VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

POP3 SERVER
POP3 SERVER

SÍŤOVÉ APLIKACE A SPRÁVA SÍTÍ
NETWORK APPLICATIONS AND NETWORK ADMINISTRATION

AUTOR **JAKUB HANDZUŠ**
AUTHOR

BRNO 2017

Obsah

Kapitola 1

Úvod

Dokumentácia popisuje riešenie projektu pre predmet Sieťové aplikácie a správa sietí. Našou úlohou bolo implementovať konkurentný POP3 server v jazyku C/C++.

Dokumentácia je rozdelená do niekoľkých kapitol. Kapitola 2 obsahuje stručný úvod do problematiky POP3 protokolu a zároveň objasňuje adresárovú štruktúru Maildir. Kapitola 3 je venovaná popisu návrhu samotnej aplikácie, ktorý je z implementačného hľadiska podrobnejšie špecifikovaný v kapitole 4. Kapitola 5 je zameraná na testovanie aplikácie. Posledné dve kapitoly (6, 7) sú zamerané na popis základných informácií o aplikácií a návod na jej použitie.

Kapitola 2

Problematika

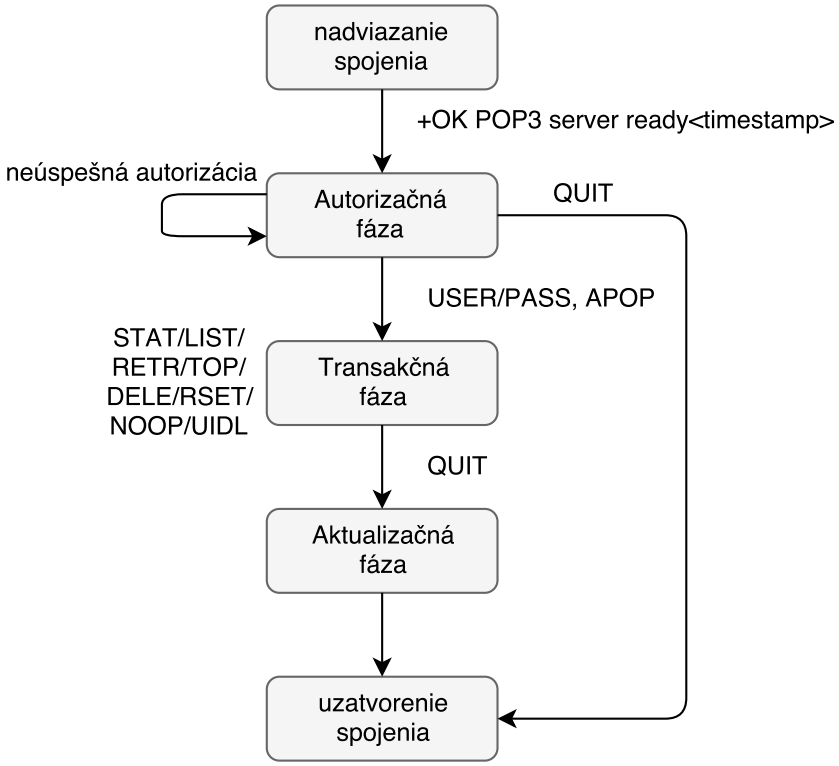
V tejto kapitole bude stručne objasnená problematika POP3 protokolu a adresárová štruktúra Maildiru.

2.1 POP3 (Post Office Protocol 3)

Post Office Protocol verzie 3 je internetový protokol definovaný v RFC 1939 [2] z roku 1996. Jedná sa o protokol aplikačnej vrstvy, ktorý sa využíva na prijímanie elektronickej pošty zo vzdialeného servera na lokálny počítač prostredníctvom TCP spojenia. Protokol definuje pravidlá, podľa ktorých má byť riadený prenos elektronickej pošty. Dokáže pracovať iba s jedinou schránkou na strane servera, ktorú môže zároveň využívať iba jeden klient. Z tohto dôvodu, po pripojení a autentifikácií klienta, umiestni POP3 server na túto schránku zámok, pričom všetky zmeny sú vykonávané až v poslednej fáze, pred korektným ukončením spojenia.

