FAKULTA INFORMAČNÝCH TECHNOLÓGIÍ
VYSOKÉ UČENÍ TECHNICKÉ BRNO

Projektová dokumentácia

Sieťové aplikácie a správa sietí

Programovanie sieťovej služby
Varianta: http nástenka

Barbora Nemčeková, xnemce06
11.11.2019

Obsah

Zadanie projektu 2
Dôležité pojmy 3
BSD schránky 3
Komunikácia klient-server 3
Vytvorenie spojenia 3
Komunikácia 3
Uzatvorenie spojenia 3
HTTP protokol 4
Požiadavka 4
Odpoveď 4
Spoločné znaky 4
Popis implementácie 5
Klient 5
Server 5
Špecifikácie 5
Obmedzenia 6
Použitie programu 7
Popis aplikácie 7
Spustenie programu 7
Literatúra 8

Zadanie projektu

Úlohou bolo vytvoriť komunikujúcu aplikáciu pomocou sieťovej knižnice BSD sockets, implementovanú v jazyku C/C++.
Aplikácia umožňuje klientom spravovať nástenky a serveri pomocou HTTP API. API umožňuje klientom vytvárať nástenky, pridávať, mazať a upravovať príspevky. Taktiež je možné vymazať aj samotnú nástenku.

Dôležité pojmy

V tejto časti dokumentu je popísaný základný princíp BSD schránok, komunikácie typu klient-server a http protokolu, ktorý je potrebné poznať pre návrh a implementáciu aplikácie http nástenka.

BSD schránky

Schránky sú nezbytným prvkom pre komunikáciu po sieti, vytvárajú koncový komunikačný bod. Táto abstraktná dátová štruktúra obsahuje všetky potrebné informácie pre sieťovú komunikáciu. Pre vytvorenie a inicializáciu schránky sa používa funkcie socket(), ktorá ako parametre berie špecifikáciu rodiny protokolov, typ schránky a typ protokolu. Pre vytvorenie komunikácie sa musia obidve schránky prepojiť a preto každá musí poznať nielen svoju IP adresu a číslo portu ale aj IP adresu a číslo portu schránky s ktorou sa chce spojiť.

Komunikácia klient-server

Klient aj server si vytvoria svoje schánky pomocou vyššie spomenutej funkcie socket(). Server taktiež použije funkciu bind() pre prepojenie schránky s konkrétnym portom a funkciu listen(), po ktorej server čaká na spojenie.

Komunikácia sa dá rozdeliť na 3 časti:

1. Vytvorenie spojenia
2. Komunikácia
3. Uzatvorenie spojenia

Vytvorenie spojenia

Spojenie zahajuje klient pomocou funkcie connect() a server ho akceptuje funkciou accept(). Menším problémom pri spojení tohto typu je, že funkcia connect() je blokujúca, čo môže spomaliť beh celej aplikácie v prípade chyby. Po vytvorení spojenie môžeme prejsť ku samotnej komunikácií.

Komunikácia

Pre prenos dám môžeme použiť až dva páry funkcií. Prvým sú funkcie read() a write(). Druhým párom sú funkcie recv() a send(), ktoré obsahujú pokročilé nastavenia spojenia. Veľmi dôležitou súčasťou prijímania a odosielania dát je veľkosť bufferu. Pri nedostatočnej veľkosti sa môže stať, že nebudeme schopný prijať celú správu správne a tak prichádzame o dáta. Preto je potrebné kontrolovať návratovú hodnotu týchto funkcií.

Uzatvorenie spojenia

Správne ukončenie spojenia je rovnako dôležité ako jeho zahájenie. Pri nekorektnom uzavretí zostávajú v systéme alokované prostriedky pre komunikáciu až pokým nevyprší časový limit čakanie na odpoveď od druhej strany. Tento limit sa nazýva maximum segment size a závisí na konkrétnej implementácií. Na ukončenie spojenia existujú 2 funkcie: close() a shutdown(). Close() uzatvára obe strany pne duplexného spojenia, čo môže trvať dlhšiu dobu, keďže na ceste môžu byť ešte dáta. V prípade, že počet referencií na schránku je väčší ako jedna, spojenie sa úplne neuzavrie ale len sa zníži počet referencií. Druhou možnosťou je použitie funkcie shutdown(), ktorá dovoľuje uzavrieť len jednu stranu komunikácie a to bez ohľadu na počet referencií na schránku.

