VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Síťové aplikace a správa sítí – Projekt

POP3 Server

20. listopadu 2017

Roman Nahálka

Obsah

1 Úvod

Tato dokumentace vznikla k projektu do předmětu Síťové aplikace a správa sítí (ISA). Cílem projektu je vytvořit POP3 [2] Server pro čtení e-mailů. POP3 Server pracuje s e-maily uloženými ve struktuře Maildir[1]. Funkcionalita POP3 serveru je popsána v RFC 1939.

2 Důležité pojmy

V této části dokumentu je popsán základní princip aplikace a popis důležitých pojmů, které se budou v dokumentaci déle vyskytovat. Bude zde popsáno, co to vlastně POP3 je, adresářovou strukturu, ve které budeme mít uloženy e-maily, a formát e-mailů.

2.1 POP3

POP je zkratka pro `Post Office Protocol`. Jedná se o internetový protokol, určený pro čtení e-mailových zpráv ze vzdáleného serveru. Dnes se používá jeho třetí verze, nazvaná právě POP3.

Protokol je popsán v RFC 1939. Tento popis vznikl v roce 1996. Protokol využívá TCP/IP spojení. Standartně probíhá komunikace na TCP portu 110 a komunikace probíhá na principu klient/server.

2.2 Maildir

Maildir je formát využívaný pro uložení e-mailových zpráv na serveru. Každá e-mailová zpráva je zde uložena jako samostatný soubor s unikátním názvem. Jedná se vlastně o složku, obsahující tři podsložky, `cur`, `new` a `tmp`.

2.3 IMF

V našem případě budou e-maily uloženy ve formátu IMF [3]. IMF je zkratka pro `Internet Message Format`. Jedná se o formát, přes který jsou posílaný textové zprávy přes internet. Uložený e-mail se skládá z hlavičky, obsahující různé informace o e-mailu, a z těla, tedy ze samotné zprávy. Tento formát je popsán v RFC 5322. Tento popis vznikl v roce 2008.

3 Implementace

Program je implementován v jazyce C/C++. Program je navržen objektově. Aplikace je vytvořena pro operační systém Linux a byla testovaná na virtuální stroji s Ubuntu 16.04 (64 bit) a na školním serveru `merlin.fir.vutbr.cz`. Přenositelnost na jiný operační systém nebyla testovaná a není tak zaručena.

3.1 Zpracování argumentů

První věc, co program musí po spuštění vykonat je zpracování argumentů příkazové řádky. Pro argumenty jsem navrhl objekt s názvem `Arguments`. Samotné zpracování pak probíhá v metodě `parseArguments()`. V této funkce se správnost argumentů ověřuje pomocí funkce `getopt()`, která nám umožní zpracovávat argumenty v libovolném pořadí. Zpracované argumenty se ukládají do atributů třídy.

3.2 Kontrola správnosti argumentů

Po zpracování argumentů je důležitá jejich kontrola. Tato kontrola probíhá pouze v případe, že nebyla vyžádaná nápověda či reset serveru bez dodatečných argumentů. O tuto kontrolu se stará metoda `checkValidity()`, která volá metody `checkAuthFile()` a `checkPort()`, které zkontrolují správnost zadání portu a správnost autentizačního souboru se jménem a heslem uživatele.

3.3 Navázání spojení

Pokud byli všechny argumenty správně zadaný, nastal čas vytvořit spojení. Pro server jsem navrhl objekt s názvem `Server`. Vytvoření a navázání spojení s klientem probíhá v metodě `connection()`. V této metodě se vytvoří a nastaví socket, obsadí se uživatelem zadaný port, a čeká se na připojení klienta. Po úspěšném navázání spojení se vytvoří nové vlákno, které začíná ve stavu autentizace klienta a provádí ji metoda `authorization()`.

3.4 Stav autentizace

Po navázání spojení se server bude nacházet ve stavu autentizace. Způsob autentizace se bude lišit v závislosti na zadaném parametru `-c`, který určuje, zda se jedná o šifrovanou nebo nešifrovanou autentizaci. Po úspěšné autentizaci se server pokusí uzamknout pro klienta `Maildir` a klient pokračuje do stavu transakce zavoláním metody `transaction()`. Pokud se uzamčení nepovede, uživatel se vrátí do stavu autentizace.

3.4.1 Šifrovaná autentizace

Šifrovaná autentizace probíhá pokud nebyl zadán přepínač `-c`. V tomto případě se uživatel připojuje pomocí příkazu `APOP`. Tento příkaz má dva argumenty, prvním je uživatelské heslo a druhým je heslo zašifrované metodou `md5` v kombinaci s časovým razítkem, které uživatel obdrží při navázání spojení či po neúspěšném pokusu o přihlášení.

Pro algoritmus `md5` jsem použil zdrojový kód dostupný na stránkách zedwood [4] .

3.4.2 Nešifrovaná autentizace

Pokud byl zadán přepínač `-c`, uživatel se přihlašuje k serveru pomocí příkazů `USER` a `PASS`. Uživatel nejdříve zadá příkaz `USER` s argumentem obsahující uživatelské jméno. Poté stejným způsobem pomocí příkazu `PASS` zadá heslo. Pokud byl některý z údajů zadán chybně, server odešle odpověď `-ERR` a bude znovu čekat na pokus o přihlášení.