Činnosť protokolu je popísaná stavovým automatom na obrázku 2.1. Po pripojení klienta sa server dostane do autorizačnej fázy, kedy čaká na autorizovanie klienta. Autorizácia nastáva buď nešifrovane, pomocou `USER` a následne `PASS` príkazu, alebo šifrovane, pomocou `APOP` príkazu, ktorý na zabezpečenie využíva MD5 hash. Existujú aj riešenia POP3 protokolu podporujúce bezpečnejšie šifrovanie pomocou SSL, prípadne TSL, keďže v dnešnej dobe sa MD5 nepovažuje za bezpečné. V takomto prípade hovoríme o protokole POP3S.

Po úspešnom prihlásení sa server dostane do transakčného stavu, kedy vykonáva príkazy nad schránkou. Zoznam podporovaných príkazov je uvedený v tabuľke 2.1. Každá klientska správa obsahujúca príkaz a argumenty musí byť ukončená `CRLF` párom. V prípade ukončenia komunikácie príkazom `QUIT` server prechádza do poslednej (aktualizačnej) fázy, kde sú vykonávané všetky klientske zmeny a odstránené správy označené na vymazanie. V prípade nečakaného ukončenia spojenia, zmeny nebudú vykonané.



Obr. 2.1: Chovanie protokolu POP3

Príkaz	**Význam**
USER	meno identifikujúce poštovú schránku
PASS	heslo pre prístup ku schránke
APOP	bezpečné prihlásenie ku schránke pomocou MD5
STAT	počet správ v schránke + ich veľkosť
LIST	zoznam správ v schránke a ich veľkosť
RETR	server pošle požadovanú správu klientovi
DELE	nastaví u správy príznak "deleted"
RSET	zmaže príznak "deleted"pri správach označených na zmazanie
NOOP	prázdny príkaz
TOP	server pošle klientovi hlavičku a zadaný počet riadkov správy
UIDL	server vypíše jednoznačný identifikátor správy
QUIT	uzatvorí spojenie, zmaže správy označené "deleted"

Tabuľka 2.1: Prehlad príkazov protokolu POP3

2.2 Adresárová štruktúra Maildir

Aplikácia pracuje s adresárovou štruktúrou Maildir [1], v ktorej sú uložené všetky mailové súbory. Každá správa je uložená ako jeden samostatný súbor. Maildir adresár, väčšinou pomenovaný ako *"Maildir"*, sa skladá z troch podadresárov pomenovaných *"tmp"*, *"new"* a *"cur"*.

Program, ktorý prijíma mailové správy, zapisuje každú správu do adresára tmp. Po prijatí celej správy jej pridelí jedinečné meno a súbor premiestni do adresára *"new"*. Aplikácia POP3 serveru tak pracuje následne až s týmto adresárom, kedy po úspešnej autorizácii klienta, aplikácia premiesti všetky nové správy do adresára *"new"*. Presúvanie musí byť zabezpečené pomocou atomickej funkcie `rename()`, aby bola zaručená konzistencia celého adresára.

Adresárová štruktúra pracuje so súbormi typu *"Internet Message Format"*, skrátene *".imf"*, definovaným v RFC 5322 [3] . Tento súbor pozostáva z hlavičky, kde sa nachádzajú informácie o odosielateľovi, príjemcovi, predmete, času odoslania atď., za ktorými nasleduje prázdny riadok a obsah správy. Pri odosielaní je nutné zaistiť, aby boli riadky v maili ukončené pomocou `CRLF`. Toto platí aj v prípade, ak by sa v adresári Maildir nachádzali aj súbory s riadkami ukončenými iným spôsobom, čo kapitola 1.1 dokumentu [3] dovoľuje.

