VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Dokumentace k projektu do predmetu IPK

Klient-server pro jednoduchý přenos souborů

12. března 2018

Stoika Anastasiia xstoik00@stud.fit.vutbr.cz

Obsah

1 Úvod

Projekt se skladá ze dvou částí. Prvním úkolem je navrhnout vlastní aplikační protokol realizující přenos souboru. Druhým úkolem projektu je naprogramovat klientskou a serverovou aplikaci v C/C++ realizující čtení/zápis souborů z/na server. Pedpokládá se, že server je konkuretní, tzv. může obsluhovat více klientů zároven.

2 Teorie

2.1 Komunikace klient - server

Klient i server jsou aplikační procesy, které komunikují přes síťové rozhraní. Může jít i o procesy běžící na stejném počítači.

Základní vlastnosti komunikace:

- Popsána protokolem.
- Klient poskytuje rozhraní pro uživatele.
- Klient posílá požadavky na zpracování.
- Server čeká na požadavky, zpracuje je a pošle odpověd'.
- Klient je iniciatorem komunikace.

2.2 Protokol

Protokol je soubor syntaktických a sémantických pravidel určujících výměnu dat. Popisuje vytvoření spojení, adresování, přenos dat, řízení toku, zabezpečení. V našem případě protokol bude popisovat komunikace mezi klientem a serverem, výměnu zpráv a dat mezi aplikačními procesy.

Protokol lze popsat neformálně (například slovně, viz standardy RFC) nebo formálně. Mezi nejčastěji používané formální popisy protokolů patří: stavové automaty (konečné automaty), gramatiky, grafové modely, algebraické prostředky.

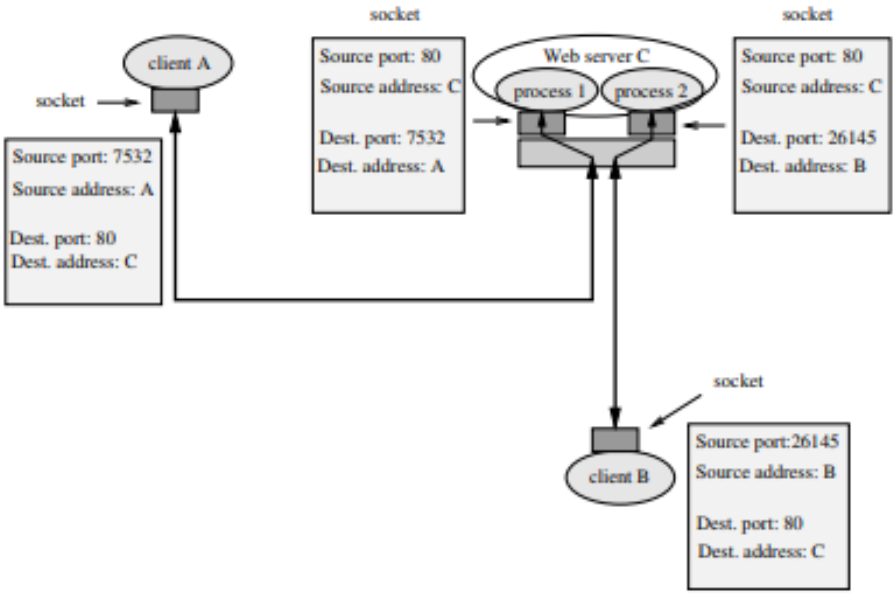
2.3 Schránky (sockets)

Základní vlastnosti:

- Vytvářejí aplikační programové rozhraní na transportní vrstvě pro přenos aplikačních dat. API pro komunikující procesy.
- Koncový komunikační bod.
- Schránka je identifikována typem, IP adresou a portem.
- Abstraktní datová struktura obsahující údaje pro komunikaci (lokální IP adresa, lokální port, vzdálená IP adresa, vzdálený port).
- Základní typy schánek: SOCK_STREAM (TCP), SOCK_DGRAM (UDP), SOCK_RAW.

Pouzití schránek pro komunikaci je dobre znazorněno na obrázku 1.

Obrázek 1: Komunikace pomocí schránek.



3 Implementace

V dané sekci budou ukázany vlastní návrh aplikačního protokolu a implementace klientské a serverové aplikace. Jako typ spojení bylo zvoleno komunikace nad TCP, pro spolehlivý přenos dat.

