VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ISA - Síťové aplikace a správa sítí

POP3 server

20. listopadu 2017 Adrián Tomašov

Obsah

1 Abstrakt

Cieľom tohto projektu bolo naimplementovať POP3 server, ktorý slúži na prenos mailovej pošty zo servera ku klientovi. Protokol tejto komunikácie je definovaný v RFC-1939. Tento server mohol bežať iba ako jeden proces v systéme(obmedzené zadaním projektu). Server bude pracovať vždy len s jednou mailovou zložkou, do ktorej sa budú pripájať uživatelia s výlučným prístupom k danej zložke. Maily sú na servery uložené vo formáte IMF, ktorý je definovaný RFC-5322.

2 POP3

POP3 protokol je je štandardizovaný textový protokol na prenos mailov zo servera ku klientovi. Zameriava sa na pár jednoduchých funkcií, ktoré sú dostačujúce na stiahnutie mailov, prípadne ich manipuláciu v zložke uloženej na servery[vymazanie]. Pre zložitejšie operácie v mailovej zložke sa prednostne používa IMAP. POP3 protokol nerozlišuje veľkosti písmen v príkazoch. Na rozdelenie riadkov sa používa CRLF.

Na spojenie užívateľa so serverom sa používa TCP spojenie, aby bol prenos mailov spoľahlivý a bez chýb. Pre nešifrovanú komunikáciu server počúva na porte 110.

POP3 protokol podporuje dva typy autentifikácie: v nešifrovanom a šifrovanom formáte. Šifrovaný formát používa pre výpočet šifry algoritmus MD5, ktorý je dosť zraniteľný. Dnes sa odporúča použiť šifrované TCP spojenie pomocou SSL/TLS. Pre šifrovanú komunikácia server počúva na známom porte 995.

3 Implementácia

Tento projekt bol implementovaný v jazyku C++11 v kombinácií s konštrukciami v jazykuC. Jazyk C bol použitý hlavne kvôli systémovým volaniam read a write nad file deskriptorom. Svoje využitie si našiel aj pri čítaní súborov, v ktorých boli uložené maily. Nad týmito súbormi boli robené rôzne operacie, aby spĺňali formát, ktorý POP3 protokol dokázal odoslať. Aj napriek tomu, že sú tieto operácie možné aj v jazyku C++11 som sa pre väčšiu efektivitu rozhodol použiť konštrukcie jazyka C.

3.1 Mailova zložka

Všetky mail na servery sú uložené v zložke formátu Maildir. Táto zložka obsahuje tri zložky: cur , new a tmp. Nové maily, ktoré príjma služba SMTP a dočasne si ic ukladá do tmp, sú presunuté do zložky new. Po spustení nami vytvoreného POP3 servera a prihlásení uživateľa, sú maily presunuté z new do cur. Ak príde nejaký mail počas doby, keď je užívateľ prihlásený, mail ostavajú v zložke new. Pre prístup k týmto mailom sa užívateľ musí ohlásiť a znova prihlásiť na server.

Pre implementáciu tohto chovania bola vytvorená trieda MailDir, ktorá pristupuje do mailovej zložky v súborovom systéme. Po prihlásení užívateľa sa zavolá metóda read_dirs() , ktorá vyhľadá všetky mail v zložkách new a cur. Tieto maily sú v uložené do objektov typu MailMsg. Tento objekt pri inicializácií prečíta celý mail zo súborového systému do pamäte programu. Pri tejto činnosti urobí aj potrebné opatrenia, aby mail splňoval všetky podmienky definované pre formát IMF . Konkrétne sa jedná nohradenie LF za CRLF, ak je to nutné. Objekt ešte kontroluje či sa na začiatku riadka nenachádza znak ”.”. Ak áno, objekt pridá ešte jeden znak bodky na začiatok riadka.

