VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Síťové aplikace a správa sítí – projekt

HTTP nástenka

18. novembra 2019 Dominik Juriga, xjurig00

Obsah

1 Abstrakt

Cieľom tejto práce je vytvorenie servera a klienta, ktorí na komunikáciu používajú protokol HTTP a adresový priestor IPv4. Táto komunikáciu umožňuje správu násteniek a ich príslušných textových príspevkov.

Táto dokumentácia popisuje teoretický základ, implementačné detaily a obmedzenia implementácie. Práca je vypracovaná v jazyku C++, s použítím štandardu C++11. Server je *client agnostic*, čiže prijíma a spracováva HTTP správy podľa štandardu RFC2616 [2], z ľubovoľného klienta.

Výstupom práce je dvojica programov *isaserver* a *isaclient* a príslušná dokumentácia.

2 Úvod

Manuál k programu popisuje použitie, časti implementácie a použité štandardy. Je napísaný v slovenskom jazyku. Projekt taktiež obsahuje README súbor, v ktorom je skrátene popísané používanie projektu.

Server HTTP nástenky poskytuje API (Application Programming Interface) rozhranie pre vytváranie a správu násteniek, ktoré obsahujú textové príspevky. Toto rozhranie má viacero ciest, ktoré môžu byť kontaktované z ľubovoľného klienta. Tieto cesty budú v tomto texte označované ako *endpointy*. Endpointy sú popísané v sekcii 4.6. Nástenka je popísaná jej unikátnym názvom, ktorý môže obsahovať malé alebo veľké písmená abecedy a čísla. Textový príspevok obsahuje vlastný text a má svoj identifikátor, ktorý je v rámci nástenky unikátny. Identifikátor príspevku značí jeho poradie v nástenke a začína od 1, nie je teda konečný a môže sa počas behu meniť.

Klient poskytuje rozhranie nad dostupnou funkcionalitou servera. Umožňuje vykonávať CRUD (**C**reate, **R**ead, **U**pdate, **D**elete) akcie nad nástenkami a ich príspevkami.

2.1 Štruktúra projektového balíka

Po rozbalení do užívateľom vybraného adresára vznikne následujúca súborová štruktúra:

```
user@dev: folder$ tree .
.
|-- board.hpp
|-- http_message.hpp
|-- isaclient.cpp
|-- isaserver.cpp
|-- Makefile
|-- manual.pdf
|-- README
|-- target.hpp
```

2.2 HTTP

HTTP (HyperText Transfer Protocol) je protokol aplikačnej vrstvy, ktorý umožňuje jednoduchú komunikáciu a prenos dát cez internet. V projekte sa využíva verzia `HTTP/1.1`, ktorá je popísaná štandardom RFC 2616. Tento protokol pracuje na princípe modelu *požiadavka - odpoveď*. Klient (väčšinou osobný počítač) zasiela *požiadavku* na server, ktorý ju následne spracováva a posiela späť dáta vo forme *odpovede*. HTTP správy sa skladajú z hlavičky a dát. Hlavička a dáta sú oddelené prázdnym riadkom. Hlavička obsahuje rôzne hlavičkové polia, ktoré popisujú napríklad dĺžku dát, typ odpovede, ...

2.2.1 HTTP požiadavka

Hlavička HTTP požiadavky je definovaná najmä jej metódou. Metóda umožní serveru rozpoznať, o akú akciu klient žiada. V tomto projekte boli použité metódy `GET`, `POST`, `PUT` a `DELETE`. Následuje adresa endpointu a verzia HTTP. Príklad hlavičky je nasledovný:

```
POST /board/myBoard HTTP/1.1
Host: 127.0.0.1
Content-Type: text/plain
Content-Length: 22

This is my first post.
```

2.2.2 HTTP odpoveď

Hlavička HTTP odpovede je definovaná najmä odpoveďovým kódom. Ten určuje, či server vykonal alebo nevykonal požadovanú akciu. V tomto projekte je možné očakávať následujúce odpoveďové kódy:
`200 OK`: server vykonal požadovanú akciu.
`201 Created`: server vytvoril požadovaný objekt.
`400 Bad Request`: pri POST/PUT požiadavku nad príspevkom bola dĺžka obsahu 0.
`404 Not Found`: požadovaný objekt neexistuje alebo iná chyba.
`409 Conflict`: pri vytváraní objektu bol objavený zhodný objekt.
Príklad HTTP odpovede je nasledovný:

```
HTTP/1.1 200 OK
Content-Type: text/plain
Content-Length: 32

myFirstBoard
1. This is my post.
```

3 Preklad a spustenie projektu

Po rozbalení zdrojového tar archívu do vybranej zložky je potrebné projekt preložiť. Archív obsahuje súbor Makefile, v ktorom sú špecifikované prepínače, s ktorými sa programy prekladajú. Na preklad je možné využiť príkaz `make`, ktorý vytvorí spustiteľné binárne súbory. Taktiež je možné preložiť jednotlivé časti projektu pomocou dvojice príkazov `make isaclient` a `make isaserver`. Taktiež je možné použiť príkaz `make clean`, ktorý zmaže preložené súbory.

