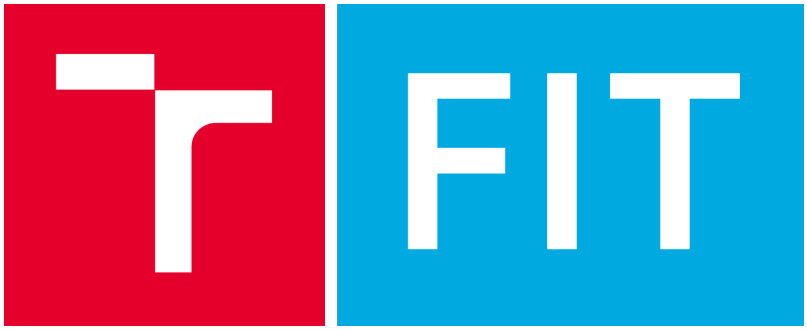
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ



Síťové aplikace a správa sítí
Dokumentace projektu

POP3 server

Vedoucí projektu: Ing. Martin Holkovič, iholkovic@fit.vutbr.cz

20. listopadu 2017 Dominik Skála, xskala11@stud.fit.vutbr.cz

Obsah

1 Zadání

Vytvořte komunikující aplikaci podle konkrétní vybrané specifikace pomocí síťové knihovny BSD sockets (pokud není ve variantě zadání uvedeno jinak). Projekt bude vypracován v jazyce C/C++. Pokud individuální zadání nespecifikuje vlastní referenční systém, musí být projekt přeložitelný a spustitelný na serveru merlin.fit.vutbr.cz.

1.1 Varianta POP3 server

Vytvořte program popser, který bude plnit úlohu POP3 [2] (dále pouze server). Na server se budou připojovat klienti, kteři si pomocí protokolu POP3 stahují data ze serveru. Server bude pracovat s e-maily uloženými ve formátu IMF [3] v adresářové struktuře typu Maildir [1].

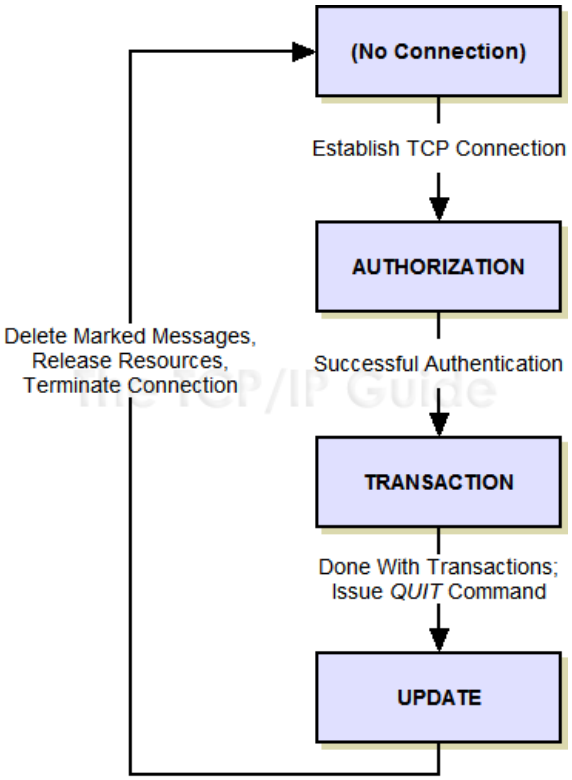
Vašim cílem je tedy vytvořit program, který zvládne pracovat s e-maily uloženými v adresářové struktuře Maildir. Program bude pracovat pouze s jednou poštovní schránkou. Takto uložené e-maily budou prostřednictvím protokolu POP3 poskytované jedinému uživateli.

2 Uvedení do problematiky

2.1 POP3 server

2.1.1 Serverové stavy

Server dosahuje 4 stavů, mezi kterými postupně iteruje. Server dosahuje stavů: Čekání na připojení, Autorizace, Transakce, Aktualizace.



Čekání na připojení Server čeká na připojení klienta

Autorizace Po započetí spojení server zašle klientovi uvítací zprávu a klient se musí autorizovat

Transakce Po úspěšné autoriazci klient provádí operace nad e-maily, po dokončení práce se odpojí příkazem QUIT

Aktualizace Po odpojení klienta dojde ke smazání všech e-mailů, které byly označeny ke smazání, ukončí se spojení a server čeká na dalšího klienta.