Kapitola 3

Návrh aplikácie

Aplikácia začína načítaním vstupných argumentov príkazového riadku, ich overením a priradením príslušných hodnôt do štruktúry. V prípade argumentu `-r`, vykoná aplikácia reset. Nasleduje prečítanie obsahu autentifikačného súboru. V prípade úspešnosti všetkých vyššie spomenutých úkonov, program vytvorí neblokujúci TCP soket, na ktorý sa následne pripájajú klienti. Pri pripojení klienta na server sa vytvorí nové vlákno pre konkrétneho klienta, ktoré je aktívne, až kým sa spojenie s daným klientom neukončí. Spojenie s klientom je možné rozčleniť do troch fáz. V prvej fáze prebieha autorizovanie klienta, ktoré v prípade úspechu prechádza do fázy druhej, v ktorej môže klient odosielať rozličné POP3 príkazy. V prípade kladnej odpovede server odošle správu začínajúcu `+OK` a v prípade zápornej odpovede `-ERR`. Server prechádza z druhej do tretej fázy pri zadaní príkazu `QUIT`, kedy sú vymazané všetky klientom označené súbory. Server sa ukončí po obdržaní signálu `SIGINT`.

Kapitola 4

Popis implementácie

Aplikácia je implementovaná v jazyku C/C++. Návrh aplikácie síce nie je objektový, ale pracujeme s niektorými objektami zo štandardných knižníc jazyka C++. Zdrojový kód je rozdelený do siedmych modulov a dvoch hlavičkových súborov. Samotná funkcia `main()` sa nachádza v súbore *popser.cpp*. Spracovanie argumentov získaných z príkazového riadku je realizované v module *arguments.cpp*. Modul *net.cpp* zabezpečuje inicializáciu soketov, vytváranie vlákien, odosielanie a prijímanie správ. Taktiež zaisťuje odchytávanie signálu `SIGINT` a bezpečné ukončenie celej aplikácie. Hlavná logika sa nachádza v module *fsm.cpp*, kde je implementovaný konečný automat prijímajúci klientske príkazy. Samotná implementácia vykonávaných príkazov sa nachádza v module *commands.cpp*. Všetká práca so súbormi je implementovaná v module *file_master.cpp*. Posledný modul využívaný aplikáciou je *md5.cpp*. Tento modul bol prevzatý zo stránky zedwood.com[1] a stará sa o vytvorenie MD5 hashu.

4.1 Spracovanie vstupných argumentov

Aplikácia má podľa zadania za úlohu pracovať s niekoľkými vstupnými argumentami, z toho tri sú jednoduché prepínače a tri sú argumenty s hodnotou. Každý z prepínačov môže byť zadaný maximálne raz. Aplikácia má tri režimy behu, ktoré budú bližšie popísané v kapitole 7. Spracovanie vstupných argumentov a ich ukladanie do odpovedajúcej štruktúry je realizované pomocou vlastnej funkcie `parseArg()`, ktorá je založená na funkcii `getopt()`. Vzhľadom na jej pokročilú funkcionalitu je možné zadávať argumenty v rôznom poradí. V prípade akejkoľvek chyby je aplikácia ukončená chybovým kódom `E_ARG`. Pokiaľ nenastala žiadna chyba, tok je prepnutý naspäť do funkcie `main()`.

4.2 Inicializácia soketu

Aplikácia komunikuje výhradne pomocou neblokujúcich soketov, na ktoré bolo potrebné využiť funkciu `select()`. Nastavenie neblokujúceho príznaku sa aplikuje ako na uvítací soket, tak aj na komunikujúci pomocou funkcie `fcntl()`. Aplikácia nastavuje soket príznakom `SO_REUSEADDR`, aby sa v prípade ukončenia aplikácie mohla okamžite pripojiť znovu na rovnaký port.