Preložené súbory je možné následne spustiť. Pre optimálnu a bezproblémovú činnosť s paketami je odporúčané spúšťať projekt s root právami. Dostupné formáty spustenia:

```
$ sudo ./isaserver [-p port_num] (-h)
$ sudo ./isaclient [-H host_name] [-p port_num] [command] (-h)
```

Argumenty uvedené v hranatých zátvorkách sú povinné. Argument -h je voliteľný a vypíše na štandardný výstup nápovedu a následne ukončí program.

`-H host_name`: Špecifikuje názov alebo IPv4 adresu zariadenia, na ktorom je spustený server.
`-p port_num`: Špecifikuje číslo portu, z intervalu 1024-65535, na ktorom je spustený server.
`command`: Príkaz, ktorý má byť vykonaný. Príkazy sú popísané v sekcii 3.1.

3.1 Možné príkazy *command*

`boards`: Vypíše všetky nástenky a vráti kód 200. V prípade že žiadna neexistuje, vracia sa kód 404.
`board add name`: Pridá nástenku s názvom `name` a vracia kód 201. Pokiaľ takáto nástenka už existuje, vracia sa kód 409.
`board delete name`: Zmaže nástenku s názvom `name` a vracia kód 200. V prípade že takáto nástenka neexistuje vracia sa kód 404.
`board list name`: Vypíše príspevky na nástenke s názvom `name`. Pokiaľ takáto nástenka neexistuje, alebo nemá žiadne príspevky, vracia kód 404.
`item add name "payload"`: Pridá príspevok na nástenku s názvom `name` a textom `payload` a vracia kód 201. Payload sa uvádza v dvojitých zátvorkách. Pokiaľ takáto nástenka neexistuje, vracia kód 404.
`item delete name id`: Zmaže príspevok na nástenke `name` s id `id` a vracia kód 200. Pokiaľ neexistuje takáto nástenka alebo príspevok, vracia kód 404.
`item update name id "payload"`: Aktualizuje príspevok na nástenke `name` s id `id` na text `payload` a vracia kód 200. Payload sa uvádza v dvojitých zátvorkách. Pokiaľ neexistuje nástenka alebo príspevok, vracia kód 404.

3.2 Príklad spustenia

```
$ sudo ./isaserver -p 8000
$ sudo ./isaclient -H localhost -p 8000 boards
$ sudo ./isaclient -H localhost -p 8000 item add myBoard "My first post"
```

4 Implementácia a implementačné obmedzenia

4.1 Dátové štruktúry

Pre potreby projektu bolo definovaných niekoľko dátových štruktúr. Ich popis je následovný:

4.1.1 http_msg

`http_msg` predstavuje HTTP správu. Obsahuje informácie o cieľovom zariadení a požadovaných akciách. Nachádza sa v súbore `http_message.hpp`, spolu s rôznymi HTTP konštantami, ktoré sú používané naprieč celým projektom.

- string `METHOD` - HTTP Metóda
- string `ADDRESS` - Adresa endpointu
- string `VERSION` - Verzia HTTP
- string `HOST` - Názov cieľového zariadenia
- string `CONTENT_TYPE` - Typ obsahu
- string `PAYLOAD` - Obsah správy
- string `RESPONSE_TEXT` - Textový popis HTTP odpoveďového kódu
- int `CONTENT_LENGTH` - Dĺžka obsahu
- int `RESPONSE` - HTTP Odpoveďový kód

4.1.2 post

`post` predstavuje jeden príspevok.

- int `id` - ID príspevku na jeho nástenke
- string `text` - Textový obsah príspevku

4.1.3 board

`board` predstavuje jednu nástenku.

- string `name` - Názov nástenky
- list<post> `posts` - Zoznam obsahujúci príspevky
- int `last_id` - Prvé dostupné ID na nástenke

4.1.4 target

`target` predstavuje informácie pre odoslanie požiadavky.

- int `port` - Port, na ktorom cieľové zariadenie očakáva spojenie
- http_msg `request` - HTTP požiadavka, ktorá bude odoslaná

4.2 Kontrola argumentov

Správnosť argumentov sa overuje pomocou funkcie `continue_program`. Funkcia overí či sú argumenty správe zadané, v opačnom prípade ukončí program s chybovou hláškou.