2.1.2 Serverové odpovědi

Server má dva stavy, kterými musí klientovi odpovídat, není přípustné, aby odpověděl jinou zprávou. Kladnou zprávou je "+OK", zápornou je "-ERR". Každá zpráva je tedy ve formátu: [+OK|-ERR] TEXT, kde TEXT je informační text související s operací (2.1.4), která byla serveru zaslána. Může obsahovat dodatečné informace jako např. uvítací zprávu, počet e-mailů, velikost, atd.

+OK Příkaz přijatý serverem byl bez problému zpracován a je zasílána kladné odpověď

-ERR Příkaz přijatý serverem byl s nějakou chybou zpracován špatně.

2.1.3 Nonce

Server po připojení klienta vygeneruje speciální nonci, která je zaslána klientovi jako součást uvítací zprávy. Samotná nonce je ve tvaru: <process-ID.clock@hostname>, kde:

process-ID je id systémového procesu

clock je čas, který uběhl od 1.1.1970 v sekundách

hostname je hostname zařízení na kterém server běží

2.1.4 E-mailové operace

Server rozumí následujícím příkazům: APOP, USER, PASS, NOOP, QUIT, STAT, LIST, RETR, DELE, RSET, UIDL a TOP. Jejich chování je předem definované [2], Pokud jsou veškeré příkazy zpracovány správně, server reaguje kladnou odpovědi (2.1.2), pokud došlo při zpracovávání k chybě, vrací zápornou odpověď. Následuje vysvětlení jednotlivých příkazů, za pomlčkou je jejich ukázkové použití. Pokud je parametr v hranatých závorkách, je parametr nepovinný a příkaz má tedy dvě varianty.

APOP - `APOP username nonce` Příkaz je serverem zpracován, může být použit pouze v autorizačním stavu za předpokladu, že není aktivní jiná forma autorizace, než APOP (2.1.5). Tento příkaz má vždy dva argumenty, uživatelské jméno a hashované heslo. Server na začátku komunikace pošle uvítací zprávu, která obsahuje speciální nonci, kterou klient použije, konkatenuje k ní heslo. Celý tento obsah hashuje přes md5 algoritmus a následně pošle serveru zpět jako heslo. Pokud hash sedí, klient je autorizován.

USER - `USER username` Příkaz je serverem zpracován, může být použit pouze v autorizačním stavu za předpokladu, že není aktivní jiná forma autorizace, než USER/PASS (2.1.5). Existuje pouze jediný argument, tím je uživatelské jméno. Server zkontroluje validitu uživatele, poté čeká na ověření hesla příkazem PASS. Server provádí kontrolu jména z argumentu vůči souboru s autorizačními údaji (3.4.2).

PASS - `PASS password` Příkaz je serverem zpracován, může být použit pouze v autorizačním stavu za předpokladu, že není aktivní jiná forma autorizace, než USER/PASS (2.1.5). Existuje pouze jediný argukent, tím je uživatelské heslo. Příkaz může být použit pouze po kladné odpovědi z příkazu USER. Server provádí kontrolu hesla z argumentu vůči souboru s autorizačními údaji (3.4.2).

NOOP - `NOOP` Příkaz je serverem zpracován,neobsahuje žádný argument. Může být použit pouze v transakčním stavu, reálně však pouze vrací kladnou odpověď, neprovádí žádné operace ani úpravy.

QUIT - `QUIT` Příkaz je serverem zpracován dvěma různými možnostmi a může být použit pouze v transakčním nebo autorizačním stavu, Neobsahuje žádný argument. Pokud je QUIT příkaz použit ve stavu Autorizace (2.1.1), dojde k odpojení klienta a návratu do stavu čekání na klienta. Pokud je příkaz QUIT použit ve stavu Transakce, musí server přejít do stavu Aktualizace, ve kterém provede smazání všech e-mailů, které byly označení ke smazání a odpojí klienta, případně uvolní veškeré alokované prostředky související s daným klientem.

STAT - `STAT` Příkaz je serverem zpracován může být použit pouze v transakčním stavu, server vrací počet zpráv, které nebyly označeny ke smazání a sumu (2.2.2) velikostí těchto e-mailů. Příkaz neobsahuje žádný argument.