[1] http://www.zedwood.com/article/cpp-md5-function

4.3 Obsluha klienta

Po pripojení klienta na server je vygenerovaná časová pečiatka v podobe `<process-ID.clock@hostname>`. Táto pečiatka je spolu s popisovačom soketu uložená do globálneho asociatívneho kontainera `glob_fd_map`, ktorý je typu `std::map`. Následne program vytvorí nové vlákno a zavolá funkciu `pop3_fsm()`, do ktorej sa pomocou ukazateľa odošlú informácie o zadaných argumentoch programu a prihlasovacie údaje klienta. Pri úspešnom vytvorení vlákna uloží hlavné vlákno do globálneho asociatívneho kontainera `glob_thread_map` (taktiež `std::map`) pár kľúč-hodnota, pozostávajuci z identifikátoru vlákna a popisovača soketu. Každý novovzniknutý proces si zistí z globálnych kontainerov svoj popisovač soketu a časovú pečiatku, ktorú bude ďalej používať. Pri tomto zisťovaní musí byť použitý mutex, aby bolo zaručené, že hlavné vlákno stihne zapísať do týchto kontainerov skôr, než z nej bude čítať novovytvorené vlákno.

Vo funkcii `pop3_fsm()` je implementovaný konečný automat prijímajúci príkazy POP3 protokolu popísaného v kapitole 2.1. Pred každým opakovaním cyklu je volaná funkcia `cmd_recognition()`, ktorá overí správnosť a rozpozná zadaný príkaz spolu z argumentami z klientskej správy. Príkazy nie sú citlivé na malé a veľké písmená, a preto je možné ich zadávať oboma spôsobmi, prípadne aj kombinovane.

4.3.1 Autorizačná fáza

V prvej, autorizačnej, fáze môže klient zadať iba príkaz `APOP` alebo `USER` a následne `PASS`. V jednom momente je možné používať iba jeden typ prihlasovania, ktorý závisí od použitia prepínača `-c`. V prípade `APOP` autentifikácie, server musí zašifrovať časovú pečiatku spolu s heslom a porovnať ich.

Po úspešnej autorizácii sa server pokúsi získať výlučný prístup do schránky, ktorý je implementovaný pomocou mutexu. Ak žiadne iné vlákno nemá získaný výlučný prístup k adresáru, daný proces si ho zamkne a premiestni si všetky súbory z podadresára *"new"* do adresára *"cur"*. Pri presune sa celý súbor prečíta a zistí sa jeho veľkosť, ktorá je následne spolu s jeho názvom uložená do pomocného súboru. Na zistenie veľkosti nie je možné použiť funkciu `tellg()`, pretože veľkosť znaku konca riadku musí byť vždy započítaná ako 2 bajty, hoci riadok môže byť ukončený aj jediným znakom. Dôvod k tomuto opatreniu bol bližšie špecifikovaný v kapitole 2.

V dalšom kroku sa vykoná rozbor súborov nad podadresárom *"cur"* funkciou `check_mailbox()`. Táto funkcia uloží do sekvenčného kontainera typu `std::vector` štruktúry `t_file`, v ktorých sa nachádzajú informácie o poradovom čísle, mene, veľkosti a príznaku vymazania pre každý súbor. Veľkosť je prečítaná z pomocného súboru pomocou funkcie `size_map()`, ktorá vráti asociatívny kontainer pozostávajúci zo všetkých uložených súborov a ich veľkostí. Týmto sa zabraňuje viacnásobnému čítaniu súborov s cieľom zistenia veľkosti. Klient je následne oboznámený o počte a veľkosti všetkých mailov a automat prechádza do ďalšieho stavu.

4.3.2 Transakčná fáza

Druhá, teda transakčná, fáza prijíma všetky príkazy definované RFC 1939 [2] vrátane `TOP` a `UIDL`. Niektoré zložitejšie príkazy sú implementované v module `commands.cpp`. V prípade, že príkaz zadaný klientom nie je korektný, server odpovie `-ERR invalid command`. V následujúcich odrážkach budú spomenuté niektoré príkazy zaujímavé na implementáciu.

