Vysoké učení technické v Brně
Fakulta informačních technologií

Síťové aplikace a správa sítí
Http nástěnka

18.11.2019 Tomáš Hampl (xhampl10)

1 Obsah

2 Úvod

Tento program byl napsán jako projekt do předmětu *Síťové aplikace a správa sítí*. Cílem projektu bylo implementovat v C/C++ stranu serveru a klienta pro http nástěnku. Strana serveru má za úkol spravovat API a odpovídat na požadavky od klienta. Strana klienta má za úkol vytvářet požadavky pro server na základě předaných argumentů a tisknout odpovědi serveru.

3 Návrh aplikace

3.1 Server

Zahájení běhu serveru spočívá ve zpracování argumentů, kde získá port, na kterém má naslouchat. Následně vytvoří *socket* a začne čekat na zaslání požadavku. Při přijmutí požadavku ověří velikost obsahu a dále zpracovává požadavek. Vytáhne si metodu a adresu, podle kterých zjistí, kterou funkci API má spustit. Následně provede požadované úkony a odpovídá klientovi. Server dále čeká na další požadavek.

3.2 Klient

Při spuštění, klient musí zkontrolovat argumenty a podle nich vytváří požadavek. Následně vytvoří *socket*, přes který odešle požadavek. Klient poté čeká na odpověď, kterou vypíše a ukončí se.

4 Popis implementace

Většina použitých knihoven je pro jazyk C, tudíž C++ bylo využito jen na usnadnění práce, jako například textové řetězce, vektory a práce s objekty.

4.1 Parsování argumentů

4.1.1 Server

Na zpracování argumentů byla použita funkce *getopt* implementována pomocí konstrukce *switch*.

4.1.2 Klient

Zpracování argumentů bylo vzhledem ke kombinaci jednoduchých argumentů s celými slovy použito manuální procházení cyklem. Tato část byla implementována v C, takže se využívá knihovny *string.h*.

4.2 Posílání a přijímání požadavků

Klient vytvoří požadavek na základě argumentů programu do formátu HTTP/1.1 a pomocí *socketu* ho odesílá serveru. Server při přijmutí požadavku zkontroluje, zda zapsal celý požadavek pomocí kontroly délky obsahu. Následně rozdělí hlavičku a tělo požadavku a začne zpracovávat hlavičku. Prvotně jsem používal funkce *regex_match* a *regex_search* z knihovny *regex,* ale na Merlinovi mi to občas vyvolalo *segmentation fault* a tak jsem se nakonec rozhodl zpracovávat hlavičku pomocí vektorů a iterátorů z knihovny *vector.*[1] Následně zavolá příslušnou funkci z API, která se vykoná a výsledek je odeslán zpět klientovi.

[1] To je důvod, proč mi server nedokáže odpovědět na požadavek vyslaný pomocí *curl,* ale dokáže odpovědět na požadavek poslaný pomocí *telnet*.[1]

4.3 API

Rozhraní je implementováno jako funkce nad strukturami *Node* a *Board,* kde *Board* představuje list nástěnek a *Node* představuje list jednotlivých příspěvků.

Rozhraní má tyto funkce:

- **GET /boards** – vypíše seznam dostupných nástěnek, jedna na řádek
- **GET /board/name** – zobrazí obsah nástěnky
- **POST /boards/name** – vytvoří prázdnou nástěnku s názvem *name*
- **POST /board/name content** – vytvoří příspěvek s obsahem *content* v nástěnce *name*
- **DELETE /boards/name** – smaže nástěnku *name* a všechen její obsah
- **DELETE /board/name/id** – smaže příspěvek z nástěnky *name* na indexu *id*
- **PUT /board/name/id content** – přepíše obsah příspěvku v nástěnce *name* na indexu *id contentem*

4.4 Požadavek

Požadavek je implementován následovně:[2]

Method\t**Address**\t**HTTP/1.1**\r\n

Host\s**:**\s**Host**\r\n

Content-Type\s**:**\s**Type**\r\n

Content-Length\s:\s**Length**\r\n\r\n

Content

5 Návod na použití

5.1 Server

Popis spuštění strany serveru:

./isaserver -p <číslo portu> [-h]

Argument „-p“ představuje číslo portu a je povinný.

Argument „-h“ vyvolá nápovědu k programu.

5.2 Klient

Popis spuštění strany klienta:

./isaclient [-h] -H <host> -p <port> <příkaz>

Argument „-h“ je nepovinný a vyvolá nápovědu k programu.

Argument „-H“ je povinný a představuje IP adresu nebo název destinace.

Argument „-p“ je povinný a představuje port.

[2] K popisu implementace jsou využity reguálnír výrazy: \s – bílý znak, \t – tabulátor \r – znak návratu \n – nový řádek

Argument „příkaz“ představuje příkaz, který se zašle na server. Může mít několik variant:

- **Boards** – GET /boards
- **Board add <name>** – POST /board/<name>
- **Board delete <name>** - DELETE /board/<name>
- **Board list <name>** - GET /board/<name>
- **Item add <name> <content>** - POST /board/<name> <content>
- **Item delete <name> <id>** - DELETE /board/<name>/<id>
- **Item update <name> <id> <content>** - PUT /board/<name>/<id> <content>

Content má omezení na znaky [a-zA-Z0-9].

5.3 Příklad spuštění

5.3.1 Server

./isaserver -p 2020

5.3.2 Klient

./isaclient -H 127.0.0.1 -p 2020 boards

./isaclient -H localhost -p 2030 item add prace popelar

./isaclient -H merlin.fit.vutbr.cz -p 2222 item add prace „udelat ISA projekt“