LIST - `LIST` [`id`] Příkaz je serverem zpracován dvěma různými možnostmi a může být použit pouze v transakčním stavu,. Pokud je u operace zadán nenulový a kladný číselný argument a pokud není. Pokud byl zadán, server vyhledá zprávu, pokud ji nalezl, zkontroluje zda nebyla označena ke smazání. Pokud nebyla, vrátí index [4] zprávy a její velikost (2.2.2). Pokud nebyl zadán žádný argument, server zašle postupně jednotlivé indexy zpráv s jejich velikostmi. Po kompletním odeslání všech zpráv zašle ukončovací zprávu: ".".

RETR - `RETR id` Příkaz je serverem zpracován, může být použit pouze v transakčním stavu. Vždy musí obsahovat jeden nenulový, kladný, číselný argument. Dle tohoto argumentu vyhledá zprávu, kterou je nutné zaslat, zkontroluje zda nebyla označena ke smazání. Pokud nebyla označena ke smazání, zašle server index zprávy, její velikost. K tomu pošle obsah e-mailu i s jeho hlavičkou a ukončovací zprávu: ".".

DELE - `DELE id` Příkaz je serverem zpracován, vždy musí obsahovat jeden nenulový, kladný, číselný argument, na základě kterého vyhledá index zprávy. Pokud již zpráva není označena ke smazání, dojde k označení zprávy ke smazání. Zpráva stále zůstává v e-mailovém adresáři dokud server nepřejde to stavu Aktualizace (2.1.1).

RSET - `RSET` Příkaz je serverem zpracován, neobsahuje žádný argument. Může být použit pouze v transakčním stavu. Po jeho použití dojde k odznačení všech e-mailů, které byly označeny ke smazání. Nebudou tedy smazány.

UIDL - `UIDL` [`id`] Příkaz je serverem zpracován dvěma různými možnostmi a může být použit pouze v transakčním stavu,. Pokud je u operace zadán nenulový a kladný číselný argument a pokud není. Pokud byl zadán, server vyhledá zprávu, pokud ji nalezl, zkontroluje zda nebyla označena ke smazání. Pokud nebyla, vrátí index [4] zprávy a její unikátní hash, který je tvořen z celé složky k e-mailovému adresáři, podsložky "/cur/", názvu e-mailu a speciální nonce, která je tvořena po připojení klienta k serveru. Pokud nebyl zadán argument, dojde k odeslání počtu zpráv, následně se zašle id každá zprávy a unikátní hash pro každou z nich, po všech zprávách se zašle ukončovací zpráva ".".

TOP - `TOP id n` Příkaz je serverem zpracován, může být použit pouze v transakčním stavu. Vždy musí obsahovat dva nenulové, kladné, číselné argumenty. První argument určuje index zprávy, druhý určuje počet řádků, které se ze zprávy zašlou. Dle indexu je zpráva vyhledána, zkontrolována, zda není označena ke smazání a je zaslána hlavička a tolik řádků ze samotné zprávy, kolik bylo vyžádáno v druhém argumentu.

2.1.5 Formy autorizace

Server umožňuje dvě formy autorizace, užitím příkazu APOP nebo kombinací příkazů USER/PASS.

APOP Během této formy autorizace je užita nonce (2.1.3) kterou server zašle příchozímu klientovi, na základě které klient zašle serveru zašifrované heslo ve formátu: nonce.pass ("."je symbolem konkatenace řetězců). Pro šifrování je využito md5 algoritmu. Celý příkaz který klient zasílá je ve formátu: APOP username nonce.pass, kde username je uživatelské jméno a pass je heslo z autorizačního souboru (3.4.2).

Ukázka komunikace (S = server, C = klient):

```
S: +OK POP3 server ready <1896.697170952@merlin.fit.vutbr.cz>
C: APOP david e038880ac553077955bd7a80d4454371
S: +OK Pass accepted
```

USER/PASS Během této formy autorizace je uživatelské jméno a heslo přenášeno v čisté formě serveru, klient po připojení k serveru nejprve pošle příkaz: USER username a pokud server klienta pozná, požaduje heslo, které klient dodá příkazem: PASS pass, kde username je uživatelské jméno a pass je heslo z autorizačního souboru (3.4.2).

