

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií



POČÍTAČOVÉ KOMUNIKACE A SÍTĚ
2017/2018

Projekt č. 1

**Varianta 2: Klient-server pro jednoduchý přenos
souborů**

Obsah

Obsah	2
Zadání	3
Implementace	4
Čtení a odeslání souboru	4
Příjem a zápis souboru	5
Ošetření chyb	5
Funkce protokolu	6
Screenshot z testování	7
Zdroje	8
Odkazy	8

Zadání

Vaším úkolem je:

1)

Seznamte se s kostrami kódů pro programování klientských a serverových síťových aplikací (přednáška třetí) za použití BSD socketů. Navrhněte vlastní aplikační protokol realizující přenos souboru a relevantní informace k projektu uveďte v dokumentaci.

Navrhněte vlastní aplikační protokol, kterými poté spolu budou komunikovat klient a server z bodu [2] tohoto zadání. Inspirovat se můžete například výměnou zpráv protokolů TFTP či FTP.

V dobré dokumentaci se očekává následující: titulní strana, obsah, logické strukturování textu, výcuc relevantních informací z nastudované literatury, popis aplikačního protokolu formou konečného automatu, popis zajímavějších pasáží implementace, demonstrace činnosti implementovaných aplikací, normovaná bibliografie.

2)

Naprogramujte jak klientskou tak serverovou aplikaci v C/C++ realizující čtení/zápis souborů z/na server.

Umístění souborů na straně serveru je v aktuálním adresáři, ve kterém se nachází server. Spuštění klienta předpokládá provedení pouze jedné operace a to uložení nebo přečtení souboru se zadaným jménem. Komunikace mezi serverem a klientem bude schopna se vypořádat se ztrátami paketů (a to buď na aplikační, nebo transportní vrstvě).

Implementace

Při vytváření aplikačního protokolu byl důležitý výběr vhodného transportního protokolu. Jelikož se jedná o protokol který má za úkol přenos souborů, je pro nás důležité, aby všechna data byla spolehlivě doručena. Proto jsem pro tento projekt vybral protokol TCP.

První krok tedy byl navázání spojení mezi klientem a serverem. Klient i server si vytvoří socket a přidělí jim argumenty pomocí funkce *bind()*. Server následně čeká na příchozího klienta, aby s ním mohl vytvořit spojení.

Jakmile je spojení navázáno, klient posílá přes socket informace o plánované akci, čili čtení nebo zápis na server a název daného souboru. Podle přijatých informací se následně provádí náležité akce.

Pokud nedojde k chybě, například chyba při zápisu do souboru, a všechna data jsou úspěšně odeslána/přijata, spojení s klientem je ukončeno a server následně čeká na nového klienta.

Čtení a odeslání souboru

```
while (send_total < file_size)
{
    long send_now = 0;
    long char_read = 0;
    char_read = fread(buff, sizeof(char), BUF_SIZE, fr);

    if ((send_now = send(sock_cl, buff, char_read, 0)) < 0)
    {
        fprintf(stderr, "Chyba pri odesilani souboru\n");
        return -1;
    }
    send_total += send_now;
    bzero(buff, BUF_SIZE);
}
```

Při čtení souboru a jeho následném odeslání je důležité, aby bylo odesláno přesně tolik dat, kolik soubor zabírá. Kvůli tomu je třeba kontrolovat počet již přečtených a následně odeslaných dat s celkovou velikostí souboru. Pokud je velikost přečtených dat menší, než je velikost bufferu, který posíláme (typicky při plnění posledního bufferu), je nutné, abychom

poslali pouze tolik dat, kolik jsme přečetli, nikoliv velikost celého bufferu, proto je potřeba do argumentu funkce *send()* velikost přečtených dat navrácených funkcí *fread()*.

Příjem a zápis souboru

```
while ((recived=recv(sock_cl, buff, BUF_SIZE,0)) > 0)
{
    written = fwrite(buff, sizeof(char), recived, fr);

    if(written != recived)
    {
        fprintf(stderr,"Chyba pri zapisu do souboru\n");
        return -1;
    }

    bzero(buff,BUF_SIZE);
}
```

Při zápisu souboru je nutné kontrolovat počet přijatých dat a porovnat je s daty zapsanými, abychom mohli indikovat případnou chybu při zápisu do souboru. Data čteme dokud přichází, poté cyklus ukončíme.

