Vysoké učení technické v Brně
Fakulta informačních technologií

ISA – dokumentácia k projektu

HTTP nástenka

18. novembra 2019 Natália Holková

Obsah

1 Úvod

Úlohou v tomto projekte bolo vytvoriť aplikáciu, ktorá by umožňovala klientom na serveri vytvárať, upravovať, prezerať a mazať nástenky a príspevky v nich pomocou HTTP API. Bolo potrebné vytvoriť naprogramovať ako klienta, tak aj server. Výsledné riešenie bolo implementované v jazyku C a využíva HTTP/1.1 protokol.

2 HTTP protokol

HTTP (Hypetext transfer protocol) je protokol slúžiaci na prenos html dokumentov medzi klientami a servermi na internete [wikiHTTP].

Tento protokol definuje požiadavky a odpovede medzi klientami a servermi. Klientom býva zvyčajne webový prehliadač, ktorý začína požiadavku nadviazaním TCP spojenia na určenom porte (štandarne 80). Server naopak počúva na danom port, kým mu klient neodošle reťazec s požiadavkou v správnom tvare.

Existuje viacero verzií HTTP protokolu, a to HTTP/0.9, HTTP/1.0 a HTTP/1.1. V súčasnosti je najpoužívanejšia verzia HTTP/1.1, ktorá bola definovaná v roku 1999 v [RFC 2616]. Je spätne kompatibilný s HTTP/1.0, pridáva ďalšie možnosti, metódy a obsahuje takmer dvojnásobný počet hlavičiek oproti predchádzajúcej verzii.

2.1 Metódy

HTTP/1.1 podporuje viacero metód, avšak najpoužívanejšie, ktoré sa zároveň využíli v projekte sú:

- **GET** - žiadanie o zdroj uvedením URL
- **POST** - podobný metóde GET, ale je pridané telo správy
- **PUT** - úprava dát
- **DELETE** - zmazanie zdroja

2.2 Klientské požiadavky

Klienská požiadavka pri použití HTTP/1.1 má nasledujúci formát [HTTP]:

```
GET URL HTTP/1.1
Host: host:port
```

Toto je hlavička HTTP požiadavky. Prvý riadok predstavuje tzv. *request line*, všeobecne býva v tvare `názov-metódy URI-požiadavky HTTP-verzia`. Za request line nasledujú zvyšné hlavičky - *request headers*. Pri HTTP/1.1 je potrebná minimálne hlavička `Host` udávajúca adresu serveru, na ktorý bude odoslaná požiadavka.

V prípade, že sa posiela aj obsah v tele správy, je potrebné nadefinovať aj hlavičku `Content-Type` definujúcu typ zasielaného obsahu, napr. `text/plain` pre čistý text alebo `text/html` pre HTML stránky. Následne je potrebné vypočítať dĺžku zasielaného obsahu a tú zapísať do hlavičky `Content-Length`. Hlavičky sú od tela správy oddelená prázdnym riadkom.

2.3 Odpoveď servera

Odpoveď servera môže byť nasledovná:

```
HTTP/1.1 200 OK
```

teda najskôr sa uvádza verzia HTTP protokolu, potom návratová hodnota a nakoniec bližší popis návratovej hodnoty. Návratová hodnota je trojčíslie označujúce výsledok vykonania požiadavky. Najbežnejšie návratové hodnoty sú `200 OK`, `404 Not Found`, `403 Forbidden` alebo `500 Internal Server Error`. V projekte ďalej využívame `201 Created`, `400 Bad Request` a `409 Conflict`. Zoznam všetkých kódov je možné nájsť v [RFC 2616].

Pokiaľ server zasiela v odpovedi aj nejaký obsah, postup je rovnaký ako pri požiadavke.

3 Implementácia

Implementácia projektu je rozdelená do viacerých zdrojových súborov a to `isaclient.c`, `isaserver.c` a `common.c`, pričom každý súbor má ešte vlastný hlavičkový súbor. Súbory `isaclient.*` obsahujú implementáciu klienstkej aplikácie, `isaserver.*` serverovej aplikácie a `common.*` zasa podporné funkcie a definície, ktoré sa využívajú ako v klientovi, tak aj na serveri.

3.1 Klient

Klient najskôr spracuje zadané argumenty z príkazového riadku. Kontroluje, či je číslo portu vrámci povoleného rozsahu. Zvyšné argumenty reprezentujúce API príkaz spojí do jedného reťazca. Z neho určí, o ktorý typ príkazu sa jedná. Podľa typu príkazu z reťazca vyberie prípadné argumenty `<name>`, `<id>` alebo `<content>`. Zároveň kontroluje ich správnosť, teda či neobsahujú nepovolené znaky.