Ukázka komunikace (S = server, C = klient):

```
S: +OK POP3 server ready <1896.697180952@merlin.fit.vutbr.cz>
C: USER david
S: +OK User accepted
C: PASS wisfit
S: +OK Pass accepted
```

2.2 Zprávy

2.2.1 Formát zpráv

Všechny e-mailové zprávy odpovídají specifikaci IMF [3], každý e-mail má hlavičku a tělo. Hlavička je od těla zprávy oddělena prázdným řádkem, tedy "\r\n\r\n". Ukázkový e-mail tedy může vypadat nějak takto:

```
Return-Path: <root@localhost>
X-Original-To: isa2015@localhost
Delivered-To: isa2015@localhost
Received: from localhost (localhost [127.0.0.1])
        by isa2015 (Postfix) with ESMTP id 3FBBC260CB
        for <isa2015@localhost>; Sun, 22 Oct 2017 16:28:25 +0200 (CEST)
Subject: Email cislo 1
Message-Id: <20171022142834.3FBBC260CB@isa2015>
Date: Sun, 22 Oct 2017 16:28:25 +0200 (CEST)
From: root@localhost

Ahoj,

diky za zpravu, vsechno jsem vyridil, o cil bylo postarano.

Cus,

JB
007
```

2.2.2 Velikost zpráv

Velikost zprávy je počítána v oktetech, což je velikost odpovídající v bytech. Každý e-mail by však měl obsahovat v ideálním případě konce řádků ve formátu: "\r\n"(označujeme jako CRLF, kde CR je "\r"a LF je "\n"), pokud tomu tak není, je nutné, aby server dopočítal správnou velikost dle nekorektních zakončení řádků. Dle specifikace je dáno, že řádky musí končit výše zmíněnou posloupností. Délka se tak poté může lišit v závislosti na systému. Windows systémy CRLF zasílají, Unix systémy však využívají pouze LF, tedy pouze: "\n", naopak oproti tomu, Macintosh systémy a Commodore využívají CR, tedy pouze: "\r".

3 Řešení projektu

3.1 Jazyk

Pro implementaci byl zadán jazyk C/C++. V projektu byl vybrán primárně jazyk C++, zejména pro jeho vylepšenou práci s paměti a prací s řetězci. Místy jsou však i využity operace a standardy, které by byly používány primárně v jazyce C.

3.2 Vývojové prostředí

Pro implementaci byl využit verzovací nástroj git, zejména tedy jeho populární provozovatel Github s využitím studentské licence.

Pro vývoj v jazyce C/C++ byl využit nástroj CLion od firmy JetBrains s využitím studentské licence.

3.3 Popis řešení

3.3.1 Přeložení a spuštění

POP3 server, dále jen server, je možné přeložit za pomocí makefile příkazem: make nebo make all. Příkaze make provede pouze znovupřeložení serveru, make all provede vyčištění adresáře od starých verzí serveru, překlad serveru a následné smazání objektových souborů. V případě potřeby je možné využít i příkazu: make clean-obj a make clean. Příkaz make clean provede smazání starých verzí serveru, make clean-obj provede smazání starých objektových souborů.

Server se následně spouští příkazem: ./popser [-h] [-a PATH] [-c] [-p PORT] [-d PATH] [-r]

Server, má tři formy běhu:

1. ./popser -h – pouhý výpis nápovědy
2. ./popser -r – pouhé provedení resetu serveru do stavu před prvním spuštěním serveru
3. ./popser -a PATH -p PORT -d PATH [-c] [-r] – běžný režim běhu serveru

3.3.2 Popis argumentů při spuštění

Každý argument při spuštění má svůj význam a je nutné jej validovat.

-h je argument spouštějící nápovědu serveru. Je možné jej zadat s jakýmkoliv jiným argumentem, pokud je však zadán, dojde pouze k výpisu nápovědy a ukončení serveru. Tento argument je možné validně využít ve všech formách běhu serveru. V serveru je toto řešeno jako přepínač, který jakmile je aktivován, dojde k vytištění nápovědy a ukončení serveru s návratovým kódem **EXIT_SUCCESS** (3.4.1).

