Vysoké učení technické v Brně
Fakulta informačních technologií

Dokumentácia k projektu ISA programování síťové služby

HTTP nástenka

Autor: Peter Horňák (xhorna14)

Table of Contents

1. Uvod

Cieľom projektu bolo vytvoriť 2 programy v jazyku C++, ktoré medzi sebou komunikujú pomocou socketov, protokolov TCP, HTTP verzie 1.1 a rozhrania nad ním. Prvá časť projektu sa zaoberá implementáciou serveru, ktorý spracováva požiadavky od klienta a ukladá si obsah dát tzv. násteniek. Klient posiela požiadavky na server a vypisuje odpovede na štandardný výstup.

2. Zadanie

2.1. Nástenka

Nástenkou sa rozumie usporiadaný zoznam textových (ASCII) príspevkov. Každý príspevok ma id a textový obsah, ktorý môže byť viacriadkový. Id nie je permanentné, korešponduje aktuálnej pozícii v zozname. Operácie nad nástenkou by nemali meniť poradie príspevkov. Názov nástenky môže obsahovať znaky a-z, A-Z a 0-9. Formát zobrazenia nástenky je:

```
[NÁZOV]
1. PRVÝ PRÍSPEVOK.
2. DRUHÝ PRÍSPEVOK.
...
N. N-TÝ (POSLEDNÝ PRÍSPEVOK).
```

2.2. HTTP API

API je definované nasledovne:

- `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. Príspevok je vložený 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.

GET, PUT a DELETE vracajú pri úspechu kód 200, POST 201. Pokiaľ žiadaná nástenka alebo príspevok neexistujú, vracia sa kód 404. Pokiaľ je snaha vytvoriť nástenku s názvom ktorý už existuje, vracia sa kód 409. Pokiaľ POST alebo PUT nad príspevkom majú Content-Length = 0, vracia sa kód 400. Na všetky iné akcie reaguje server odpoveďou 404.

3. Návrh implementácie

Server a klient sú rozdelené do dvoch súborov. Z dôvodu nie prílišnej komplexnosti jednotlivých zdrojových kódov som sa rozhodol tieto moduly rozdeľovať do viacerých súborov. Pri implementácií boli využité triedy, pre lepšiu štruktúru jednotlivých častí projektu. Moduly uskutočňujú komunikáciu pomocou protokolu HTTP, ktorého detaily sú popísané v RFC 2616, priebeh komunikácie, ktorý nie je popísaný v tomto dokumente je v súlade s vyššie spomenutým dokumentom. V tejto kapitole je stručne popísaný postup implementácie oboch modulov.

4. Server

Zdrojový kód sa nachádza v súbore ISASERVER.CPP, preložiť ho je možné pomocou príkazu MAKE. Testovanie prebehlo na referenčnom stroji MERLIN, na ktorom v tom čase bežal operačný systém CentOS verzie 7.

4.1. Spustenie

Spustiť program je možné pomocou príkazu:
`./ISASERVER -P PORT [-H]`
Kde parametre znamenajú:
`-H` – Vypíše pomocnú správu ako program spustiť. Tento parameter je voliteľný .
`-P PORT` – Určuje číslo portu na ktorom bude daný server počúvať.

4.2. Implementácia

Po spustení, program spracuje argumenty a skontroluje ich správnosť. V prípade neúspechu vracia návratový kód -1. Následne vytvorí socket v neblokujúcom móde na danom porte. Program začne na tomto porte počúvať a pomocou funkcie POLL(), čaká na požiadavku od klienta v nekonečnom cykle. To nám zaručuje, že server po prijatí požiadavku a následnom obslúžení, bude opäť čakať na klienta. Následne sa vytvorí spojenie medzi klientom a serverom. Server číta správy v častiach o veľkosti 4096 bytov až pokiaľ nenačíta správu s veľkosťou 0, čo signalizuje prázdnu správu. Na základe toho, je vytvorená inštancia triedy HEADER, ktorá spracuje prvý riadok správy, ktorý musí zodpovedať nasledujúcemu regulárnemu výrazu:

```
"(GET|POST|PUT|DELETE) */([a-zA-Z0-9]*)(/([a-zA-Z0-9]*)((\\d*))?)? *(HTTP/1.1|0)"
```