Ošetření chyb

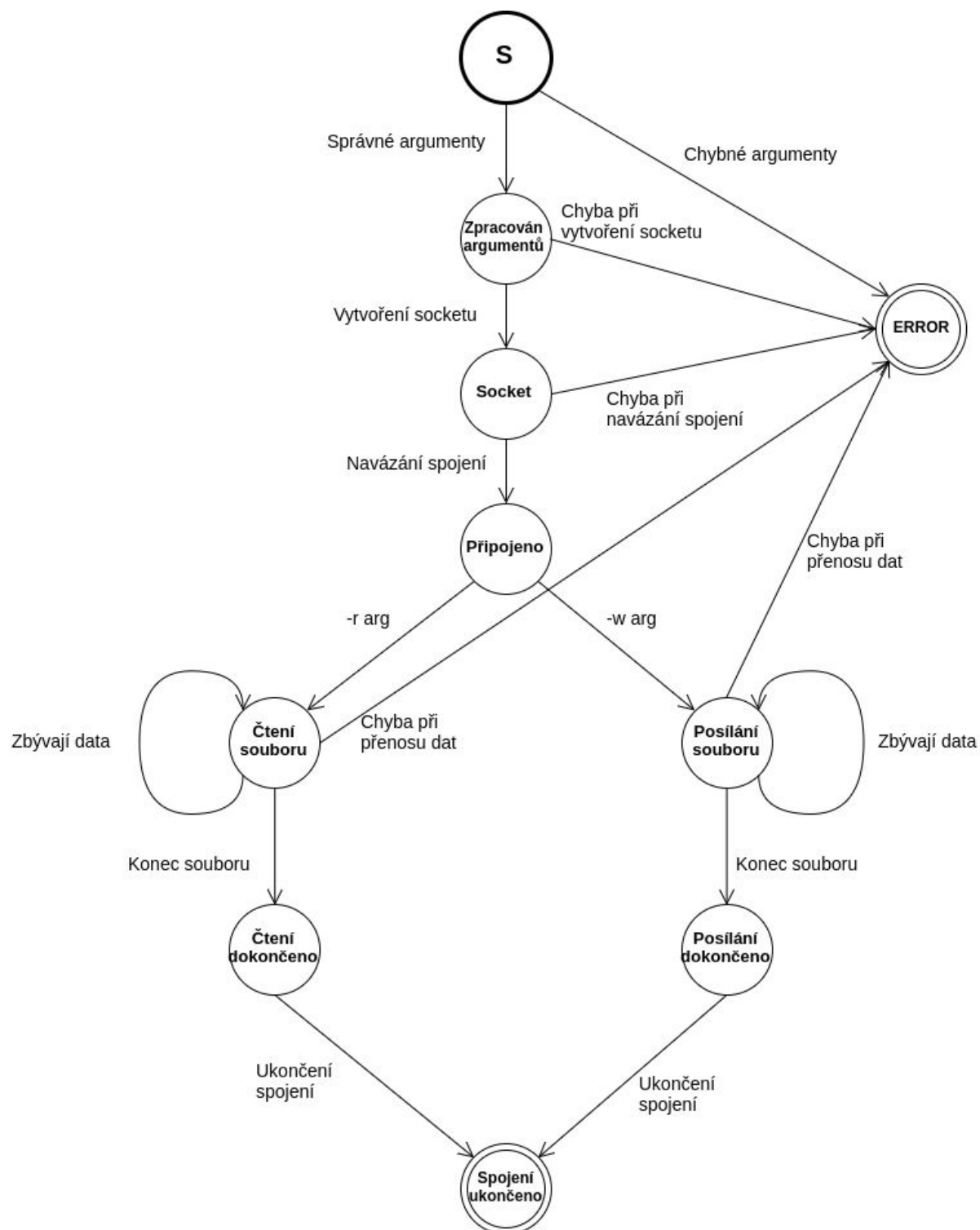
Program je ošetřen proti několika různým chybám, v případě jejich zachycení vypíše zprávu na standardní chybový výstup a navrací hodnotu -1.

Při špatném zadání parametrů tj. chybný počet nebo špatná struktura program ukončí svoji aktivitu a vypíše chybovou hlášku. Jako chybu mimo jiné bere i špatně zadané číslo portu(přítomnost písmen nebo špatný rozsah čísla).