-c je argument, který umožňuje výběr typu autorizace v serveru. Pokud není zadán, je možné využít pouze autorizace přes příkazy APOP, pokud byl zadán, je možné využít pouze autorizace přes kombinaci příkazů USER/PASS. Tento argument je možné validně využít pouze v třetím režimu běhu. V serveru je toto řešeno jako přepínač, který jakmile je aktivován, rozhoduje povoluje jednu z těchto autorizačních metod.

-r je argument provádějící reset serveru, tedy návrat do stavu před prvním spuštěním serveru, tj. dojde k načtení všech e-mailů ze speciálního konfiguračního souboru, v serveru je vždy pojmenován: mail.cfg a je umístěn v root složce serveru. Po provedení resetu se e-maily přesunou ze složky cur do new, změní se název na původní a server se tváří jako by nebyl spuštěn. Toto neplatí pro již smazané e-maily, ty nelze obnovit. Pokud je tento argument předán samostatně, dojde k resetu serveru a ukončení serveru s návratovou hodnotou **EXIT_SUCCESS** (3.4.1), pokud není zadán samostatně a je v kombinaci s třetí variantou spuštění, přejde se k dalšímu spouštění serveru. Tento argument je možné validně využít pouze v druhém a třetím režimu běhu. Popis implementace resetu je popsán v sekci 3.4.7.

-a PATH je argumentem určujícím cestu k souboru s autorizačními údaji uživatele serveru. Tento argument je povinný v třetím režimu běhu. Pokud není v tomto režimu běhu zadán, dojde k ukončení běhu serveru s návratovým kódem **EXIT_FAILURE** (3.4.1), pokud je zadán, dojde k načtení a validaci souboru s autorizačními údaji.

-d PATH je argumentem určujícím cestu k e-mailovému adresáři. Tento argument je povinný v třetím režimu běhu. Pokud není v tomto režimu běhu zadán, dojde k ukončení běhu serveru s neúspěšným návratovým kódem **EXIT_FAILURE** (3.4.1), pokud je zadán, je cesta uložena, ověřena validita cesty a až v pozdější fázi serveru jsou testovány bližší podmínky.

-p PORT je argumentem určujícím číslo portu, na kterém server poběží. Tento argument je povinný v třetím režimu běhu. Pokud není v tomto režimu běhu zadán, dojde k ukončení běhu serveru s návratovým kódem **EXIT_FAILURE** (3.4.1), pokud je zadán, číslo portu je uloženo a je s ním dále pracováno až při spouštění serveru.

3.3.3 Běh serveru

Po spuštění serveru dojde podle režimu běhu k různému chování.

Po spuštění serveru v prvním režimu běhu dojde k vytištění nápovědy a ukončení serveru s návratovým kódem **EXIT_SUCCESS** (3.4.1).

Po spuštění serveru v druhém režimu běhu dojde k resetu serveru a k ukončení programu s návratovým kódem **EXIT_SUCCESS** (3.4.1), pokud nedošlo k žádným problémům a následně se smaže soubor mail.cfg.

Po spuštění serveru v třetím režimu běhu dojde k validaci všech argumentů, následně dojde k validaci autorizačního souboru. Poté dojde k vytvoření socketu, jeho nastavení, k nabindování serverového socketu. Poté server přejde do stavu naslouchání na daném socketu. Pokud vše proběhlo v pořádku, server je spuštěn a čeká na připojení od klienta. Blíže je toto specifikováno v sekci 3.4.5.

Po připojení klienta se klient musí autorizovat, je tedy ve stavu autorizace. Dokud neproběhne úspěšná autorizace, nedostane klient přístup k e-mailům. Jakmile se autorizuje, je mu vyhrazen přístup k e-mailové složce nad kterou může výhradně on provádět e-mailové operace. V tomto momentě je v transakčním stavu. Po odpojení klienta dojde k přesunu do stavu aktualizace, ze kterého úspěšném dokončení přejde do stavu aktualizace, ve kterém smaže e-maily pro smazání a odhlásí klienta a povolí přístup dalšímu klientovi ke složce maildiru. Následně se přesune do stavu čekání na dalšího klienta. Server je schopen přijímat zároveň více klientů, výhradní přístup je však možné dát v jeden moment pouze jednomu klientovi. Podrobně je běh serveru a jeho implementace řešena v sekci 3.4.6.