Tento regulárny výraz kontroluje požiadavku od klienta, či zodpovedá správnemu formátu. Spracovanie spočíva v zistení ktorému príkazu rozhrania požiadavka zodpovedá. V prípade, že klient chce pridávať alebo upravovať príspevok na nástenke, musí byť tak isto obsiahnutá hlavička CONTENT-LENGTH, ktorá značí veľkosť správy v bytoch. Inak server odpovie chybovou hláškou. Ostatné hlavičky sú odignorované, čo však znamená, že server pracuje s každou požiadavkou ako keby bola hlavička CONTENT-TYPE nastavená na hodnotu PLAIN/TEXT. Potom je oddelená hlavička od prípadného tela správy. Telo je potom uložené v objekte BODY. Následne je vytvorená inštancia triedy HANDLER, ktorá jednotlivé požiadavky obslúži. Jedným z jej atribútov, ktorý stojí za zmienenie je DASHBOARD_LIST, ktorý je typu MAP. Tento atribút obsahuje objekty typu DASHBOARD namapované na názov konkrétnych nástenieк, ktorý je typu STRING. Trieda DASHBOARD si ukladá svoje príspevky v atribúte typu VECTOR<STRING>, čo zaručuje vlastné ID každého príspevku pomocou ich indexov. Po vykonaní danej rutiny, v prípade neúspechu server odpovedá chybovou správou, v opačnom prípade klientovi pošle správu o úspechu a poprípade dáta o ktoré klient požiadal. Server môže byť ukončený systémovým volaním. V prípade internej chyby alebo neúspechu, program končí s návratovou hodnotou -1 alebo s hodnotou, ktorú vráti funkcia, ktorá chybu vyvolala.

5. Klient

Zdrojový kód sa nachádza v súbore ISACLIENT.CPP, preložiť ho je možné pomocou príkazu MAKE. Testovanie prebehlo na referenčnom stroji MERLIN, na ktorom v tom čase bežal operačný systém CentOS verzie 7.

5.1. Spustenie

Spustiť program je možné pomocou príkazu:

```
./ISACLIENT -H HOST -P PORT "COMMAND" [-H]
```

Kde parametre znamenajú:
-H – Vypíše pomocnú správu ako program spustiť. Tento parameter je voliteľný .
-H – Určuje IPv4 adresu na ktorej beží server.
-P PORT – Určuje číslo portu na ktorom bude daný server počúvať.
COMMAND – Určuje typ požiadavky ktorý sa má poslať na server. Môže nadobúdať jeden z nasledujúcich tvarov a zodpovedá nasledujúcemu tvaru požiadavku :

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>

Pričom <name> značí názov nástenky a <id> číslo príspevku.

5.2. Implementácia

Program na začiatku vytvorí inštanciu triedy PARSER, ktorá skontroluje správnosť parametrov a na ich základe si uloží všetky potrebné dáta, ktoré z nich vie získať. Tak isto sa vypočíta dĺžka správy, ktorá má byť odoslaná na server a vytvorí sa hlavička CONTENT-LENGTH s priradenou dĺžkou. Následne sa vytvorí socket, ktorý sa pokúsi spojiť so serverom. V prípade neúspechu program čaká dokedy to bude možné. V opačnom prípade pošle požiadavku na server a počká na jeho odpoveď. Tak isto aj v tomto prípade bude klient čakať, pokiaľ mu server odpovie. Následne klient prečíta odpoveď po častiach o veľkosti 4096 bytov. Následne oddelí hlavičky od prípadného tela správy a vypíše hlavičky na štandardný chybový výstup a telo odpovede na štandardný výstup. V prípade neúspechu alebo chyby, sa program ukončí s návratovou hodnotou -1.

6. LITERATURA

[1] RFC2616 Hypertext Transfer Protocol -- HTTP/1.1. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee. June 1999.(Format: TXT, PS, PDF, HTML) (Obsoletes RFC2068) (Obsoleted by RFC7230, RFC7231, RFC7232, RFC7233, RFC7234, RFC7235) (Updated by RFC2817, RFC5785, RFC6266, RFC6585) (Status: DRAFT STANDARD) (DOI: 10.17487/RFC2616)

[2] J. F. Kurose and K. W. Ross.Computer Networking: A Top-Down Approach Featuring the Internet.Addison-Wesley, 2nd edition, 2003.

[3] W. Richard Stevens, Bill Fenner, and Andrew M. Rudoff.UNIX Network Programming. The Sockets Networking API.Addison-Wesley, 3rd edition, 2004

[4] Example: Nonblocking I/O and select() Dostupné z: https://www.ibm.com/support/knowledgecenter/ssw_ibm_i_72/rzab6/xnonblock.htm