4.3 Spracovávanie argumentov

V prípade, že funkcia `continue_program` vrátila hodnotu `true`, volá sa funkcia `parse_arguments`, ktorá u serveru vracia číslo portu, respektíve štruktúru `target` u klienta. Táto štruktúra obsahuje údaje o cieľovom zariadení, na ktoré sa bude odosielať HTTP požiadavka a príkaz, ktorý má byť vykonaný.

4.3.1 Preklad názvu hosta na adresu

Pri spracovávaní argumentov sa meno hosta prevádza na IPv4 adresu pomocou funkcie `gethostbyname()`. Pokiaľ je host zadaný vo forme IPv4 adresy, nevykonáva sa žiadny preklad, a adresa sa jednoducho skopíruje do štruktúry `hostent` vracanej funkciou `gethostbyname()`. V opačnom prípade je vykonané vyhľadávanie v enviromentálnej premennej `HOSTALIASES`, z ktorej je následne vrátená IPv4 adresa, alebo je program ukončený chybovou hláškou.

4.4 Ukladanie dát na serveri

Dáta sa na strane servera ukladajú do dynamických dátových štruktúr typu *list*. Príspevky sú definované ako štruktúra `post`. Táto štruktúra obsahuje položky `id` a `text`, ktoré uchovávajú id a textový obsah príspevku. Nástenky sú definovaný ako štruktúra `board`. Štruktúra `board` obsahuje položky `name`, `last_id`, `posts`, ktoré uchovávajú názov, prvé voľné ID príspevku a zoznam príspevkov. Tieto štruktúry sú uložené v súbore *board.hpp*.

4.5 Princíp funkčnosti programov

4.5.1 Klient

Na základe údajov získaných z argumentov volania programu sa vo funkcii `parse_arguments` volajú funkcie podľa jednotlivých HTTP metód, nazvané `create_[METHOD]_request()`, pričom [METHOD] označuje žiadanú HTTP metódu. V týchto funkciách sa vytvorí premenná typu `http_msg`, v ktorej sa nastavia príslušné údaje, a je vrátená funkcii `parse_arguments()`, ktorá ju ukladá to štruktúry `target`. Následne sa nastavia potrebné premenné štruktúry `target`, ako napríklad host adresa, verzia HTTP,... Celý proces tvorby HTTP správy je zakončený vstupom do funkcie `generate_request_string()`, ktorá túto štruktúru spracuje a dosadí jej hodnoty do stringu, ktorý predstavuje HTTP požiadavku.

Požiadavka sa odošle pomocou funkcie `send()` [1], a následne klient očakáva odpoveď, skrz nekonečný cyklus *while(true)*, v ktorom sa pomocou funkcie `recv()` načítava správa po 1024 bajtoch. V prvej prijatej správe sa vyhľadáva reťazec *Content-Length*. Pokiaľ bol nájdený, prijíma sa dovtedy, pokým nebola dosiahnutá dĺžka hlavičky a obsahu. Po prijatí odpovede je hlavička HTTP správy vytlačená na štandardný chybový výstup, a obsah HTTP správy vytlačený na štandardný výstup.

4.5.2 Server

Server vo funkcii `server_listen()` vytvorí socket a príslušne ho nastaví tak, aby umožňoval komunikáciu pomocou protokolu TCP [1]. Následne aktívne čaká na spojenia vo funkcii `await_connections()`, na porte definovanom pri spustení programu. Po prijatí spojenia zo soketu načíta správu o maximálnej veľkosti 16384 bajtov (bližší popis v sekcii 4.6.1). Následuje spracovanie správy vo funkcii `create_response()`, pri ktorom sa správa rozdelí na jednotlivé tokeny. Na základe tokenov sa zistí požadovaná HTTP metóda, teda akcia, ktorá bude vykonaná. Vytvorí sa premenná typu `http_msg`. Prejde sa do funkcie `verify_[METHOD]`, kde [METHOD] je požadovaná HTTP metóda. Funkcia overí, či boli uvedené všetky požadované informácie, vykoná ich a vráti návratový HTTP kód, ktorý je priradený do štruktúry `http_msg`, spolu s ostatnými potrebnými informáciami. Tvorba je zakončena vstupom do funkcie `generate_response_string()`, ktorá túto štruktúru spracuje a dosadí jej hodnoty do stringu, ktorý predstavuje HTTP odpoveď. Tento string je odoslaný ku klientovi a spojenie je ukončené. Server následne očakáva ďalšie spojenie.