3.1 Zpracování vstupních parametrů

Na stráně klienta pro parsování vstupních parametrů byla použitá funkce getopt(). Na stráně serveru jsou parametry parcovány jenom pomocí vlastní funkce getParams().

3.2 Aplikační protokol

Pro účely projektu byl navržen vlastní aplikační protokol, určující vyměnu dat mezi klientem a serverem. Protokol je popsán formou konečného automátu na obrázku č.2.

3.3 Programování komunikace nad TCP

Činnost komunikace TCP lze rozdělit na tři základní části:

1. Vytvoření spojení TCP.
2. Komunikace, výměna dat.
3. Uzavření spojení.

3.3.1 Vytvoření spojení TCP

TCP vytváří spojení pomocí mechanismu třífázové synchronizace ”three-way handshake”, který zahrnuje výměnu paketů SYN, SYN+ACK,ACK. Spojení vyvolává blokující funkce connect() na straně klienta TCP. Tato funkce způsobí vyslání paketu TCP s příznakem SYN. Na straně serveru čeká proces na přijetí spojení funkcí accept(). Po výměně synchronizačních paketů je ustaveno spojení.

Pro vytvoření spojení byli použity funkce:

- socket() - slouží k vytvoření nového socketu požadovaného typu.

  int socket(int family, int type, int protocol)

  První argument, family, rozhoduje o výběru způsobu komunikace. Nastaveno na AF_INET (komunikace přes internet pomocí IPv4). Druhý argument určuje typ spojení, v našem případě nastaveno na SOCK_STREAM. Třetí argument je nastaven na 0, což znamená defaultní protokol.

- bind() - svázání schránky na straně serveru s konkrétním portem.

  Na stráně klienta není potřeba používat bind(), při žádosti o komunikaci volný port klientovi přidělí síťová knihovna. Na straně serveru port zadá uživatel spouštějící server, přičemž není možné použit čísla rezervovaných portů (do 1024).

- listen() - pasivní otevření na straně serveru, server čeká na spojení.

- connect() - aktivní otevření na straně klienta.

- accept() – přijetí spojení na stráně serveru, ustavení komunikace.

Informace o aktuální schránce (například číslo lokálně přiděleného portu) je získána pomoci funkce getsockname().

Také byla použita funkce gethostbyname() pro DNS rezoluci.

3.3.2 Výměna dat

Pro výměnu dat klient a server používají funkce recv() a send().

3.3.3 Ukončení spojení

Server běží v nekonečné smyčce. Ukončení spojení TCP iniciuje klient. Pro uzavření schránky se používá funkce close(). Činnost serveru ukončí uživatel.

