VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Dokumentácia k projektu do predmetu ISA
(Sít'ové aplikace a správa sítí)

HTTP nástenka

18. listopadu 2019

Kristián Liščinský (xlisci01)

Obsah

1 Úvod

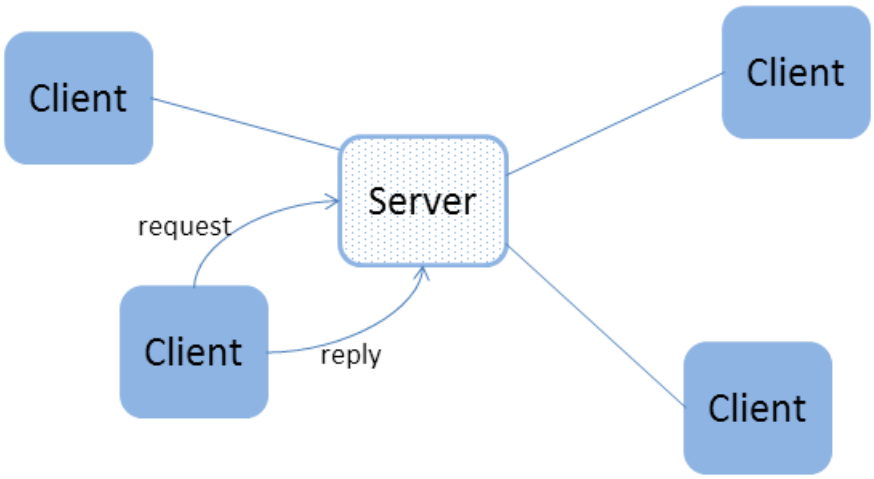
Úlohou projektu bolo navrhnúť, implementovať a otestovať aplikáciu na manipulovanie s nástenkami pomocou HTTP API. V nasledujúcich kapitolách sú opísané dôležité časti projektu. V kapitole 2 prebieha úvod do problematiky, ozrejmenie nevyhnutných pojmov. Kapitola 3 sa zaoberá implementáciou programu. Kapitola 4 poskytuje prehľad nad používaním programu. V kapitole 5 sú uvedené základné informácie o programe. Posledná kapitola zahrňuje získané vedomosti a skúsenosti z projektu.

2 Súhrn pojmov

Táto kapitola obsahuje vysvetlenie jednotlivých pojmov súviiacich s projektom a priblíženie problematiky týkajúcej sa projektu.

2.1 Klient server architektúra

Štandardná schéma komunikácie medzi dvoma procesami sa nazýva model klient - server. Klient i server sú aplikacné procesy, ktoré komunikjú cez sieťové rozhranie. Môže ísť aj o procesy bežiace na rovnakom počítači. Základnými činnosťami klienta je posielanie žiadostí o nejakú sieťovú službu. Server čaká na prichádzajúce požiadavky, príjma ich, spracováva a posiela späť odpoveď.



Obrázek 1: Komunikácia klient server

Dôležité je, že spracovanie požiadavkov prebieha výhradne na strane servra. Klient iba predáva požiadavky a zobrazuje odpoveď. U modelu klient - server komunikáciu obvykle začína klient. Server beží v nekonečnej smyčke. Ukončenie servera je viazané na SIGINT signál. Komunikácia medzi serverom a klientom je popísaná tzv. protokolom. Protokol je súbor synktatických a sémantických pravidiel určujúcich výmenu informácie medzi aspoň dvoma entitami.

2.2 HTTP

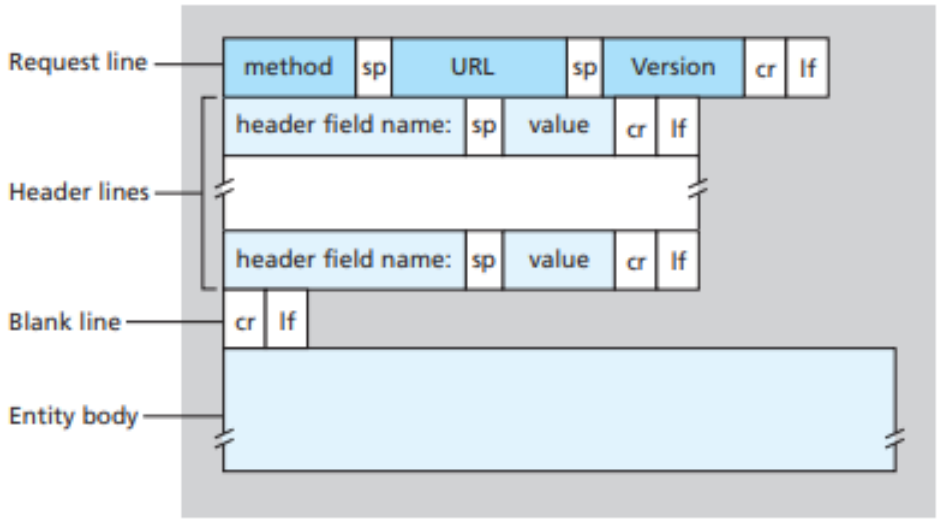
Protokol **HTTP** (Hypertext Transfer Protocol) je jeden z najdôležitejších protokolov Internetu pre prenos dát medzi serverom a klientom. Klienti HTTP predstavujú väčšinou webové prehliadače (browsery) a HTTP servery sú väčšinou webové servery. Protokol HTTP komunikuje na porte 80 TCP protokolu, pri zadávaní adresy vo formátu URL je však možno zadať aj iný port. Naviac protokol HTTP nie je závislý na protokole TCP a môže pracovat aj s inými spoľahlivými protokolmi. Vývoj HTTP koordinovalo World Wide Web Consortium a pracovné skupiny Internet Engineering Task Force, čím vytvorili sadu dokumentov RFC, predovšetkým **RFC 2616** definujúci HTTP/1.1, dnes používanú verziu HTTP. Klient zvyčajne začne požiadavku nadviazaním TCP spojenia na určenom porte vzdialeného stroja (štandardne port 80). HTTP server počúvajúci na danom porte čaká, kým klient pošle reťazec s požiadavkou[1].

2.3 HTTP správa

HTTP správy pozostávajú z HTTP požiadavkov od klienta k servru a HTTP odpovedí od servru ku klientovi. Žiadosť aj odpoveď používajú všeobecný formát správy **RFC 822**. Oba typy správ sa skladajú z počiatočného riadku, nula alebo viac polí hlavičiek, prázdneho riadka označujúceho koniec polia hlavičiek a prípadne z tela správy.

2.4 HTTP požiadavok (HTTP Request)

Správa s požiadavkou od klienta na server má nasledujúcu štruktúru:



Obrázek 2: HTTP request

2.4.1 Request line

Request line, alebo riadok so žiadosťou pozostáva z troch častí. Začína názvom metódy (napr. GET, POST..) za ktorým nasleduje identifikátor URI a verzia protokolu (v našom prípade sa jednalo o

verziu HTTP/1.1). Celé to končí postupnosťou znakov CR a LF. Jednotlivé prvky sú od seba oddelené medzerou (znak SP).

2.4.2 Hlavičky

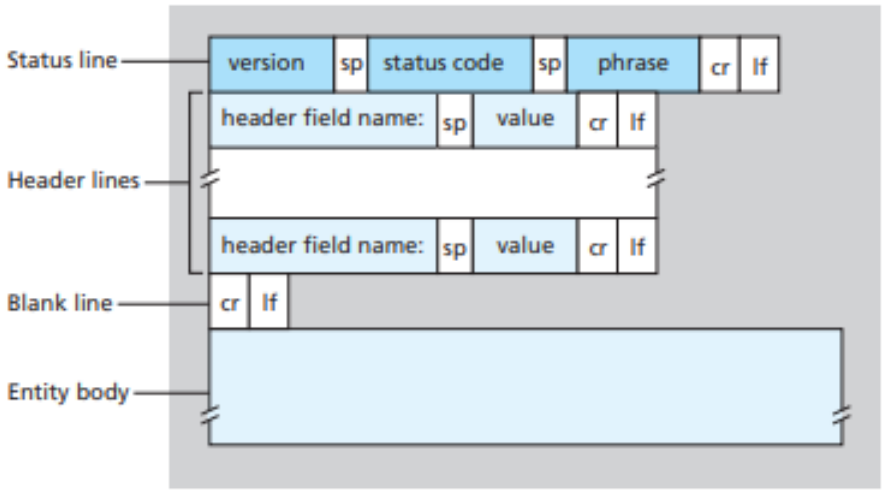
Ďalšiou časťou HTTP požiadavku je ľubovoľný počet hlavičiek, ktoré umožňujú klientovi odovzdať ďalšie informácie o žiadosti. Hlavičky končia postupnosťou znakov CR a LF, pričom rovnaká postupnosť znakov sa nachádza za poslednou hlavičkou hneď dvakrát, tkz. sekvencia identifikujúca koniec polia hlavičiek. Jednotlive hlavičky sa skladajú z názvu a hodnoty, pričom sú od seba oddelené dvojbodkou a medzerou.

2.4.3 Telo

Nakoniec nasleduje samotné telo správy, ktoré nie je u všetkych typov správ vyžadované.

2.5 HTTP odpoveď (HTTP Response)

Spáva s odpoveďou od servru ku klientovi má nasledujúcu štruktúru:



Obrázek 3: HTTP response

2.5.1 Status line

Status line pozostáva z troch častí. Začína verzia protokolu (v našom prípade sa jednalo o verziu HTTP/1.1) za ktorou nasleduje Status Code a Reason Phrase. Celé to končí postupnosťou znakov CR a LF. Jednotlivé prvky sú od seba oddelené medzerou (znak SP). **Status code** je trojčíselný kód, ktorým server vyjadruje to, ako sa mu podarilo vysporiadať zo žiadosťou. Prvé číslo kódu určuje triedu odpovede. Existuje 5 tried:

- **1xx:** Informačné

- **1xx:** Úspech
- **1xx:** Presmerovanie
- **1xx:** Chyba na strane klienta
- **1xx:** Chyba na strane servru

Reason Phrase je krátky textový popis status kódu. Je určený ľudských používateľov.

2.5.2 Hlavičky

Ďalšiou časťou HTTP odpovede je ľubovoľný počet hlavičiek, ktoré umožňujú servru odovzdať ďalšie informácie o odpovedi. Hlavičky končia postupnosťou znakov CR a LF, pričom rovnaká postupnosť znakov sa nachádza za poslednou hlavičkou hneď dvakrát, tkz. sekvencia identifikujúca koniec polia hlavičiek. Jednotlive hlavičky sa skladajú z názvu a hodnoty, pričom sú od seba oddelené dvojbodkou a medzerou.

2.5.3 Telo

Nakoniec nasleduje samotné telo správy, ktoré nie je u všetkych typov správ vyžadované[2].

3 Implementácia programu

Program je implementovaný v programovacom jazyku **C++** so štadndardom **C++11** a pozostáva z dvoch častí, z klienta a zo serveru.

3.1 Server

Server je implementovaný iteratívne, požiadavky spracováva jeden po druhom. Má jeden povinny argument a to je **číslo portu**. Server najprv načíta riadok s požiadavkom (request line) a hlavičky HTTP requestu od klienta. Ak sa mu podarí nájsť hlavičku **Content-Length**, znamená to, že server očakáva, že správa od klienta obsahuje aj telo o dĺžke, ktorú uvádza táto hlavička. Ak hlavička **Content-Length** chýba, server nepokračuje v čítaní správy a to ani v prípade, že obsahuje telo. Implicitne počíta s tým, že telo je v takomto prípade o veľkosti 0.

Po načítaní správy (HTTP requestu) prichádza k overovaniu, či sa jedná o validný formát requestu podľa protokolu HTTP. O to sa stará funkcia *parseWholeRequest(string request)*. Ďalej nasleduje vykonávanie žiadanej akcie, ktorú server získa z requestu. Potom dochádza na strane serveru k vytváraniu odpovede podľa protokolu HTTP (HTTP response). Ak server vyhodnotí request od klienta ako nevalidný podľa protokolu HTTP, odpovedá odpoveďou s kódom 404. U väčšiný prípadov server doplní do tela odpovede krátku informáciu o vykonanej akcii, napr. po úspešnom updatovani položky číslo *x* v nástenke *y* odpoveď ponesie okrem návratového kódu 200 taktiež v tele správu, že sa táto akcia vykonala. Ako posledné na strane serveru dochádza v odoslaniu vytvorenej HTTP odpovede späť ku klientovi.

Server okrem prípadu, kedy je spustený s parametrom **-h** nevypisuje žiadne informácie na štandardný výstup STDOUT. Na štandardný chybový výstup STDERR vypisuje server informácie v prípade, ak dôjde k nejakej chybe behom vytvárania alebo priebehu sieťovej komunikácie (napr. chyba pri bindovaní, chyba pri odosielaní správy a podobne) a taktiež v prípade, ak je spustený s nesprávnymi argumentami. Na chyby týkajúce sa zlého formátu HTTP requestu reaguje HTTP odpoveďou s kódom 404. Server nevyžaduje v HTTP requeste žiadne hlavičky, v takomto prípade však považuje, že request nemá telo a je spracovávaný len prvý riadok (request line).

Na manipuláciu s nástenkami slúžia funkcie: *boards(), board_list(string name), board_add(string name), item_add(string name, string content), board_delete(string name), item_delete(string name, string id), item_update(string name, string id, string content)*.

Keďže sa od serveru nepožaduje perzistencia dát, tak na ukladanie nástenok a ich obsahuje je použitá globálna premenná, ktorá je dátovej štruktúry mapa, *map <string, list <string» zoznamNastenok*.

Server pri vytváraní HTTP odpovede doplní tieto hlavičky: **Date, Content-Length, Content-Type**.

3.2 Klient

Klient má povinné argumenty **port, hostname a požadovaný príkaz nad nástenkou**. V prípade, že sú argumenty zadané nesprávne klient je ukončený návratovým kódom **1** (EXIT_FAILURE) a na štandardný chybový výstup **STDERR** je vypísaný dôvod ukončenia. Ak sú arguemnty v poriadku,

tak na ich základe je pomocou funkcie *createHttpRequest()* vytvorená správa (HTTP request), ktorá je odoslaná na stranu servera. Po tomto nasleduje čakanie na odpoveď zo strany servera. Očakáva sa, že server zašle odpoveď podľa protokolu HTTP. Najprv klient načíta prvý riadok odpovede, tkz. status line a tiež hlavičky. Ak sa medzi hlavičkami nachádza hlavička **Content-Length**, klient načíta aj telo odpovede. Ak hlavička **Content-Length** nie je prítomná, klient predpokladá, že odpoveď od servera neobsahuje žiadne telo. Za tým nasleduje overovanie, či prijatá odpoveď odpovedá formátu, ktorý popisuje protokol HTTP. Na to slúži funkcia *parseHeaderAndStatusLine()*. V prípade, že HTTP odpoveď nie je v správnom formáte, je klient ukončený s návrátovým kódom **1** (EXIT_FAILURE) a na štandardný chybový výstup **STDERR** je vypísaný dôvod takéhoto ukončenia programu. Ak je HTTP odpoveď v správnom formáte, tak je jej prvý riadok (status line) spolu so všetkými prítomnými hlavičkami a tiež deliacim riadkom (\r \n), ktorý oddeľuje hlavičky od tela odpovede, prípadne, ak telo chýba ukončuje HTTP odpoveď vypísaný na štandardný chybový výstup **STDERR** a telo odpovede (ak existuje) je vypísané na štandardný výstup **STDOUT**.

