Vysoké učení technické v Brně
Fakulta informačních technologií

Síťové aplikace a správa sítí

Dokumentácia k projektu
HTTP nástenka

18. listopadu 2019

Tomáš Smädo (xsmado00)

Obsah

1 Popis aplikácie

Aplikácia nástenka pozostáva z klienta a servera, ktoré spolu navzájom komunikujú pomocou protokolu HTTP verzie 1.1. Užívateľovi poskytuje možnosť pomocou klienta pridávať, zobrazovať, upravovať či mazať jednotlivé nástenky/príspevky na serveri.

1.1 Implementácia

Pri implementácii aplikácie som postupoval podľa prednášok, štandardov RFC a manuálových stránok Linuxu. Aplikácia je implementovaná v jazyku C++ pomocou sieťovej knižnice BSD sockets, na písanie kódu bolo použité IDE CLion. Skladá sa z 2 zdrojových súborov a jedného hlavičkového súboru:

- isaclient.cpp - implementácia klientskej časti aplikácie
- isaserver.cpp - implementácia serverovej časti aplikácie
- isaproject.h - obsahuje makrá pre stavové kódy HTTP, spoločné funkcie a includnuté knižnice

1.2 Klient

Klient ako prvé spracováva argumenty pomocou funkcie `checkArgs`, ktorá využíva `getopt`, a vracia načítané hodnoty vo vektore. Nasleduje funkcia `commandTransform`, ktorá transformuje načítané argumenty tvoriace `<command>` na ekvivalentné API (spôsob spustenia popísaný v sekcii 2 a API v podsekcii 1.4). Potom klient zaháji pokus o pripojenie sa na server, odošle svoju správu danému serveru a očakáva od neho odpoveď.

Po prijatí odpovede je odpoveď rozdelená funkciou `splitString` na základe dvoch po sebe idúcich CR-LF párov na hlavičku a prípadné dáta. Hlavička je vypísaná na `stderr` a ak boli v správe aj dáta, sú vypísané na `stdout`. Po úspešnom vykonaní server končí s kódom `0`, pri akejkoľvek chybe je návratová hodnota `-1` a užívateľ je informovaný krátkou chybovou hláškou.

1.3 Server

Server po spustení spracuje argumenty rovnako ako klient, a potom čaká na prichádzajúce pripojenia na špecifikovanom porte v nekonečnom cykle `while`. Po pripojení klienta a prijatí správy začína jej spracovanie. Správa je rozdelená rovnako ako u klienta na hlavičku a prípadný prijatý text.

Následne je na základe HTTP metódy z prijatej hlavičky rozhodnuté, o akú požiadavku sa jedná a ako bude vyhodnotená. Pomocou regulárnych výrazov sú skontrolované API príkazy a verzia HTTP (musí byť 1.1). Ak sa jedná o metódu, pri ktorej sa očakávajú prijaté textové dáta, kontroluje funkcia `checkContentHeaders`, či hlavička obsahuje príslušné polia a ich správnu hodnotu (pri nájdení chyby odpovedá klient kódom 400). Server ďalej pomocou funkcií `getName` a `getId` získava z prijatého API príkazu meno nástenky, prípadne číslo príspevku s ktorým bude manipulovať.

1.3.1 Nástenky

Jednu nástenku tvorí mnou definovaná štruktúra `boards`, obsahujúca `string name` - meno nástenky a `vector<string> content` - obsah nástenky (príspevky). Všetky nástenky sú potom uložené v premennej `vector<board> boards`. Pre vyhľadávanie násteniek sa využíva `findPosition`, `existsEntry` pre vyhľadávanie príspevkov. Obe funkcie vracajú `pair<unsigned long, bool>`, pričom číslo určuje pozíciu hľadaného prvku a pravdivostná hodnota či hľadaný prvok vôbec existuje. Funkcia `printBoardContent` vypíše príspevky z požadovanej nástenky do stringu, ktorý sa potom posiela ako odpoveď klientovi.