HTTP protokol

Http protokol je protokol na aplikačnej vrstve používaný najmä v komunikácií klient-server pre službu WWW. Používa sa ako generický protokol pre komunikáciu medzi používateľským agentom, proxy bránami a inými internetovými systémami.

Tento protokol definuje požiadavky a odpovede medzi klientmi a servermi.

Požiadavka

Požiadavka klienta pozostáva z metódy, URI a verzií protokolu. Ďalej nasleduje MIME správa, ktorá obsahuje hlavičky a prípadné telo správy. Verzia protokolu udáva formát správy a nasledovnej komunikácie. URI je v http protokole považovaný za formátovaný string, ktorý definuje zdroj, či už podľa mena, cesty alebo inej charakteristiky. Metóda označuje, čo sa bude vykonávať na zdroji. Existujú nasledovné metódy:

- GET
- HEAD
- POST
- PUT
- DELETE
- TRACE
- CONNECT.

Hlavička požiadavku umožňuje odoslať upresňujúce informácie ako napríklad akceptovaný charset, encoding, jazyk, podmienené akcie a podobne.

Odpoveď

Odpoveď serveru sa skladá zo Statusu na prvom riadku, ktorý obsahuje verziu protokolu, číselný status kód a jeho slovnú špecifikáciu. Status kódy sa delia do skupín, podľa prvej číslice tohto trojčíslia:

- 1xx – informačné
- 2xx – úspech
- 3xx – presmerovanie
- 4xx – Error na strane klienta
- 5xx – Error na strane serveru.

Status kódy sú rozšíriteľné. Aplikácia nemusí poznať všetky kódy, iba ich skupiny, čiže v prípade, že v odpovedi dostane kód, ktorý nepozná, napríklad 432, aplikácia musí vyhodnotiť, o ktorú skupinu ide. V tomto prípade aplikácia vie, že niečo nebolo správe v žiadosti. V hlavičke odpovedi je možné, rovnako ako v hlavičke požiadavky, spresniť správu.

Spoločné znaky

Hlavičky správy obsahuje aj žiadosť klienta aj odpoveď servera. Tieto hlavičky nesú metainformáciu o tele správy, metóde alebo status kóde. Niektoré hlavičky sú povinné v rôznych prípadoch, napríklad pri konkrétnej metóde alebo status kóde.

Po požiadavke/statuse ale aj po každej hlavičke nasleduje znak CRLF (nový riadok). V prípade, že správa nesie aj telo, toto telo je od hlavičiek oddelené jedným prázdnym riadkom. Prípady, kedy je telo správy povolené sú definované v jednotlivých požiadavkách a odpovediach.

Popis implementácie

Základom tejto aplikácie je klient a server, ktorý navzájom spolu komunikujú. Ich komunikačné rozhranie je implementované pomocou BSD schránok. Pri vytvorení schránky je špecifikovaná rodina protokolov na AF_INET, keďže táto aplikácie používa iba IPv4 adresy a typ schránky na SOCK_STREAM, nakoľko http komunikácia využíva spojenie TCP.

Klient

Aplikácia klienta pracuje jednoducho. Najprv sa rozanalyzujú argumenty, z ktorých sa vyčíta adresa serveru, taktiež jeho port a akcia, ktorú s užívateľ praje vykonať. Aplikácia vytvorí schránku a pripojí sa na server. Zostaví správu podľa želania užívateľa a odošle ju. Správa obsahuje minimálny počet hlavičiek. Klient čaká na odpoveď od serveru. Po prijatí vypíše na štandardný výstup obsah správy a na štandardný errorový výstup vypíše hlavičky, vrátane riadku so statusom.

V prípade, že akcia, ktorú si užívateľ želal vykonať dopadla úspešne, návratový kód aplikácie je 0. V opačnom prípade sa vráti -1.

Server

Server čaká v nekonečnej smyčke na klienta. Potom ako sa klient pripojí, prijme sa správa. Aplikácia komunikuje pomocou HTTP protokolu, z čoho vyplýva, že hlavička a telo správy sú oddelené dvoma znakmi nového riadku, čo sa v implementácií využíva na oddelenie týchto dvoch informácií. Z hlavičky sa oddelí prvý riadok, ktorý nesie informáciu o požiadavke klienta, a ďalej sa rozanalyzuje, aby boli zavolané potrebné funkcie. Potom čo sú všetky potrebné funkcie vykonané, zostaví sa odpoveď tak, aby zodpovedala formátu HTTP a odošle sa.