3.4 Implementace přenosu souborů

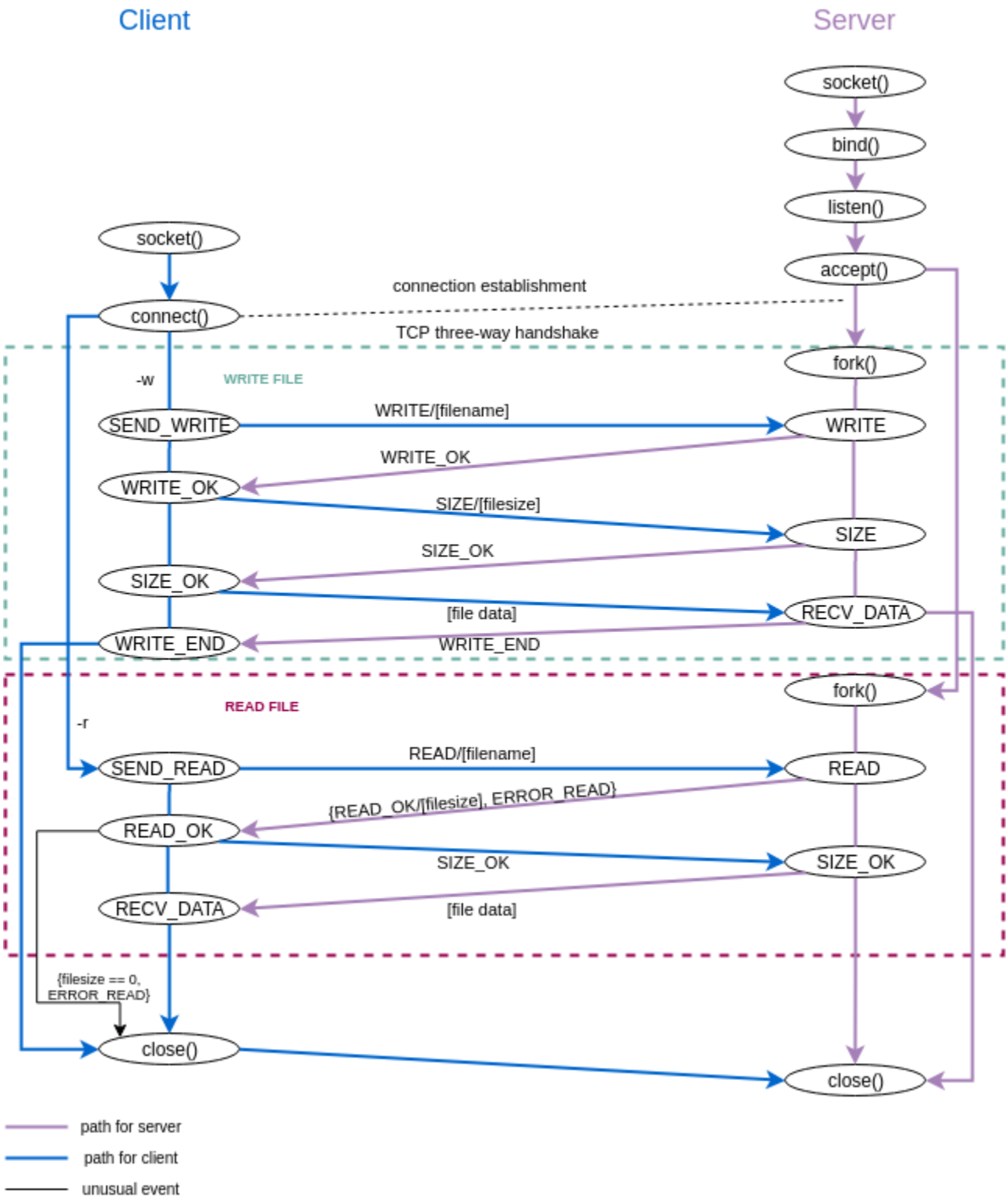
Pro otevrení souboru byla použita funkce fopen s paramentry otevření pro čtení/zápis binárně. Funkce fseek() a ftell() pro zjištění velikosti přenášeného souboru bud' ze strány klienta, nebo servera.

Přenos dat se provádí po blocích o velikosti 4096 B. Aplikace, která provádí přenos souboru, otevírá soubor pro čtení, v cyklu načítává data pomocí funkce fread(), která umožnuje hned načíst velký blok dat, až fread() nevrátí 0, a posílá data pomocí send() se zpožděním 5000 mikrosekund. Pro zpoždění se používá funkce usleep().

Aplikace která přijímá data. Pokud velikost souboru je méně, než jeden blok o velikosti 4096, okamžitě zapíše data pomocí fwrite() a klient ukončí činnost. V případě když soubor je větší než buffer, zjistí se do kolika bloků bude rozdělen soubor a velikost posledního bloku. Zápis probíhá ve smyčce pomocí fwrite() pokud nejsou načtěny všechny bloky. Po tom se zapíše posledný block dat.

Uzavření souboru pomocí fclose().

Obrázek 2: Aplikační protokol realizující přenos souborů.