- Príkaz `DELE` slúži na označenie mailu, ktorý bude vymazaný v tretej fáze. V implementácii sa jedná iba o označenie mailu príznakom `to_be_del`.
- Príkaz `LIST` môže byť zadaný buď bez, alebo s parametrom. V implementácií je striktne dané, aby príkaz `LIST` bez parametra neobsahoval za sekvenciou štyroch znakov žiadnu medzeru alebo iný znak. V opačnom prípade bude vyhodnotený ako príkaz s parametrom, nad ktorým budú vykonávané overenia argumentu. To platí taktiež aj pre ostatné príkazy s argumentom.
- Pri príkaze `RETR` bolo potrebné zaručiť, aby každý odoslaný riadok mailu bol ukončený znakom `CRLF`. Z tohto dôvodu je využitá funkcia `getline()`, ktorá číta obsah vždy do konca riadku. Podľa posledného znaku reťazca vráteného touto funkciou sa program rozhodne, či pridá na koniec práve analyzovaného riadku `CRLF` pár, alebo iba `LF`.
- Server obsahuje implementáciu príkazu `TOP`, ktorý bol rozšírením projektu. Implementácia sa nachádza vo funkcii `cmd_top()`. Táto funkcia sa spolieha na korektnosť mailových súborov v imf formáte, teda počíta s prázdnym riadkom medzi hlavičkou a obsahom správy. Samotný príkaz sa skladá z troch písmen, pričom ako jediný má dva povinné argumenty, teda bolo potrebné preň vytvoriť mierne odlišné spracovanie.
- Bezparametrový príkaz `UIDL` je implementovaný vo funkcii `cmd_uidl`, ktorá sa spolieha na jedinečnosť názvov súborov už pred prijatím mailov do schránky, preto tento príkaz užívateľovi zobrazí iba názov súboru.

4.3.3 Aktualizačná fáza

Konečný automat sa dostáva do tretej fázy pri obdržaní príkazu `QUIT`. V tejto fáze vymaže všetky označené súbory a odomkne výlučný prístup do zložky. Ďalej odošle správu o počte mailov, ktoré zostali v schránke a odstráni z globálnych kontainerov svoje dáta o popisovači soketu a identifikátore vlákna. Po vykonaní všetkých úkonov automat ukončuje vlákno.

4.4 Komunikácia s klientom

Prijímanie a odosielanie správ je implementované vo funkcii `send_and_recv()`. Táto funkcia je volaná pri každej odpovedi servera, keďže sa očakáva, že po každej takejto odpovedi sa bude klient dopytovať servera znovu. Ako bolo spomenuté v kapitole 2, každá správa od klienta musí byť ukončená `CRLF` párom.

4.5 Ukončenie servera

Server sa po obdržaní signálu `SIGINT` bezpečne ukončí. Táto funkcionalita je implementovaná vo funkcii `signal_handler()`, ktorá signál zachytí a zaháji ukončovanie. Ako prvé sú ukončené všetky aktívne vlákna vytvorené pre obsluhu klientov, pričom informácia o aktívnych vláknach sa nachádza v globálnom kontaineri `glob_thread_map`. Ďalším krokom je ukončenie všetkých otvorených spojení s klientami, ktoré sú zaznamenané v kontaineri `glob_fd_map`. Nakoniec, v prípade, že by nejaké vlákna mali pred ukončením otvorené súbory, sú tieto súbory korektne zatvorené. Po vykonaní všetkých úkonov je ukončený samotný beh programu.

Kapitola 5

Ladenie a testovanie

Vývoj aplikácie prebiehal na serveri *merlin.fit.vutbr.cz* pod operačným systémom CentOS 6.5 a na osobnom počítači so systémom Ubuntu 16.04. Pre ladenie aplikácie bol využívaný nástroj telnet, ktorý pripojením na náš server simuloval klientsku stranu. Pre odchytávanie komunikácie medzi klientom a serverom bol využitý program Wireshark.