1.4 API

- GET /boards - Vráti zoznam dostupných nástenok, jedna na riadok.
- POST /boards/name - Vytvorí novú prázdnu nástenku s názvom name.
- DELETE /boards/name - Zmaže nástenku name a všetok jej obsah.
- GET /board/name - Zobrazí obsah nástenky name.
- POST /board/name - Vloží nový príspevok do nástenky name (na koniec zoznamu)
- PUT /board/name/id - Zmení obsah príspevku číslo id v nástenke name.
- DELETE /board/name/id - Zmaže príspevok číslo id z nástenky name.

1.5 Komunikácia klient-server

Komunikácia prebieha pomocou socketov, pre preklad hosta na adresu sa používa `gethostbyname`. Veľkosť buffera pre odosielanie a prijímanie je nastavená na `BUFSIZ` (defaultné C++ makro, veľkosť 8192), čo môže byť zbytočne veľa pre jednoduché požiadavky, avšak dáva priestor a variabilitu pre pridávanie dlhých príspevkov do násteniek.

Hlavička requestu od klienta sa vždy skladá minimálne z:

- metódy, API, verzie HTTP (napr. GET /boards HTTP/1.1)
- hosta (napr. Host: merlin.fit.vutbr.cz)

Odpoveď od serveru sa líši len v prvom riadku, ktorý vyzerá nasledovne:

- verzia HTTP, stavový kód (napr. HTTP/1.1 200 OK)

Server odpovedá na všetky neznáme requesty kódom `404 Not Found`, na chyby alebo requesty popísané v zadaní odpovedá príslušnými kódmi.

Ak request alebo odpoveď obsahuje aj nejaké textové dáta na odoslanie (content), pridávajú sa do hlavičky:

- Content-Type: text/plain
- Content-Length: <dĺžka obsahu>

2 Použitie

Klient aj server akceptujú dva spoločné argumenty:

- **-h** - vypís pomoci k danej aplikácii (akceptovaný len ak je použitý samostatne)
- **-p** - očakáva 1 povinný číselný argument určujúci port

2.1 Spustenie serveru

Server podporuje len 2 vyššie uvedené parametre a jeho spustenie vyzerá nasledovne
`./isaserver -p <port>`

2.2 Spustenie klienta

Klient má navyše oproti serveru ešte ďalšie parametre a to:

- **-H** - adresa serveru pre komunikáciu (host)
- <**command**> - príkaz pre server

kde <command> môže byť (za pomlčkou vždy nasleduje ekvivalenté API):

- boards - GET /boards
- board add <name> - POST /boards/<name>
- board delete <name> - DELETE /boards/<name>
- board list <name> - GET /board/<name>
- item add <name> <content> - POST /board/<name>
- item delete <name> <id> - DELETE /board/<name>/<id>
- item update <name> <id> <content> - PUT /board/<name>/<id>

Spustenie klienta vyzerá nasledovne `./isaclient -H <host> -p <port> <command>` pričom poradie argumentov musí byť dodržané. Ak <name> nespĺňa požadované limity (a-zA-Z0-9), klient je okamžite ukončený s návratovou hodnotou -1.

3 Testovanie

Projekt bol testovaný na mojom notebooku s Ubuntu a na serveri merlin. Na uľahčenie testovania som používal aj jednoduché .sh skripty, taktiež som kompatibilitu servera testoval aj voči kamarátovmu klientovi a pomocou nástroja `curl`. Funkčnosť by mala na základe mojich výsledkov odpovedať zadaniu, pričom dodadtky alebo odchýlky sú popísané v tejto dokumentácii.

4 Použité zdroje

https://en.wikipedia.org/wiki/ListofHTTPstatuscodes
https://regexr.com/
https://medium.com/from-the-scratch/http-server-what-do-you-need-to-know-to-build-a-simple-http-server-from-scratch-d1ef8945e4fa
https://en.wikipedia.org/wiki/HypertextTransferProtocolRequestmethods