3.4 Implementace řešení

3.4.1 Návratové stavy

V serveru jsou využity dva návratové stavy. **EXIT_SUCCESS** a **EXIT_FAILURE**. Každý z těchto stavů je makrem pro číselnou hodnotu, bylo vhodně využito těchto maker pro transformovanost a přeložitelnost na různých systémech. Různé typy systémů mohou považovat různé návratové hodnoty za jiný stav. Využitím těchto maker tomuto lze jednoduše předejít.

EXIT_SUCCESS je brán jako hodnota úspěšného ukončení běhu programu. Standardně se na Unixových systémech jedná o číselnou hodnotu **0**.

EXIT_FAILURE je brán jako hodnota neúspěšného ukončení běhu programu. Standardně se na Unixových systémech jedná o číselnou hodnotu **1**.

3.4.2 Soubor s autorizačními údaji

Soubor s autorizačními údaji obsahuje přístupové údaje uživatele k serveru, obsahuje vždy pouze uživatelské jméno a heslo v následujícím formátu:

```
username = NAME
pass = PASS
```

Položka NAME je jméno uživatele, položka PASS je přístupové heslo. Obě tyto položky jsou uloženy v nehashované formě.

3.4.3 Konfigurační soubor

Konfigurační soubor e-mailů obsahuje veškeré e-maily, které po ukončení serveru (3.4.8) server uloží do souboru mail.cfg, který je vždy uložen ve stejné podsložce jako mailserver samotný, kde každému e-mailu jsou vytvořeny dva řádky v jednoduchém formátu:

```
dir = PATH
name = NAME
```

Položka PATH je neúplná cesta k souboru, kterou má server uloženu v paměti ve speciální struktuře mailStruct (3.4.4). Tyto e-maily jsou vždy uloženy ve složce e-mailového adresáře, v podsložce /cur, vždy je tedy uložena pouze cesta bez /cur.

Položka NAME je jméno e-mailu, které je taky uloženo v paměti ve speciální struktuře mailStruct (3.4.4).

Tento soubor je po zapnutí server načten vždy, jsou z něj vždy načteny e-maily. Pokud dojde k resetu (3.4.7), provede se přesun e-mailů do daných souborů do původních složek, tedy do složky e-mailového adresáře, do podsložky /new.

3.4.4 Struktury a jejich kolekce

Server využívá struktur dvou typů.

Jednou z nich je struktura typu threadStruct, která je strukturou vlákna, obsahující potřebné informace pro komunikaci s daným klientem v daném vlákně. Tato struktura je definována následovně:

```
typedef struct {
        string mailDir = "";
        string usersFile = "";
        bool isHashed = true;
        string clientUser = "";
        string serverUser = "";
        string clientPass = "";
        string serverPass = "";
        int commSocket = -1;
        string pidTimeStamp = "";
        bool authorized = false;
} threadStruct;
```

Struktura obsahuje:

string mailDir kořenovou složku e-mailového adresáře

string usersFile cestu k souboru s autorizačními údaji klienta

bool isHashed formu autorizace

string clientUser jméno klienta, které obdrží po komunikaci s klientem

string serverUser jméno klienta, které bylo uloženo v souboru s autorizačními údaji, je zde pro urychlení práce s danými daty

clientPass heslo klienta, které obdrží po komunikaci s klientem

serverPass heslo klienta, které bylo uloženo v souboru s autorizačními údaji, je zde pro urychlení práce s danými daty

int commSocket komunikační socket, pro komunikaci s daným klientem, při inicializaci má hodnotu -1

string pidTimeStamp speciální nonce, která je po startu komunikace klienta se serverem zaslána klientovi

bool authorized stav autorizace klienta

Druhou strukturou, která je v serveru použita, je struktura e-mailu, její definice je následovná:

```
struct mailStruct{
        unsigned long id;
        string name;
        size_t size;
        string dir;
        bool toDelete;
} *mailStructPtr;
```

Struktura obsahuje následující položky:

unsigned int id id jednotlivých e-mailů

string name jméno e-mailu

size_t size velikost daného e-mailu v oktetech 2.2.2

string dir cesta ke kořenové složce e-mailového adresáře

bool toDelete příznak označující e-mail ke smazání

Jednotlivá vlákna jsou pro usnadnění práce s nimi uložena v jednom vektoru vláken s názvem threads. Vektorem můžeme označit dynamické pole umožňující rychlý a efektivní přístup k jednotlivým prvkům.