3.1.1 Logovanie činnosti zložky

Aby sme umožnili vrátenie zmien na súborovom systéme, bola implementovaná trieda MailDirLogger. Obsahuje 2 statické metódy. Prvá metóda, add_log($souce, destination$), slúži na pridanie záznamu do presunutí mailu medzi zložkami. Druhá metóda, reset(), slúži na zvrátenie presunov mailov na súborovom systéme. Táto metóda vola funkciu rename($destination, source$) pre každý záznam, aby

vrátila zmeny. Návratový kód tejto funkcie je ignorovaný z dôvodu vymazania mailu zo súborového systému.

3.2 Komunikácia s klientami

Vzhľadom k tomu, že server beží len v jednom procese a len v jednom threade bolo nutné použiť neblokujúce sockety. Pre obsluhu viacerých socketov v jedno procese bolo použité systémové volanie `select()`, do ktorej sa ako argument zadal list socketov. Jadro OS na tomto volaní čaká, až kým nie je aspoň nad jedným zo socketov umožnené urobiť pre nás potrebnú operáciu(`read`, `accept`). Toto chovanie bolo implementované v triede `Server` v metóde `run()`. Pri pripojení nového klienta sa vytvárajú a ukladajú 2 objeky: `HostConnection` a `Host`.

`HostConnection` obsahuje potrebné objekty pre komunikáciu klienta so serverom. Kľúčový objekt pre komunikáciu je `MsgBuffer`. Tento objekt obsahuje vstupné a vystupné fronty komunikácie pre prípad, že správa nepríde celá alebo bolo prijatých viac príkazov než len jeden. Ďalej obsahuje metódy pre zapisovanie raw dát na socket. Tieto metódy sú využívané pri spracovávaní príkazov `RETR` a `TOP`.

Objekt `Host` si drží v pamäti potrebné informácie pre autonomnú komunikáciu klienta so serverom. Vytvorí si pre seba vlastnú inštanciu triedy `POP3server`. Tento objekt si ukladá aj ukazateľ na objekt `HostConnection`, ktorý vytvára welcome socket. Prácu s welcome socketom zaobaluje objekt typu `Connection`. Ak objekt `Server` zaznamená udalosť na sockete, ktorý patrí objektu `Host -> HostConnection` zavolá sa metóda `run()`. Tu sa načíta správa z `MsgBuffera` a je predaná `POP3Serveru` na spracovanie. Na základe tejto správy POP3server vykoná potrebné operácie a vráti odpoveď do `MsgBuffera`, ktorý ju následne pošle klientovi.

Objekty typu `Host` sú uložene v slovníku, pre správnu prácu s dynamicky alokovanou pamäťou. Kľúč do slovníka je číslo file deskriptora na socket, v ktorom je pripojený užívateľ. Unikátnosť klúča nám zaručí jadro OS, pretože neotvorí dva krát ten istý file deskriptor. Pri odhláseni hosta sa záznam v tomno slovníku vymaže a pamät správne uvolní.

```
std::map<int, Host*> hosts_list;
```

3.3 Trieda POP3Server

Implementácia samotného POP3 protokolu sa nachádza v triede `POP3Server`. Hlavnou myšlienkou tejto implementácie je stavový automat. Pri predaní prvej správy je v server v stave `Authorization`. Po autentifikácií užívateľa sa server dostáva do stavu `Transaction`. Po prevedení všetky príkazov, užívateľ opúšťa server a mal by sa dostať do stavu `Update`. Tento stav je v tejto implementácií zanedbaný. Všetky operácie, ktoré sa vykonávajú v tomto stave, sú implementované v deštruktoroch daných objektov. Jedná sa konrétne o deštruktor v triede `MailMsg`, ktorý v prípade potreby vymaže daný mail zo súborovéoh systému.

Ako už bolo spomínané, každému pripojenému užívateľovi je vytvorená vlastná inštancia triedy `POP3server`. Správy sa tomuto objektu zasielajú pomocou metódy `recv_msg()`. Zpráva sa v tejto metóde rozdelí na jednotlivé slová a ako oddeľovač sa použiju biele znaky. Slová sa uložia do listu, ktorý je ako argument predavaný ostaným metódam totho objektu, ktoré predstavujú jednotlivé operácie definovane v POP3 protokole. Jedinú výnimku tvorí operácia `pass()`, pretože v `RFC-1939` je definované, že helslo môže obsahovať aj medzery.