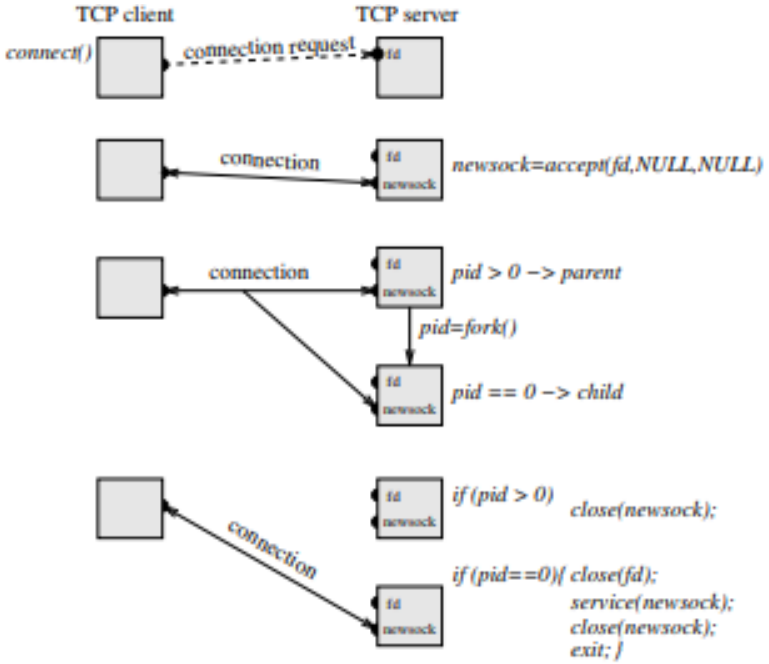


3.5 Vytvoření konkurentního serveru TCP

1. socket() - vytvořit schránku
2. bind() - připojení schránky na port
3. listen() - čekání na spojení
4. accept() - přijetí spojení, vytvoření nové schránky
5. fork() - vytvoření synovského procesu
6. recv(), send() - výměna dat v rámci synovského procesu
7. close() - uzavření nově vytvořené schránky
8. exit() - ukončení synovského procesu
9. close() - rodičovský proces zavře schránku.

Na obrázku 3 je znazorněno použití schránek u konkurentního serveru TCP.

Obrázek 3: Použití schránek u konkurentního serveru TCP.



4 Testování

V dané sekci budou ukázany výsledky testování implementovaných aplikací demonstrující jejich činnost.

1. **Zápis souboru na server**

   Zápis souboru na server byl úspěšně otestován na souborech různých velikostí a typů. Na server je možné uspěšně přenest i soubory s velkými velikosti, napr. bylo vyskoušeno odeslat přednášku o velikosti 677,7 MB, nebo velké textové soubory. Zápis takového souboru na server trvá cca 10 minut.

2. **Čtení souboru ze serveru**

   Čtení souboru také bylo úspěšně otestováno na souborech různých velikostí a typů.

3. **Čtení prázdného souboru**

   Při pokusu načíst prázdný soubor ze serveru, klient vytvoří soubor s takovým jmeném a ukončí svoje činnost.

4. **Čtení neexistujícího souboru**

   Při pokusu načíst soubor, ktery na serveru neexistuje, server pošle klientoví zprávu o chybě (jak je vidět z aplikačního protokolu), a klient ukončí svoje činnost s chybou.

5. **Současné čtení více klienty**

   Současné čtení stejného/různých souborů více klienty proběhlo úspěšně.

6. **Současný zápis na server více klienty**

   Současný zápis souborů na server více klienty současně proběhlo úspěšně.

7. **Současný zápis stejně pojmenovaného souboru více klienty**

   Při současném zápisu na server souboru se stejným jmenem více klienty bude docházet k přepisu existujícího souboru.

8. **Současné čtení souboru jedným klientem při jeho zápisu jiným klientem**

   Aplikace neošetřuje situaci když jeden klient bude posílat na server soubor, který zároven chce načíst z serveru druhý klient. V takovém případě, klient který chce číst soubor načte jenom tu jeho část, která stíhla se nahrat na server.

Porovnání souborů probíhalo pomocí příkazu: diff -q server/[filename] client/[filename]

4.1 Problémy během vývoje

Při okamžitém odesílání velkých dat (soubory velikostí nad cca 10 MB) předpokládaně docházelo k přeplnění buffru a zahazování paketů. Přenos souboru o velikosti 677,7 MB trval cca 5 sekund, zřejmě nebyl přenesen korektně. Proto byla potřeba zavest odesílání dat se zpožděním. Zvolené zpoždění 5000 mikrosekund.

5 Literatura

- O. Ryšavý, J. Ráb, IPK - BSD schránky - 3. přednáška
- Ing. Petr Matoušek, Ph.D., M.A., ISA - přednáška Pokročilé programování sítí TCP/IP
- P. Grygárek, Softwarová rozhraní systémů UNIX pro přístup k síťovým službám,
  `http://www.cs.vsb.cz/grygarek/LAN/sockets.html`