4.6 Server a jeho endpointy

Server pracuje ako API (Application Programming interface), čo znamená, že poskytuje preddefinované endpointy. Pokiaľ je tento preddefinovaný endpoint kontaktovaný akýmkoľvek klientom, vždy mu budú odoslané dáta v preddefinovanom formáte. V našom prípade sa jedná o HTTP odpoveď. Táto architektúra umožňuje vykonávať CRUD akcie nad nástenkami a ich príspevkami.
Dostupné endpointy sú:
`GET /boards`: Vráti zoznam existujúcich násteniek.
`POST /boards/name`: Vytvorí nástenku `name`.
`DELETE /boards/name`: Zmaže nástenku `name`.
`GET /board/name`: Vráti zoznam príspevkov na nástenke `name`.
`POST /board/name`: Pridá príspevok na nástenku `name`.
`PUT /board/name/id`: Upraví text príspevku s id `id` na nástenke `name`.
`DELETE /board/name/id`: Zmaže príspevok s id `id` na nástenke `name`.

4.6.1 Implementačné obmedzenia servera

1. Obmedzenie dĺžky prenosu na strane serveru na 16KiB:

   Klient číta správy zo soketu pomocou nekonečného cyklu *while(true)*, pokým nenačíta celú správu. Pri pokuse o implementáciu podobného prístupu na strane servera prišlo však k rôznym problémom.

Príkladom takého problému bola nekompatibilita s prehliadačom pri testovaní cez sieť, ktorý nedostával odpovede, nakoľko sa server zacyklil.

Veľkosť som zvolil na základe kompromisu medzi použiteľnosťou a výpočetnými nárokmi. Pre jednu správu to znamená viac ako 8 normostrán, vrátane hlavičiek komunikácie. Pre potrebné zdroje to znamená pri dnešnej bežne dostupnej konfigurácii počítačov menej ako jedna tisícina percenta dostupnej pamäte.

5 Testovanie

Pre otestovanie prekladu a funkčnosti projektu bol využitý server `merlin.fit.vutbr.cz`. Testovanie funkčnosti prebehlo taktiež pomocou aplikácie Postman, ktorá umožňuje posielať ľubovoľné HTTP správy. Funkcionalita servera bola v obmedzenej miere otestovaná aj na prehliadači Google Chrome v stabilnej verzii 78.0.3904.97. Ten však nedokáže nasimulovať odosielanie určitých HTTP metód a užívateľom definovaných HTTP správ.

Testovanie pozostávalo z overenia funkčnosti všetkých prvkov. Boli vykonané následovné kroky (v chronologickom poradí)

1. Vypísanie násteniek (žiadna neexistuje) - odpoveď 404
2. Pridanie nástenky *myBoard* - odpoveď 201
3. Vypísanie násteniek - odpoveď 200
4. Pridanie tej istej nástenky - odpoveď 409
5. Pridanie príspevku na neexistujúcu nástenku - odpoveď 404
6. Pridanie príspevku na *myBoard*, bez obsahu - odpoveď 400
7. Pridanie príspevku na nástenku *myBoard* - odpoveď 201
8. Pridanie príspevku na nástenku *myBoard* - odpoveď 201
9. Pridanie príspevku na nástenku *myBoard* - odpoveď 201
10. Vypísanie príspevkov na neexistujúcej nástenke - odpoveď 404
11. Vypísanie príspevkov na nástenke *myBoard* - odpoveď 200
12. Úprava prvého príspevku na nástenke *myBoard*, bez obsahu - odpoveď 400
13. Úprava neexistujúceho príspevku na nástenke *myBoard* - odpoveď 404
14. Úprava príspevku na neexistujúcej nástenke - odpoveď 404
15. Úprava prvého príspevku na nástenke *myBoard* - odpoveď 200
16. Zmazanie neexistujúceho príspevku na nástenke *myBoard* - odpoveď 404
17. Zmazanie prvého príspevku na nástenke *myBoard* - odpoveď 200

Po vykonaní by mala existovať jedna nástenka s názvom *myBoard*, ktorá obsahuje dva príspevky.
Výpis príspekov na nástenke *myBoard*:

```
HTTP/1.1 200 OK
Content-Type: text/plain
Content-Length: 45
1. Viacriadkovy
prispevok
2. Treti prispevok
```

Citácie

[1] S. Nitsch. *Sockets Tutorial*. [online]. [vid. 2019-10-25]. 2018. URL: <http://www.linuxhowtos.org/C_C++/socket.htm>.

[2] J. Mogul R. Fielding J. Gettys. *Hypertext Transfer Protocol – HTTP/1.1*. [online]. [vid. 2019-10-25]. jún 1999. URL: <https://tools.ietf.org/html/rfc2616>.