Jadro celého klienta je vo funkcii `send_and_get_http_response()`, ktorá realizuje HTTP komunikáciu. Najskôr vytvorí request line na základe typu API volania, potom host header a v prípade, že sa zasiela obsah pridá hlavičky `Content-Type` a `Content-Length`. Jednotlivé hlavičky sú oddelené od seba reťazvom $\backslash r \backslash n$ posledná hlavička je od tela oddelená $\backslash r \backslash n \backslash r \backslash n$.

Po zostavení požiadavku sa vytvorí socket vo funkcii `establish_connection()` a pokiaľ všetko správne prebehlo, odošle sa funkciou `send()`. Vytvorí sa buffer konečnej veľkosti, do ktorého sa bude ukladať odpoveď po jednotlivých znakoch pomocou funkcie `recv()`. Z prijatej odpovede sa získa iba telo správy, ktoré sa nachádza za reťazcom $\backslash r \backslash n \backslash r \backslash n$ a to sa vypíše.

3.2 Server

Server najskôr spracuje argumenty, konkrétne iba číslo portu, na ktorom má načúvať a ktoré musí byť v správom rozsahu. Pre ukladanie nástenieк a položiek sa využíva lineárny zoznam. Inicializuje sa štruktúra `Boards_t` obsahujúca všetky nástenky, ktorá obsahuje odkaz na prvú nástenku v zozname. Každá nástenka je definovaná štruktúrou `Board_t` a má svoje meno, odkaz na prvú položku a odkaz na nasledujúcu nástenku v zozname. Štruktúra položky v nástenke `Item_t` obsahuje iba odkaz na ďalšiu položku a jej obsah vo forme reťazca. Id položky sa zisťuje rovno pri prechádzaní zoznamu položiek, pretože sa môže za behu servera meniť.

Po inicializácii zoznamu nástenieк začne server načúvať na danom porte (až pokiaľ nenastane chyba alebo nie je násilne ukončený program). Po prijatí packetu sa najskôr oddelí request line, podľa ktorej sa zistí o ktoré API volanie sa jedná. Porovnáva sa tu druh metódy a URI. Podľa typu príkazu sa z URI získajú parametre `<name>` a `<id>`, `<content>` sa získa z tela požiadavky. Zároveň sa kontroluje správnosť argumentov.

Vykoná sa žiadaný príkaz na strane serveru pomocou funkcie `run_command()`, ktorá vráti návratovú hodnotu a obsah na odoslanie. Následne sa iba zostaví odpoveď na odoslanie a pošle sa späť klientovi.

4 Spustenia a používanie programu

4.1 Preklad

Kompilácia programu sa realizuje cez `Makefile`. Ten sa môže volať s nasledujúcimi parametrami:

- **bez argumentov/all** - preloží zároveň klienta aj server
- **isaserver** - preloží iba server
- **isaclient** - preloží iba klienta
- **clean** - čistenie
- **tar** - vytvorenie tar zložky na odovzdanie

Po kompilácii vznikne aplikácia `isaclient` pre spustenie klienta a `isaserver` pre spustenie servera.

4.2 Spustenie

Server sa spúšťa nasledovne:

```
./isaserver -p <port>
```

kde `<port>` reprezentuje číslo portu, na ktorom sa má počúvať.

Klient sa spúšťa príkazom:

```
./isaclient -H <host> -p <port> <command>
```

kde `<host>` a `<port>` spolu reprezentujú adresu serveru s HTTP nástenkou. Parameter `<command>` reprezentuje API príkaz, ktorý môže mať jeden z nasledujúcich tvarov:

- **boards** - vypíše zoznam dostupných nástenie k
- **board add** <**name**> - vytvorí novú nástenku so zadaným názvom
- **board delete** <**name**> - vymaže nástenku so zadaným názvom
- **board list** <**name**> - vypíše obsah zadanej nástenky
- **item add** <**name**> **¡content**> - pridá nový príspevok na koniec zadanej nástenky
- **item delete** <**name**> <**id**> - vymaže príspevok s daným ID zo zvolenej nástenky
- **item update** <**name**> <**id**> <**content**> - upraví obsah zvoleného príspevku v nástenke

Obe aplikácie podporujú ešte spustenie iba s paramentrom `-h`, kedy sa vypíše nápoveda k používaniu programu.

4.3 Obmedzenia

Aplikácia podporuje iba komunikáciu pomocou **IPv4**. Názov nástenky môže obsahovať iba znaky **a-z**, **A-Z** a **0-9**.

Citácie

[RFC 2616] R. Fielding. *Hypertext Transfer Protocol – HTTP/1.1*. RFC 2616. Jún 1999. URL: https://www.rfc-editor.org/rfc/rfc2616.txt.

[HTTP] *HTTP (HyperText Transfer Protocol)*. 2009. URL: https://www3.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html.

[wikiHTTP] *Hypertext Transfer Protocol*. 2019. URL: https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol.