HTTP nástěnka

Dokumentace

Filip Jeřábek (xjerab24)

VUT FIT 2019

ISA

1 Obsah

2 Zadání

2.1 Společné zadání pro všechny varianty

Vytvořte komunikující aplikaci podle konkrétní vybrané specifikace pomocí síťové knihovny BSD sockets (pokud není ve variantě zadání uvedeno jinak). Projekt bude vypracován v jazyce C/C++. Pokud individuální zadání nespecifikuje vlastní referenční systém, musí být projekt přeložitelný a spustitelný na serveru merlin.fit.vutbr.cz pod operačním systémem GNU/Linux. Program by však měl být přenositelný. Hodnocení projektů může probíhat na jiném počítači s nainstalovaným OS GNU/Linux, včetně jiných architektur než Intel/AMD, jiných distribucí, jiných verzí knihoven apod. Pokud vyžadujete minimální verzi knihovny (dostupnou na serveru merlin), jasně tuto skutečnost označte v dokumentaci a README.

Vypracovaný projekt uložený v archívu .tar a se jménem xlogin00.tar odevzdejte elektronicky přes IS. Soubor nekomprimujte.

- **Termín odevzdání je 18.11.2019 (hard deadline)**. Odevzdání e-mailem po uplynutí termínu, dodatečné opravy či doplnění kódu není možné.
- Odevzdaný projekt musí obsahovat:
    1. soubor se zdrojovým kódem (dodržujte jména souborů uvedená v konkrétním zadání),
    2. funkční *Makefile* pro překlad zdrojového souboru,
    3. dokumentaci (soubor *manual.pdf*), která bude obsahovat uvedení do problematiky, návrhu aplikace, popis implementace, základní informace o programu, návod na použití. V dokumentaci se očekává následující: titulní strana, obsah, logické strukturování textu, přehled nastudovaných informací z literatury, popis zajímavějších pasáží implementace, použití vytvořených programů a literatura.
    4. soubor *README* obsahující krátký textový popis programu s případnými rozšířeními/omezeními, příklad spuštění a seznam odevzdaných souborů,
    5. další požadované soubory podle konkrétního typu zadání.
- Pokud v projektu nestihnete implementovat všechny požadované vlastnosti, je nutné veškerá omezení jasně uvést v dokumentaci a v souboru README.
- Co není v zadání jednoznačně uvedeno, můžete implementovat podle svého vlastního výběru. Zvolené řešení popište v dokumentaci.
- Při řešení projektu respektujte zvyklosti zavedené v OS unixového typu (jako je například formát textového souboru).
- Vytvořené programy by měly být použitelné a smysluplné, řádně komentované a formátované a členěné do funkcí a modulů. Program by měl obsahovat nápovědu informující uživatele o činnosti programu a jeho parametrech. Případné chyby budou intuitivně popisovány uživateli.
- Aplikace nesmí v žádném případě skončit s chybou SEGMENTATION FAULT ani jiným násilným systémovým ukončením (např. dělení nulou).
- Pokud přejímáte krátké pasáže zdrojových kódů z různých tutoriálů či příkladů z Internetu (ne mezi sebou), tak je nutné vyznačit tyto sekce a jejich autory dle licenčních podmínek, kterými se distribuce daných zdrojových kódů řídí. V případě nedodržení bude na projekt nahlíženo jako na plagiát.
- Konzultace k projektu podává vyučující, který zadání vypsal.

- Před odevzdáním zkontrolujte, zda jste dodrželi všechna jména souborů požadovaná ve společné části zadání i v zadání pro konkrétní projekt. Zkontrolujte, zda je projekt přeložitelný.

Hodnocení projektu:

- **Maximální počet bodů za projekt je 20 bodů.**
  - Maximálně 15 bodů za plně funkční aplikaci.
  - Maximálně 5 bodů za dokumentaci. Dokumentace se hodnotí pouze v případě funkčního kódu. Pokud kód není odevzdán nebo nefunguje podle zadání, dokumentace se nehodnotí.
- Příklad kriterií pro hodnocení projektů:
  - nepřehledný, nekomentovaný zdrojový text: až -7 bodů
  - nefunkční či chybějící Makefile: až -4 body
  - nekvalitní či chybějící dokumentace: až -5 bodů
  - nedodržení formátu vstupu/výstupu či konfigurace: -10 body
  - odevzdaný soubor nelze přeložit, spustit a odzkoušet: 0 bodů
  - odevzdáno po termínu: 0 bodů
  - nedodržení zadání: 0 bodů
  - nefunkční kód: 0 bodů
  - opsáno: 0 bodů (pro všechny, kdo mají stejný kód), návrh na zahájení disciplinárního řízení.

2.2 Zadání pro variantu HTTP nástěnka

Update 11.10.2019: Upresnená verzia IP pre komunikáciu (**IPv4**).
Update 09.10.2019: Upresnený formát názvov nástenieк: **[a-zA-Z0-9]**.
Update 08.10.2019: Opravený príkaz **boards list <name>** na **board list <name>** u klienta.

Vašou úlohou je naimplementovať klient-server aplikáciu nástenka.

Neformálny popis aplikácie

Aplikácia umožnuje klientom spravovať nástenky na serveri pomocou HTTP API. API im umožnuje prezerať, pridávať, upravovať a mazať príspevky na nástenkách ako aj nástenky samotné. Nástenka obsahuje zoradený zoznam textových príspevkov.

Nástenky

Nástenkou sa rozumie usporiadaný zoznam textových (*ASCII*) príspevkov. Každý príspevok ma **id** (*číslované od 1*) 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).
```

Príklad nástenky:

```
[priklad]
1. Jednoriadkovy prispevok.
2. Viacriadkovy prispevok.
Pokracuje na druhom riadku.
3. Dalsi jednoriadkovy prispevok.
```

Po zmazaní príspevku 2. bude nástenka vyzerať následovne:

```
[priklad]
1. Jednoriadkovy prispevok.
2. Dalsi jednoriadkovy prispevok.
```

HTTP API

Aplikácie medzi sebou komunikujú pomocou protokolu HTTP a rozhrania nad ním. Použitá verzia HTTP je **1.1**. Stačí použiť minimálny počet hlavičiek potrebných na správnu komunikáciu, aplikácia by však mala byť schopná *vysporiadať sa aj s neznámymi hlavičkami* (t.j. preskočiť a ignorovať). V prípadoch kedy sa posielajú dáta je použitý **Content-Type:text/plain**. Riadky obsahu sú oddelené len **\n**.

API je definované následovne:

- **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**.

Rozhranie aplikácií

Skompilovaný klient sa bude volať **isaclient**, server bude mať názov **isaserver**.

Oba programy po spustení s parametrom **-h** vypíšu na stdout informácie o spôsobe spustenia.

Server akceptuje jeden povinný parameter, **-p**, ktorý určuje port na ktorom bude server očakávať spojenia. (**./isaserver -p 5777**)

Spôb spustenia klienta vyzerá následovne:

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

kde **<command>** môže byť (za - vždy následuje ekvivalenté API):

- **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>

Klient vypíše hlavičky odpovede na stderr a obsah odpovede na stdout.

Implementácia

Vaše riešenie by malo spĺňať podmienky popísané v spoločnom zadaní. Projekt bude napísaný v jazyku C/C++ a preložiteľný na serveri **merlin**. Použitie neštandardných knižníc (**libcurl** etc.) nie je povolené. Aplikácie budú komunikovať pomocou IPv4.

3 Obecný popis

Aplikace implementuje stranu klienta a serveru pro správu nástěnky pomocí HTTP API. API umožňuje klientovi prohlížet, přidávat, upravovat a mazat jak příspěvky na nástěnkách, tak i nástěnky samotné. Nástěnka obsahuje seřazený seznam textových příspěvků.

3.1 Úvod

Pro tvorbu aplikace byl zvolen programovací jazyk `C++`. Kód je tvořen v IDE CLion od společnosti JetBrains na operačním systému `Ubuntu 18.04`. Ke správě verzí byl zvolen verzovací systém git.

3.2 Klient

Klient dle zadaných argumentů odesílá požadavky pro správu nástěnky na server. Jednotlivé způsoby volaní jsou uvedeny v kapitole TODO. Klientská aplikace má návratovou hodnotu **0**, pokud se daný požadavek na serveru povedl a hodnotu **-1**, pokud server vrátil chybovou návratovou hodnotu.

3.3 Server

Server čeká na navázání spojení a příjem požadavku od klienta. Tento požadavek si poté zpracuje a odešle na něj příslušnou odpověď. Argumenty akceptující server jsou popsány v kapitole TODO. Implementace

3.4 Zpracování argumentů a volání programu

3.4.1 Klient

Program nejprve pomocí funkce `checkHelp()` zkontroluje, zda byl zadán přepínač *–h* a případně vypíše nápovědu a ukončí se.

Pro zpracování argumentů je použita doporučená rodina `getopt()`, konkrétně `getopt_long_only()`. Funkce pro načtení argumentů je pojmenována `loadArgs()`.

Ta nejprve zavolá funkci `checkArgCount()`. Tato funkce kontroluje, zda je zadán počet argumentů v rozsahu min max.

Poté načte do struktury `myArgs` argumenty s přepínači a ve struktuře zanechá informace o tom, které přepínače byly načteny. Dále probíhá validace zadání povinných argumentů.

Dále je zavolána funkce `loadCommand()` která pomocí `regexů` v argumentech vyhledává jeden z předem očekávaných příkazů pro správu http nástěnky. Pokud nalezne shodu, pak do struktury `myArgs` vloží typ a hodnoty příkazu, jako je například „`board add jméno`“.

3.4.1.1 Přijímané argumenty

Způsob volání klienta:

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

kde

- <host> je adresa hosta, na kterou se bude požadavek odesílat
- <port> je port, na kterém bude probíhat komunikace.
- <command> je:
  - `boards` - Vráti seznam dostupných nástěnek, jedna na řádek.
  - `board add <name>` - Vytvoří novou prázdnou nástěnku s názvem name.
  - `board delete <name>` - Smaže nástěnku name a veškerý její obsah.
  - `board list <name>` - Zobrazí obsah nástěnky name.
  - `item add <name> <content>` - Vloží nový příspěvek do nástěnky name. Příspěvek je vložený na konec seznamu.
  - `item delete <name> <id>` - Změní obsah příspěvku číslo id v nástěnce name.
  - `item update <name> <id> <content>` - Smaže příspevek číslo id z nástěnky name.

kde:

- <name> je řetězec znaků a-z A-Z a 0-9
- <id> je číselná hodnota v rozsahu 0-LONG_MAX (`2147483647` ($2^{31}-1$)).
- Příklad volání pak bude:

```
./isaclient -H localhost -p 8080 boards
```

3.4.2 Server

Program nejprve pomocí funkce `checkHelp()` zkontroluje, zda byl zadán přepínač *–h* a případně vypíše nápovědu a ukončí se.

Poté pomocí funkce `checkArgCount()` ověří počet zadaných argumentů. V tomto případě 2. Přepínač –p a jeho hodnotu.

Jelikož program neočekává žádné další argumenty, kontroluje, zda byl spuštěn s výhradně jen přepínačem –p a jeho hodnotou. Tato hodnota udává číslo portu, na kterém bude aplikace naslouchat. Pokud je zadán neplatný řetězec (abc), nebo port mimo rozsah (max `2147483647` ($2^{31}-1$)), pak program vrátí odpověd `400 Bad Request`.

3.4.2.1 Přijímané argumenty

Server přijímá pouze přepínač –h, nebo přepínač –p a k němu jeho hodnotu. Ostatní argumenty povedou k chybě.

Příklad volání pak bude:

```
./isaserver -p 8080
```

3.5 Komunikace serveru a klienta

Server a klient spolu komunikují přes API za pomocí jednoduchých http requestů pomocí síťové knihovny BSD sockets. Velikost bufferu byla nastavena na 2048, což pro naši aplikaci představuje optimální velikosti. Pro příspěvky zadávané jako argument je dostatečně velký a naopak ne zbytečně velký.

3.5.1 Request klienta

Request se skládá ve funkci `createRequest()`, kde se pomocí příkazu switch vybere příslušná funkce pro naplnění requestu. Funkce nese podobný název, jako požadovaný příkaz (cmdBoards, cmdBoardAdd..).

Request se skládá pouze z nezbytných hlaviček:

- metoda, příkaz API, verze HTTP 1.1
- host
- content-type
- content-length

Takto vytvořený request se poté přidá do hlavní funkce main a odešle se serveru.

Překlad argumentů programu na API vypadá následovně:

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>`

3.5.2 Response serveru

Server zpracuje požadavek klienta a podobně jako klient pomocí funkce `getCommand()` a `createResponse()` zpracuje požadavek, provede příslušné operace nad nástěnkami, vytvoří odpověď a odešle ho.

3.5.3 Zpracování response klientem

Klient pomocí `regexů` rozparsuje odpověď a zkontroluje, zda server nevrátil http odpověď s návratovým kódem 4xx. Poté dle zadání vypíše na stderr a stdout.

3.6 Výstup klienta

Klient vypisuje na `stderr` hlavičku odpovědi a na `stdout` případný kontext – názvy nástěnek a příspěvky nástěnky.

Klient vrací návratovou hodnotu 0 v případě, že operace, která mu byla zadána, proběhla na serveru úspěšně.

Pokud nastala chyba, jak na straně klienta, tak na straně serveru (server vrátil odpověď 4xx), pak vrací hodnotu -1.

4 Testování

Program jsem testoval jak na OS linux na svém počítači, tak i na serveru Merlin. Pro minimalizaci chyb při sestavování a zpracovávání odpovědi a požadavku proběhlo i testování stylem můj server / klient mých kolegů Daniela Konečného a Tomáše Ryšavého, kteří měli stejně zadání a stejně tak i kombinaci můj klient / server kolegů.

5 Užitečné odkazy

http request

[1] https://www.tutorialspoint.com/http/http_requests.htm

http response

[2] https://www.tutorialspoint.com/http/http_responses.htm

regex pro zpracovávání požadavku a odpovědi

[3] http://www.cplusplus.com/reference/regex/smatch/

BSD sockets

[4] http://wiki.treck.com/Introduction_to_BSD_Sockets

[5] https://www.earchiv.cz/a93/a315c110.php3

[6] https://www.keil.com/pack/doc/mw/Network/html/group__net_b_s_d___func.html

6 Závěr

Projekt byl přesně a srozumitelně napsaný. Nebyl nijak složitý, ale přesto velice dobře splnil svůj účel – osahat si http komunikaci. Veškeré podmínky pro server i klienta v zadání jsou implementovány a funkční.