Jednotlivé e-maily jsou pro usnadnění práce s nimi uloženy v jednom globálním listu nazvaném mailList. List je provázaný seznam prvků umožňující nám snadnou iteraci skrze jednotlivé prvky.

3.4.5 Implementace programu

Server je implementován jako TCP server, který po spuštění provede kontrolu argumentů a jejich rozparsování za pomocí funkcí: `checkParams()` a `parseParams()`. Pokud nejsou správně zadány parametry odpovídající jednomu ze tří běhů aplikace (3.3.1), dojde k ukončení programu s chybovým kódem **EXIT_SUCCESS** (3.4.1), V opačném případě se pokračuje v běhu programu. Dojde ke kontrole parametrů HELP (3.3.2) a RESET (3.3.2). V dalším kroku se provede validace souboru s autorizačními údaji uživatele serveru (3.4.2). Pokud soubor není v pořádku, server vrací návratový kód **EXIT_FAILURE** (3.4.1). Po úspěšné kontrole všech těchto parametrů se začne inicializovat server, nejprve se vytvoří SERVERSOCKET za pomocí funkce `socket()`, následně je tento socket nastaven funkcí `setsockopt()`. Pokud je vše v pořádku, server vytvoří CLIENTADDR a SERVERADDR proměnné typu SOCKADDR_IN, které se naplní vhodnými hodnotami, jako typ spojení a číslo portu. Poté je za pomocí funcke `bind()` socket nabindován a začne poslouchat na daném socketu s názvem SERVERSOCKET. Pro chování TCP serveru je využit nekonečný WHILE cyklus na jehož počátku dojde k přijetí spojení funkcí `accept()`, následně dojde k vytvoření a naplnění struktury THREADSTRUCT, následně dojde k vytvoření vlákna a přístupu do funkce `clientThread()`. Pokud došlo k chybě při přijetí spojení, cyklus se ukončí a program končí. Pokud ne, přechází se už k funkčnosti samotného POP3 serveru, která je podrobně řešena v sekci 3.4.6

3.4.6 Implementace logiky POP3

Ve funkci `clientThread()` se provádí příjem jednotlivých zpráv, je zde tedy druhý nekonečný cyklus (tentokráte FOR), který doplňuje logiku serverového chování. Při přijetí spojení se klientovi okamžitě zašle nonce (2.1.3), na základě které klient ví, že je připojen a odpovídá na ni e-mailovými operacemi. Nyní je server v autorizačním stavu (2.1.1). V tomto momentě může klient zasílat pouze APOP, USER, PASS nebo QUIT příkazy. Funkce je vysvětlena v sekci 2.1.4. Dokud se klient neautorizuje korektními údaji, nemůže přejít do dalšího, transakčního (2.1.1), stavu. Po úspěšné autorizaci klienta dojde k výhradnému uzamčení e-mailového adresáře funkcí `lockMaildir()`, může jej tedy využít pouze daný klient, nikdo jiný. Pokud se autorizace nezdaří, neobdrží přístup a musí se autorizovat znovu. Celou autorizaci řeší funkce `authorizeUser()`, která vrací hodnotu BOOL, ta je uložena do struktury daného vlákna (3.4.4) pro každého klienta zvlášť.

Po uzamčení e-mailového adresáře dojde k vytvoření listu e-mailů pomocí funkce `createListFromMails()` z e-mailů v adresáři /new a pokud se nejedná o první spuštění serveru, musí být přítomen i konfigurační soubor e-mailu obsahující starší e-maily, které byly ve složce /cur, které si také načte funkcí `loadMailsFromCfg()`. Během tohoto tvoření listu jsou e-maily čteny, je zjišťována jejich velikost funkcí `getFileSize()`(2.2.2) a zároveň, pokud se jedná o e-maily ve složce /new, přesouvány do složky /cur. Po dokončení této operace přestoupí server do transakčního stavu přes funkci `executeMailServer()`, která řeší logiku celého POP3 serveru.