Po zavolaní operácie sa ako prvé skontrolujú obmedzenia danej operácie ako sú napríklad stav `POP3servera` alebo povinnosť argumentov. Kontrolujú sa aj rozsahy mailovej zložky, aby mail zadaný číslo neukazoval na neexistujúci mail. Po splnení všetkých vstupných podmienok sa vyhodnotí telo operácie, ktorá vráti odpoveď pre užívateľav datovom type `std::string`. V prípade operácií `TOP` a `RETR` sa vyvolá výnimka, pretože by konverzí dat z mailu, ktoré sú uložené v buffry typu `char*`, na `std::string` mohla nastať chyba. Ďalšiu výnimku tvorí operácia `quit`, ktorá upozorní objekt

Server, že daný host chce ukončiť komunikáciu. V tomto prípadne mu server pošle goodbye-message, uzavrie sa socket v HostConnection->socket a uvolní dynamicky alokovanú pamäť.

3.4 MD5

V diskusnom fóre k projektu bolo napísané, že sa po nás nepredpokladá implementácia funkcií pre výpočet hashu pomocou MD5 algoritmu. Celý modul md5 bol stiahnutý zo stránky http://www.zedwood.com/article/cpp-md5-function. Z tohoto modulu sa používa funckia md5().

3.5 Výnimky

V celom projekte boli použité výnimky pre ošetrenie špecialnych stavovov programu. Ak server sa program dostal do chybového stavu a už nemohol ďalej pokračovať vo svojej činnosti vyvoval výnimku Exception(). Táto výnimka je definovaná v našom programe a dedí z výnimky std::exception. Pri dedení sa preťažuje metóda what(), ktorá má vypísať chybovú hlášku o skončení programu na std::cerr. Táto metóda sa volá automaticky, takže nami upravená metóda vracia reťazec, ktorý bude vypisaný ako chyba.

3.5.1 ArgParser výnimky

Pre triedu, ktorá ma za úlohu získať informácie z príkazového riakdu boli impementované 3 výnimky pre špeciálne stavy. Ak sa ako argument programu na príkazovom riadku objeví prepínač -h, vyvolá sa výnimka HelpException, ktorá informuje program o tom, že má vypísať krátku manuál na std::cout a ukončiť svoju činnosť. Ďalej sú definované výnimky, ktoré súvisia s prepínačom -r. Ak je tento prepínač zadaný ďalších argumentov na príkazovom riadku, vyvolá sa výnimka ResetAndExit, ktorá informuje program o tom, že má zavolať statickú metódu MailDirLogger::reset() a ukončiť svoju činnosť. Ak sa však na príkazovom riadku objavia aj iné prepínače vrátane -r, skontroluje sa ich integrita a následne sa vyvolá výnimka ResetAndStart. Táto výnimka informuje program o tom, že pred spustením hlavného cyklu programu Server::run(), má dať mailovú zložku do pôvodného stavu.

3.5.2 POP3Server výnimky

Objekty tohto typu vyvolávajú tri typy podmienok. Prvou z nich je QuitServer a informuje program o vyžiadaním o ukončenie spojenia s užívateľom. Druhá výnimka používaná týmito objektami je SendMailMsg. Informuje objekt typu Host, aby zavolal špeciálnu metódu MsgBuffer::direct_send(), ktorá pošle užívateľovi odpoveď od POP3Servera a k tomu pribalí mial. Mail je uložený a správne naformatovaný v MailMsg->content. Za toto všetko sa prida znak "." a CRLF, ktorý nám označí koniec správy zaslanej serverom. Poslednou výnimkou generovanou objektami POP3Server je SendMailMsgLines Táto výnimka vynutí program zavolať metódu MsgBuffer::direct_send_lines(), ktorá má takmer rovnakú úlohu ako MsgBuffer::direct_send(), ale pošle iba užívateľom definovaný počet riadkov mail.