3.5 Stav transakce

Po úspěšně autentizaci klienta se server přesune do stavu transakce. Ihned po přesunu do tohoto stavu, server přesune všechny soubory z podsložky `new` do podsložky `cur` zavoláním metody `newToCur()`. V tomto stavu server od klienta očekává zadávání příkazů. Povolené příkazy na serveru jsou příkazy `LIST`, `STAT`, `RETR`, `DELE`, `RSET`, `NOOP`, `UIDL` a `QUIT`. Velikost písmen se u příkazů neřeší a příkaz tak může být zadán velkými i malými písmeny, popř. jejich kombinací.

Klient zasílá serveru příkazy až do doby, kdy mu pošle příkaz `QUIT`, kterým se ukončuje spojení klienta se serverem. Po zadání tohoto příkazu se server přesune do stavu update zavoláním metody `update()`.

3.6 Stav update

Po skončení práce klienta se serverem se server přesune do stavu update. V tomto stavu server smaže všechny e-maily označené ke smazání pomocí příkazu `DELE`. Poté, co server smaže všechny označené e-maily, informuje klienta a ukončí s ním spojení. Následně se server vrátí zpět do stavu autentizace.

3.7 Reset

Pokud při spuštění serveru bude zadán přepínač `-r`, znamená to, že máme provést restart serveru, jakoby náš server nebyl nikdy před tím spuštěn. V tomto případě server přesune všechny soubory ze složky `cur` zpět do složky `new` a smaže všechny vytvořené pomocné soubory.

4 Zajímavé části implementace

V této sekci popíši některé zajímavější části implementace aplikace.

4.1 Mazání zpráv

Jedna ze zajímavější častí POP3 serveru je mazání zpráv. Server nemaže soubory s e-maily okamžitě, ale až ve stavu update. Pokud se server při spojení s klientem z nějakého důvodu do stavu update nedostane, soubory nebudou smazány.

Zprávy uložené ke smazání mám tedy uloženy ve `vectoru`, který je atributem třídy `Server` a jeho název je `delMark`. Do tohoto `vectoru` ukládám čísla zpráv, které se mají smazat. Jedná se tedy o `vector`, do kterého ukládám datový typ `int`.

4.2 Pomocné soubory

Pro usnadnění práce serveru vytvářím dva pomocné soubory. V jednom si uchovávám důležité informace o jednotlivých e-mailech a v druhém cestu k Maildiru. První soubor mám pojmenovaný `pom.pop` a je v něm uložený název souboru, velikost v bajtech, kterou je nutné přičíst k jeho velikosti, aby velikost odpovídala řádkování `CRLF`, a posledním uloženým údajem je jeho unikátní identifikační číslo pro příkaz `UIDL`. Tyto informace jsou uloženy z důvodu, aby server nemusel soubory dokola otevírat při každém spuštění. Informace jsou tedy do souboru uloženy při přesunu souborů. Druhý soubor jsem pojmenoval `dir.pop` a je v něm uložená cesta k Maildiru, pro případ, že uživatel bude chtít provést reset serveru bez specifikace cesty k Maildiru.

4.3 Unikátní identifikační číslo

Aby měl uživatel přehled o jednotlivých e-mailech, je každé zprávě přiřazeno unikátní identifikační číslo. Já jsem se rozhodl toto číslo vytvářet pomocí funkce `std::hash`. Do této funkce pošlu název souboru, ze kterého mi funkce vytvoří identifikační číslo, které bude pro každý soubor unikátní. Klient toto číslo zjistí pomocí příkazu `UIDL`. Identifikační číslo je každému souboru přiřazeno při přesunu souborů a je zaznamenáno v pomocném souboru.

5 Použití aplikace

Program se spouští následujícím způsobem:
`./popser [-h] [-a PATH] [-c] [-p PORT] [-d PATH] [-r]`, kde:

- h – Vypíše nápovědu
- a – Cesta k autentizačnímu souboru
- c – Povolení nešifrované autentizační metody
- p – Číslo portu, na kterém bude server běžet
- d – Cesta do složky Maildir
- r – Reset serveru

Server může běžet ve 3 režimech běhu a to v následujících:

- Výpis nápovědy – Zadaný parametr `-h`
- Pouze reset – Zadaný pouze parametr `-r`
- Běžný režim – Zadané parametry `-a`, `-p` a `-d` a volitelné parametry `-c` a `-r`

6 Závěr

Aplikace spustí POP3 server a multi-vláknově obsluhuje klienty. Funkčnost aplikace byla řádně ověřena na virtuální stroji s operačním systémem Ubuntu 16.04 a na školním serveru `merlin.fit.vutbr.cz`. Program je překládán překladačem `g++`. Pro překlad aplikace je přiložen soubor `Makefile`.

Reference

[1] Bernstein, D.: Maildir. [online].
URL <https://cr.yp.to/proto/maildir.html>

[2] Myers, J. G.; Rose, M. T.: RFC 1939. [online].
URL <https://www.ietf.org/rfc/rfc1939.txt>

[3] Resnick, P. W.: RFC 5322. [online].
URL <https://tools.ietf.org/html/rfc5322>

[4] Thilo, F.: C++ md5 function. [online].
URL <http://www.zedwood.com/article/cpp-md5-function>