Tato funkce obdrží vždy operaci, která má být vykonávána a zprávu, která byla od klienta přijata, tu zpracuje a na základě ní odešle adekvátní reakci. V této fázi může přijímat pouze příkazy: LIST, NOOP, STAT, RETR, DELE, RSET, UIDL, TOP a QUIT (2.1.4). Každá z těchto operací má svou vlastní C++ funkci, přesněji:

`listOperation()`,`noopOperation()`,`statOperation()`,`retrOperation()`,`deleOperation()`, `rsetOperation()`,`uidlOperation()`,`topIndexOperation()`,`quitOperation()`, výjimkou jsou operace LIST, UIDL. Tyto příkazy mají dvě funkce, protože jsou rozlišeny podle počtu argumentů. Jedná se tedy o funkce: `listIndexOperation()` a `uidlIndexOperation()`. Implementace těchto funkcí je řešena dle specifikace RFC 1939 ([2]), které je věnována sekce 2.1.4.

Klient komunikuje s klientem tak dlouho, dokud klient nezašle příkaz QUIT, načež server přejde do stavu aktualizace, kdy dojde ke smazání všech e-mailů, které byly označeny ke smazání, v e-mailovém adresáři. Tento stav je prováděn funkcí `quitOperation()`, která provede smazání všech e-mailů funkcí `deleteMarkedForDeletion()`. Tato funkce zkontroluje všechny e-maily v listu a ty které byly označeny příznakem toDelete (3.4.4), smaže.

Pokud smazání proběhlo v pořádku, server pošle klientovi zprávu obsahující počet e-mailů, které v e-mailovém adresáři zůstali a odpojí jej. Pokud došlo k problému při mazání, server na tento fakt upozorní klienta a taktéž jej odpojí.

Pokud klient nebyl před zasláním příkazu QUIT v transakčním stavu, musel být ve stavu autorizačním, poté dojde k použití stejné funkce, jelikož ale nebyl list naplněn, žádný e-mail se nesmaže a je klient odpojen. Veškerá vlákna klientů se mažou a uvolňuji z paměti až při ukončení serveru (3.4.8).

3.4.7 Reset serveru

Reset serveru do původního stavu před prvním spuštěním probíhá za využití souboru mail.cfg, který je uložen ve stejné složce jako je samotný server. Jeho obsah je zmíněn v sekci 3.4.3. Obsah tohoto souboru je načten za pomocí ifstream, což je třída pracující nad soubory a jejich obsahy, a veškeré e-maily, jejich názvy v tomto souboru jsou, jsou přesunuty do původních pozic, ve kterých byly. Toto neplatí pro e-maily, které již byly smazány. Celý tento proces obstarává funkce `resetMail()`, která opravdu nedělá nic než načtení souboru, postupné načítání názvů a složek jednotlivých e-mailů a jejich přesouvání z /cur do /new jejich e-mailových adresářů.

3.4.8 Implementace ukončení běhu serveru

Ukončení běhu serveru je možné pouze v případě, že je serveru předán signál **SIGINT**. Takový signál je signál označený jako vyvolaný z klávesnice. Je to ukončující signál, jeho účelem je terminovat proces, potažmo server. Lze jej vyvolat z klávesnice stisknutím kláves: CTRL^C. Při takovémto ukončení dojde k odpojení veškerých klientů přes funkci `closeThreads()`, korektně se uloží veškeré informace o e-mailech do konfiguračního souboru (3.4.3) s e-maily užitím funkce `createMailCfg()`, uvolní se veškerá struktury z paměti za užití funkce `disposeList()` a server se ukončí s návratovým kódem **EXIT_SUCCESS** (3.4.1). Toto vše je nutné pro korektní ukončení programu a následné bezproblémové spuštění při opakovaném spuštění serveru.

4 Neimplementované součásti

Nebyla implementována součást automatického odhlášení ze sekce 3, RFC 1939 [2].

5 Implementovaná rozšíření

Proběhla implementace příkazu TOP dle sekce 7, RFC 1939 [2].

Literatura

[1] D. J. Bernstein. Using maildir format. https://cr.yp.to/proto/maildir.html.

[2] Marshall T. Rose John G. Myers. Post Office Protocol - Version 3, 1996. https://www.ietf.org/rfc/rfc1939.txt.

[3] Peter W. Resnick. Internet Message Format, 2008. https://tools.ietf.org/html/rfc5322.

[4] Wikipedie. Index (databáze) – Wikipedie. https://en.wikipedia.org/wiki/Database_index.