Klient pri vytváraní HTTP requestu doplní tieto hlavičky: **Host, Content-Length, Content-Type**.

V prípade príkazov, ktoré vyžadujú ako posledný argument obsah (jedná sa o príkazy *item update* a *item add*), tak sa predpokladá, že obsah je zadaný ako jeden argument pomocou úvodzoviek, ale program pracuje tiež s verziou, že obsah je načítavaný ako všetky zvyšné argumenty programu. V prípade, ak chceme, aby obsah bol viacriadkový a obsahoval znak \n je potreba použiť tkz. ANSI C like strings notáciu a daný reťazec obaliť do $' '. [3]

4 Návod na použitie

Aplikácia sa skladá z dvoch časti, z **klienta** a zo **serveru**. Obe sa spúšťajú cez terminál.

4.1 Server:

Pri zadaní prepínača **-h** sa vypíše text o programe a jeho prepínačoch. V prípade neznámeho, či chybne použitého prepínača (nesprávna/chýbajúca hodnota prepínača) alebo pri akejkoľvek chybe v sieťovej komunikácií sa program ukončí a o probléme informuje používateľa správou na štandardný chybový výstup.

./isaserver -p <port>

port je číslo portu

Program sa ukončuje pomocou **Ctrl-C**, resp. pomocou príkazu **"kill -INT <pid>"** v termináli.

4.2 Klient

Pri zadaní prepínača **-h** sa vypíše text o programe a jeho prepínačoch. V prípade neznámeho, či chybne použitého prepínača (nesprávna/chýbajúca hodnota prepínača) alebo pri akejkoľvek chybe v sieťovej komunikácií sa program ukončí a o probléme informuje používateľa správou na štandardný chybový výstup.

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

<command> má nasledujúcu štruktúru:
boards – vráti zoznam dostupných násteniek, jednu na riadok
board add <name> – vytvorí prázdnu nástenku naznanú <name>
board delete <name> – zmaže nástenku <name> a všetok jej obsah
board list <name> – zobrazí obsah nástenky <name>
item add <name> <content> – vloží nový príspevok na koniec nástenky <name>
item delete <name> <id> – vymaže príspevok <id> v nástenke <name>
item update <name> <id> <content> – upraví príspevok <id> v nástenke <name>

5 Informácie o programe

Program sa skladá zo súboru **Makefile**, ktorý slúži na zostavenie programu a nasledovných zdrojových súborov:

- isaclient.c
- isaserver.c
- chelper.c
- chelper.h
- shelper.c
- shelper.h

6 Záver

Projekt mal za cieľ vyskúšať si programovanie sieťovej služby. Bolo potrebné si naštudovať HTTP protokol z RFC dokumentov a následne tieto získané znalosti aplikovať v implementácií samotného programu. Projekt overil nielen komplexne znalosti (analýza RFC dokumentov, práca s BSD schránkami, programovanie v C++, atď.), ale aj programatorské zručnosti - návrh, implementácia, ladenie a testovanie programu. Novozískané vedomosti z tohto projektu sa týkali hlavne programovanie sieťových aplikácií a protokolov (HTTP), na čo slúžia, čo umožňujú a ako fungujú.

Reference

[1] Hyper Text Transfer Protocol
http://www.cs.vsb.cz/grygarek/kotasek/htp02.htm

[2] RFC of Hyper Text Transfer Protocol
https://tools.ietf.org/html/rfc2616

[3] ANSI C like notation
https://wiki.bash-hackers.org
/syntax/quoting?s%5b%5d=ansi&s%5b%5d=sequence#ansi_c_like_strings