Správa obsahuje minimálny počet hlavičiek a to konkrétne odpoveď serveru o úspešnosti akcie (Response), typ obsahu(Content-type), ktorý je v tomto projekte vždy text/plain a veľkosť obsahu správy(Content-Lenght).

Po odoslaní správy server zatvára socket a čaká na ďalšieho klienta.

Nová nástenka sa vytvára ako inštancia objektu Board. Tento objekt obsahuje všetky potrebné funkcie na manipuláciu s obsahom nástenky. Zároveň, keď sa nástenka vytvorí, pridá sa aj do objektu List, kde sú uložené názvy všetkých násteniek a odkazy na ne. Príspevky na nástenke sa mapujú na premennú typu std::map, kde kľúčom je automaticky pridávané id tak, aby príspevky išli vzostupne za sebou a aby užívateľ mohol pomocou tohto id pristupovať ku príspevkom.

Špecifikácie

V prípade, že klient si žiada vykonať akciu, ktorá mu nič nevracia (pridanie/zmazanie príspevku/nástenky, modifikácia príspevku), sa počíta s tým, že klient potrebuje byť informovaný o úspešnosti svojej žiadosti. Z tohto dôvodu sa vždy odosiela telo obsahujúce túto informáciu.

Veľkosť bufferu je v implementácií implicitne daná a to na hodnotu 4096, ktorá by mala pokryť aj hraničný prípad. Tým sa očakáva obsah správy veľkosti jednej A4, čo by malo

obsahovať približne 3000 znakov a taktiež sa predpokladá, že by boli použité všetky štandardné hlavičky, ktoré by niesli každá informáciu o 20 znakoch.

Ak klienta žiada zobrazenie násteniek, napriek tomu, že žiadnu ešte nevytvoril, server vráti odpoveď 404 Not found. Poslať prázdnu správu s dĺžkou 0 by mohlo byť pre užívateľa mätúce. Táto situácia nebola v zadaní špecifikovaná, preto som sa rozhodla pre takéto riešenie.

Obmedzenia

V implementácií sú použité regulárne výrazy, čo spôsobuje dlhší preklad programu. Môže trvať niekoľko sekúnd. Pri testovaní preklad trval zvyčajne cca 3 sekundy.

Použitie programu

Popis aplikácie

Klient môže vytvárať na serveri nástenky. Názov nástenky môže obsahovať len alfanumerické znaky. Po tom, ako sa nástenka vytvorí, je prázdna. Následne môže klient pridávať do nástenky príspevky, ktoré sa radia vzostupne za sebou a každému príspevku je priradené id podľa poradia. Nový príspevok sa pridá vždy na koniec nástenky. Príspevok môže obsahovať od jedného znaku, až po niekoľko riadkov. Užívateľ môže nástenku s jej obsahom zobraziť. Príspevky na nástenke je možné modifikovať alebo vymazať. Ku konkrétnym príspevkom sa pristupuje pomocou jeho id.

Nástenky na serveri ostávajú po celú dobu behu programu. V prípade, že užívateľ server ukončí, prichádza o všetky nástenky a ich obsah.

Spustenie programu

Pre využívanie aplikácie je potrebné spustiť ako prvé server. Server sa spúšťa jednoducho, a to formou:

```
./isaserver -p <port>
```

Kde <port> je číslo portu, na ktorom bude server očakávať komunikáciu.

V prípade, že si užívateľ praje ukončenie serveru, musí ho vykonať manuálne.

Klient sa spúšťa formou:

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

<host> môže byť ip adresa alebo názov serveru

<port> je číslo portu servera, na ktorom sa klient pripája na server

<command> môže byť:

- **boards** – zobrazí všetky názvy násteniek
- **board add <name>** - pridá nástenku s názvom <name>
- **board delete <name>** - zmaže nástenku s názvom <name>
- **board list <name> -** zobrazí obsah nástenky <name>
- **item add <name> <content>** - pridá na koniec nástenky s názvom <name> príspevok s obsahom <content>
- **item delete <name> <id>** - zmaže z nástenky s názvom <name> príspevok číslo <id>
- **item update <name> <id> <content>** - zmení na nástenke s názvom <name> príspevok číslo <id> na obsah <content>

Literatúra

IETF Documents. (dátum neznámy). Cit. 10. November 2019. Dostupné na Internete: tools.ietf.org: https://tools.ietf.org/html/rfc2616

J. F. Kurose, K.W.Ross. (2003). *Computer Networking: A Top-Down Approach Featuring the Internet.* Adison-Wesley.

Matoušek, P. (2014). *Síťové aplikace a jejich architektura.* VUTIUM.