4 Zapnutie debug výpisov

Pri vývoji tohto projektu bolo vyvynuté makro echo(). Makro vypisuje súbor, číslo riadku a funkciu z ktorej sa toto makro volá.Ak chcete program preložiť s týmito výpismi je nutné pridať do premennej CXXFLAGS v makefile prepínač -DSERVER_LOGGING a projekt znova preložiť.

```
#define echo(a) std::cout << __FILE__ << " : " << __LINE__ << " : " \
<< __FUNCTION__ << " >> "<< (a) << std::endl;
```

4.1 Príklad debug vypisu

```
main.cpp : 40 : main >> Parsed arguments
MailDirLogger.cpp : 29 : reset >> Reseting direcotry
MailDirLogger.cpp : 63 : reset >> Logger file waas NOT found
main.cpp : 59 : main >> Arguments checked and OK
main.cpp : 60 : main >> Creating connection
Connection.cpp : 20 : Connection >> Conncection Constructor
Connection.cpp : 22 : Connection >> 2001
Connection.cpp : 41 : open_socket >> 2001
main.cpp : 64 : main >> Creating and running server
Server.cpp : 41 : run >> Listening on connection socket
Host.cpp : 24 : Host >> initializing pop3server object in Host constructor
Pop3Server.cpp : 374 : read_auth_file >> xtomas32
Pop3Server.cpp : 375 : read_auth_file >> trolo
Server.cpp : 90 : create_host >> Adding new host to container
Server.cpp : 93 : create_host >> say hello to new host
Host.cpp : 62 : say_hello >> Saying helo
Pop3Server.cpp : 389 : greeting >> Timestamp = <3484.1511205148@netperf-at-t460s
Host.cpp : 29 : run >> host run
MsgBuffer.cpp : 48 : recv_msg >> -1
Host.cpp : 32 : run >> msg received and forwarded to pop3server
Host.cpp : 29 : run >> host run
MsgBuffer.cpp : 48 : recv_msg >> -1
Host.cpp : 32 : run >> msg received and forwarded to pop3server
Pop3Server.cpp : 286 : pass >> trolo
Pop3Server.cpp : 287 : pass >> 5
Pop3Server.cpp : 293 : pass >> Real pass >> trolo
MailMsg.cpp : 32 : read_content >> Reading : data/adio/cur/6
MailMsg.cpp : 32 : read_content >> Reading : data/adio/new/1,DR
MailMsg.cpp : 32 : read_content >> Reading : data/adio/new/2
MailMsg.cpp : 32 : read_content >> Reading : data/adio/new/3
MailMsg.cpp : 32 : read_content >> Reading : data/adio/new/4
Host.cpp : 29 : run >> host run
MsgBuffer.cpp : 48 : recv_msg >> -1
Host.cpp : 32 : run >> msg received and forwarded to pop3server
Host.cpp : 29 : run >> host run
MsgBuffer.cpp : 48 : recv_msg >> -1
Host.cpp : 32 : run >> msg received and forwarded to pop3server
Host.cpp : 37 : run >> SendMailMsg exception raised
MsgBuffer.cpp : 90 : direct_send >> Response size >> 16
MsgBuffer.cpp : 91 : direct_send >> Content size >> 254
MsgBuffer.cpp : 101 : direct_send >> Buffer size >> 270
Host.cpp : 29 : run >> host run
MsgBuffer.cpp : 48 : recv_msg >> -1
Host.cpp : 32 : run >> msg received and forwarded to pop3server
Host.cpp : 29 : run >> host run
```

```
MsgBuffer.cpp : 48 : recv_msg >> -1
Host.cpp : 32 : run >> msg received and forwarded to pop3server
Server.cpp : 75 : run >> Host will be deleted
^Cmain.cpp : 84 : sig_int_handler >> Signal capture in main
main.cpp : 85 : sig_int_handler >> Killing server
```