípade ak riadok končí iba `\n` tak k celkovej veľkosti je prirátaný ešte jeden bajt, keďže protokol `POP3` udáva že každý riadok má byť ukončený `\r\n`. Výsledná veľkosť súboru sa teda skladá z fyzickej veľkosti plus počet riadkov, ktoré boli ukončené iba s `\n`.

Pri akejkol'vek internej chybe na serveri je vypísaná chybová hláška, klient je odpojený a vlákno sa skončí.

3.4 Ukončenie signálom SIGINT

Na zachytenie signálu je použitá funkcia `signal` z knižnice `<csignal>`. Pri zachytení signálu je nastavená globálna premenná, ktorá sa kontroluje v podmienkách cyklov pre prijímanie pripojení vo funkcii `main` a pre komunikáciu s klientom vo funkcii pre vlákno. Táto kontrola je možná kvôli použitiu neblokujúcich socketov, kde sa cyklí v nekonečnej smyčke. Vlákno ukončí socket na ktorom komunikuje a vhodne sa ukončí. Funkcia `main` čaká na ukončenie každého vlákna. Táto časť je riešená pomocou globálnej premennej, kde sa ukladá aktuálny počet vlákien. Ak sa vytvorí vlákno tak premennú inkrementuje, pri ukončení dekrementuje. Funkcia `main` sleduje túto premennú a ak už nie je aktívne žiadne vlákno tak zatvorí pasívny socket (použitý pre `listen`) a skončí program.

3.5 Pomocné súbory, reset

Kvôli príkazom `RESET`, `UID1` a kvôli tomu že obsah súboru okrem príkazov `RETR` a `TOP` môžeme čítať iba raz boli zavedené pomocné súbory. Jeden súbor s názvom *log.txt* obsahuje potrebné informácie o súbore vo formáte `názov/unikátny_identifikátor/veľkosť_súboru`. Ako oddelovač bolo použité lomítko, ktoré nie je povolené v názve súboroch na UNIXových systémoch takže jednoznačne sa dá určiť kde jedna informácia končí a druhá začína. Súbor obsahuje pre každý e-mail jeden riadok.

Do druhého súboru sú ukladané absolútne cesty všetkých e-mailov ktoré boli presunuté z `new` do `cur`. Pri zadaní prepínaču `-r(reset)` všetky súbory, ktoré obsahuje súbor `reset` a sú fyzicky dostupné na disku sú presunuté z `cur` do `new`. Na konci sa všetky pomocné súbory vymažú.

4 Použitie programu

Program má tri režimy behu:

1. `./popser -h`: Vypíše sa nápoveda, program sa správne ukončí.
2. `./popser -r`: Reset, `Maildir` sa vráti do stavu, ako keby server nebol nikdy spustený, program sa správne ukončí.
3. `./popser -p číslo_portu -a autentifikačný_súbor -d cesta_k_maildiru [-r] [-c]`: Server sa spustí, komunikuje na zadanom porte, pracuje so zadaným `Maildirom`, načíta užívateľské meno a heslo zo súboru `autentifikačný_súbor`. Pri zadaní parametru `-c` sa povolí autentifikácia typu `USER/PASS`, inak je povolená autentifikácia pomocou príkazu `APOP`. Parameter `-r` má rovnaký význam ako v príklade č. 2, čiže resetne servern pričom teraz sa neukončí ale beží ďalej.

5 Záver

Program bol spočiatku vyvíjaný na Ubuntu 16.04.03 LTS, neskôr však radšej som to presunul na školský server `Merlin` kvôli tomu že tu sa bude testovať aj odovzdaná verzia. Program je preložený prekladačom `g++` pomocou `Makefile`, ktorý je v odovzdaných súboroch.

Literatúra

- [1] Bernstein, D. J.: *Using maildir format*. [Online].
URL <http://cr.yp.to/proto/maildir.html>
- [2] Myers, J.; Mellon, C.; Rose, M.: *Post Office Protocol - Version 3*. 1996, [Online].
URL <http://www.ietf.org/rfc/rfc1939.txt>
- [3] P. Resnick, E.: *Internet Message Format*. 2008, [Online].
URL <http://tools.ietf.org/html/rfc5322>