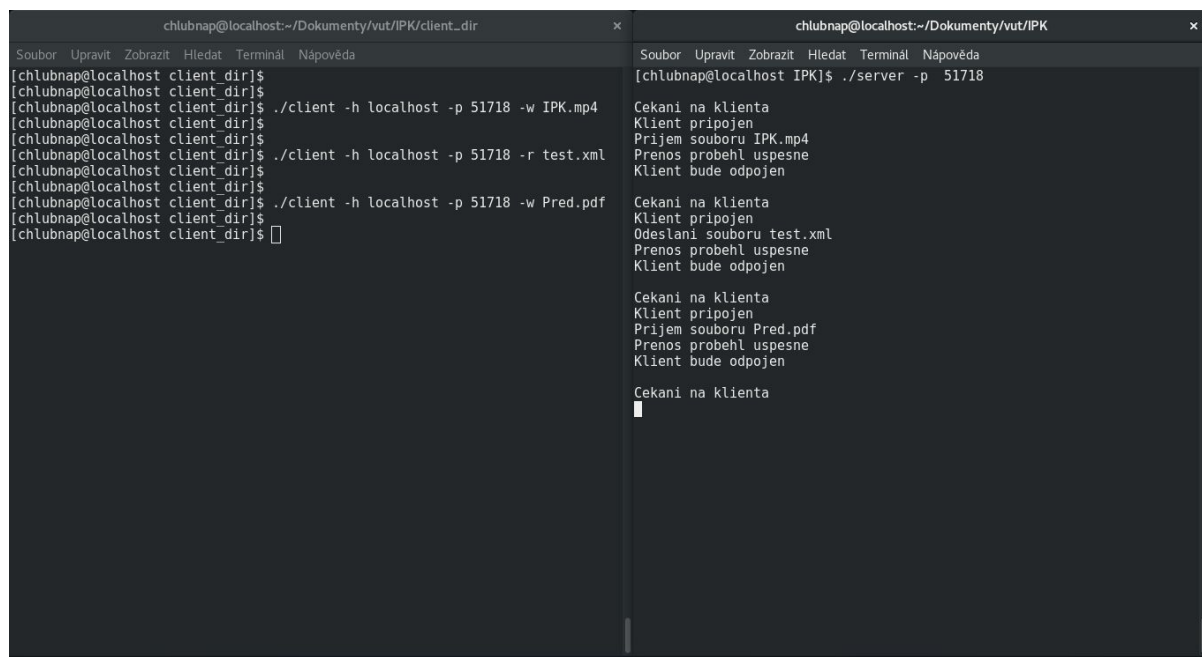
Dále program vypisuje chybovou hlášku pokud dojde k neočekávané chybě při vytváření socketu, nebo navazování spojení.

Program také zachytává chyby při zápisu nebo čtení ze souboru a při chybě při přenosu dat.

Funkce protokolu



Screenshot z testování



```
chlubnap@localhost:~/Dokumenty/vut/IPK/client_dir x
Soubor Upravit Zobrazit Hledat Terminál Nápověda
[chlubnap@localhost client_dir]$
[chlubnap@localhost client_dir]$
[chlubnap@localhost client_dir]$ ./client -h localhost -p 51718 -w IPK.mp4
[chlubnap@localhost client_dir]$
[chlubnap@localhost client_dir]$ ./client -h localhost -p 51718 -r test.xml
[chlubnap@localhost client_dir]$
[chlubnap@localhost client_dir]$ ./client -h localhost -p 51718 -w Pred.pdf
[chlubnap@localhost client_dir]$
[chlubnap@localhost client_dir]$
```

```
chlubnap@localhost:~/Dokumenty/vut/IPK x
Soubor Upravit Zobrazit Hledat Terminál Nápověda
[chlubnap@localhost IPK]$ ./server -p 51718
Cekani na klienta
Klient pripojen
Prijem souboru IPK.mp4
Prenos probehl uspesne
Klient bude odpojen
Cekani na klienta
Klient pripojen
Odeslani souboru test.xml
Prenos probehl uspesne
Klient bude odpojen
Cekani na klienta
Klient pripojen
Prijem souboru Pred.pdf
Prenos probehl uspesne
Klient bude odpojen
Cekani na klienta
█
```

```
[chlubnap@localhost client_dir]$ ./client -h localhost -p 51718 -k Pred.pdf
Chyba v argumentech
[chlubnap@localhost client_dir]$
[chlubnap@localhost client_dir]$ ./client -h localhost -p 517a18 -w Pred.pdf
Spatne zadany port
```

Zdroje

Při práci na projektu jsem čerpal z článku na stránce Linuxhowtos (odkaz níže), zabývajícím se TCP, prací se sockety a navázáním spojení. Další znalosti jsem hledal ve slidech z přednášek.

Opravy a rady k případným problémům jsem hledal na fórech jako například stackoverflow nebo již dříve zmíněný Linuxhowtos.

Dokumentace byla psána v Google docs a automat jsem vytvořil pomocí stránky draw.io.

Odkazy

<https://www.draw.io>

http://www.linuxhowtos.org/C_C++/socket.htm

<https://stackoverflow.com>