Vo finálnej fáze bola aplikácia testovaná rôznymi spôsobmi. V prvom prípade boli porovnávane výstupy so serverom Dovecot. V druhom prípade bol použitý testovací skript vytvorený kolegom Petrom Rusiňákom. V tretom prípade bola naša aplikácia využívaná klientskou aplikáciou kolegu Petra Drahovského, ktorá maily sťahovala.

Všetky testovania prebiehali na sade reálnych emailov od veľkosti niekoľko bajtov po niekoľko megabajtov.

Kapitola 6

Informácie o programe

Aplikácia pozostáva zo súboru Makefile, ktorý slúži na zostavenie programu a nasledovných zdrojových súborov:

- `popser.cpp`
- `popser.h`
- `fsm.cpp`
- `net.cpp`
- `arguments.cpp`
- `commands.cpp`
- `file_master.cpp`
- `md5.cpp`
- `md5.h`

Aplikácia pre svoj beh využíva funkcie, štruktúry či konštanty z následujúcich knižníc:

- C POSIX:
  - `string.h`
  - `unistd.h`
  - `netdb.h`
  - `arpa/inet.h`
  - `pthread.h`
  - `fcntl.h`
  - `dirent.h`
  - `sys/stat.h`
  - `stdio.h`
  - `stdlib.h`
- C++ standart:
  - `iostream`
  - `vector`
  - `map`
  - `mutex`
  - `csignal`
  - `fstream`

Kapitola 7

Návod na použitie

Po preložení aplikácie pomocou príkazu `make` sa vygeneruje spustiteľný súbor nazvaný `popser`, ktorý podporuje 3 režimy behu.

1. Režim nápovedy, kedy je pri zadanom parametri `-h` vypísaná nápoveda a program sa ukončí.
2. Režim reset, kedy je pri zadanom jedinom parametri `-r` vykonaný reset, a program následne ukončený.
3. Bežný režim, ktorý vyžaduje tri povinné parametre a to:
   - `-a` spolu s cestou k autentifikačnému súboru
   - `-p` spolu s číslom portu
   - `-d` spolu s cestou do zložky *Maildir*

   Ďalej je možne v tomto režime použiť 2 voliteľné parametre a to:
   - `-c`, po ktorom server akceptuje autentizačnú metódu, ktorá prenáša heslo v nešifrovanej podobe
   - `-r`, kedy je pred spustením servera vykonaný reset, ktorý pozostáva z vymazania pomocných súborov a premiestnenia mailov z adresára */cur* do */new*

Príklad použitia:

```
./popser -a auth.txt -c -p 4242 -d Maildir -r
```

Pri akejkoľvek chybe je na štandardný chybový výstup vypísaná hláška špecifikujúca chybu a program je ukončený príslušnou hodnotou. V prípade vynúteného ukončenia aplikácie je nutné odoslať signál `SIGINT`.

Kapitola 8

Záver

Cieľom tejto práce bolo vytvoriť konzolovú sieťovú aplikáciu. Pri implementácií bolo najprv potrebné naštudovať základne princípy fungovania POP3 protokolu na strane konkurentného servera a prácu s adresárovou štruktúrou Maildir. Projekt zároveň poukázal na dôležitosť schopnosti efektívne vyhľadávať informácie v súboroch RFC a overil programovacie schopnosti v jazyku C++, ako prácu s BSD schránkami, multivláknové programovanie a mnoho ďalších. Nadobudnuté vedomosti z oblasti tejto problematiky sú nesmierne dôležité pre ďalší kariérny vývoj.

Literatúra

[1] *maildir(5) - directory for incoming mail messages.*
URL http://www.qmail.org/qmail-manual-html/man5/maildir.html

[2] Myers, J.; Mellon, C.; Rose, M.: Post Office Protocol - Version 3. RFC 1939, RFC Editor, May 1996.
URL http://www.rfc-editor.org/rfc/rfc1939.txt

[3] Resnick, P.: Internet Message Format. RFC 5322, RFC Editor, October 2008.
URL http://www.rfc-editor.org/rfc/rfc5322.txt