

# VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Dokumentácia k projektu z predmetu ISA  
TFTP klient

15. listopadu 2021

Kristián Královič

# Obsah

<b>1</b>	<b>Úvod</b>	<b>2</b>
<b>2</b>	<b>Implementácia</b>	<b>2</b>
2.1	Základné informácie . . . . .	2
2.2	Popis programu . . . . .	2
2.3	Začiatok práce programu a spracovanie parametrov . . . . .	2
2.4	Vytvorenie socketu . . . . .	3
2.5	St'ahovanie súboru zo serveru . . . . .	3
2.5.1	St'ahovanie súboru bez options . . . . .	3
2.5.2	St'ahovanie súboru s options . . . . .	3
2.6	Odosielanie súboru na server . . . . .	3
2.6.1	Odosielanie súboru s options . . . . .	3
2.6.2	Odosielanie súboru bez options . . . . .	3
2.7	Stratégia spracovania options . . . . .	4
<b>3</b>	<b>Preklad a spustenie programu</b>	<b>4</b>
<b>4</b>	<b>Záver</b>	<b>4</b>
<b>5</b>	<b>Použitá literatúra</b>	<b>5</b>

# 1 Úvod

Cieľom projektu bolo vytvoriť klientsku časť TFTP protokolu v jazyku C/C++. Tento klient má byť schopný sťahovať a ukladať súbory na server. Klient má taktiež podporovať rozšírenia `blocksize option`, ktorý umožňuje užívateľovi ovládať veľkosť dátových paketov, `timeout option`, ktorý umožňuje užívateľovi požadovať od serveru nastavenie timeoutu, `transfersize option`, ktorý slúži na odoslanie/získanie veľkosti prenášaného súboru a `multicast option`, ktorý umožňuje odosielanie/prijímanie dát cez multicast. Moja implementácia podporuje všetky zmienené rozšírenia okrem `multicast option`. Program po spustení a zadaní vstupných parametrov odošle žiadosť na server a spracuje pakety, ktoré obdrží od serveru. Program o všetkom informuje užívateľa výpisom na `stdout`, ktorý sa skladá z časového razítka a informácie o stave programu.

## 2 Implementácia

### 2.1 Základné informácie

Trivial File Transfer Protocol je protokol, ktorý umožňuje prenos súborov po sieti a je to zjednodušená verzia FTP (File Transfer Protocol). K prenosu dát sa používa Internet User Datagram protocol (UDP). TFTP protokol podporuje 3 módy prenosu súboru a to `octet`, ktorý prenáša súbor po bajtoch, `netascii`, ktorý upravuje prenášané dáta podľa normy RFC 854 a mód `mail`, ktorý je zastaralý a môj klient daný mód nepodporuje. Prenos dát prebieha nasledovne. Ako prvé odošle klient paket, ktorý špecifikuje typ prenosu dát pomocou príslušného `OPCODE`, ktorý nadobúda hodnoty 01 pri žiadosti o stiahnutie súboru a 02 pri žiadosti o zápis súboru na server, názov prenášaného súboru za ktorým nasleduje mód prenosu dát a prípadne options spomínané v úvode. Následne pokračuje prenos samotných dát v paketoch s `OPCODE` 03. Aby sa zamedzilo strate dát pri prenose, musí byť každý dátový paket potvrdený paketom ACK, ktorý obsahuje `OPCODE` 04 a príslušné číslo dátového paketu, ktorý potvrdzuje. V prípade, že nastane chyba pri prenose dát a nie je možné ďalej pokračovať s prenosom, server odošle ERROR paket s `OPCODE` 05 a prenos sa ukončí. V prípade, že požadujeme od servera options server odpovedá paketom s `OPCODE` 06, ktorý obsahuje hodnoty options na ktoré server pristúpil alebo navrhuje nové hodnoty. V prípade, že server zamietol options a požaduje ukončenie spojenia odošle paket s `OPCODE` 08. Prenos dát končí obdržaním/odoslaním menšieho množstva dát ako je `blocksize option` alebo menej ako 512bytov.

### 2.2 Popis programu

Celý kód je uložený v súbore `mytftpclient.cpp` a `mytftpclient.hpp`. Pre tvorbu paketu a prácu s ním používam knihovňu `sys/socket.h` a hlavičky `netinet`, `arpa`. Kód je rozdelený do 12 pomocných funkcií ktorých jednotlivé funkcie budú uvedené ďalej v dokumentácii.

### 2.3 Začiatok práce programu a spracovanie parametrov

Hlavná časť programu sa nachádza vo funkcii `main()`. Ako prvé program načíta vstupné parametre a skontroluje či nebol zadaný parameter `exit`, ktorý slúži na ukončenie programu. Následne je `string` obsahujúci vstupné parametre rozdelený pomocou funkcie `Split_arguments()` do vektora, ktorý obsahuje jednotlivé vstupné parametre a ich hodnoty. Následne je vektor odoslaný do funkcie `Check_arguments()`, ktorá skontroluje správnosť vstupných parametrov a vyplní štruktúru `Parameters`. Štruktúra `Parameters` obsahuje hodnoty jednotlivých vstupných parametrov a ďalšie pomocné premenné potrebné pre vykonávanie programu. V prípade, že boli zadané chybné hodnoty parametrov program vypíše chybovú hlášku a požaduje opätovné zadanie parametrov.

## 2.4 Vytvorenie socketu

Pre vytvorenie socketu používam funkcie `getaddrinfo()`, ktorá zistí potrebné informácie pre vytvorenie socketu a `socket()`, ktorá vytvorí socket. Následne program zistí pomocou funkcie `ioctl()` najnižšie MTU zo všetkých rozhraní. Hodnota MTU je potrebná pri `blocksize` option.

## 2.5 Sťahovanie súboru zo serveru

Sťahovanie súboru začína vytvorením RRQ. Na vytvorenie žiadosti slúži funkcia `Create_request()`, ktorá vytvorí paket byte po byte a postupne do paketu ukladá potrebné hodnoty a v prípade zadania parametrov `-s`, `-t` aj príslušné options a ich hodnoty. Následne je paket odoslaný pomocou `sendto()` na server.

### 2.5.1 Sťahovanie súboru bez options

Program očakáva dátový paket od serveru, ten následne ukladá do premennej `recvbuffer`. Obsah bufferu je následne zapísaný do súboru s názvom, ktorý program obdržal v parametre `-d`. V prípade, že bol zadán mód prenosu `netascii`, je zavolaná funkcia `Convert_from_netascii()`. Funkcia upraví obdržané dáta, ktoré sú následne zapísané do súboru. Ak už daný súbor existuje bude prepísaný. Zakaždým obdržaným paketom odosiela klient ACK. V prípade, že klient odoslal ACK paket ale neodbržal ďalší dátový paket nastane timeout s hodnotou 3 sekundy. Po tomto časovom úseku sa klient pokúsi znova odoslať paket. Klient sa pokúsi paket odoslať 3 krát ak ani tak neodbrží nový dátový paket, ukončí sa prenos chybou.

### 2.5.2 Sťahovanie súboru s options

Program spracuje OACK paket vo funkcii `Handle_OACK()` a odošle ACK paket, ktorý potvrdzuje OACK. Prenos pokračuje rovnakým spôsobom ako sťahovanie súboru bez options.

## 2.6 Odosielanie súboru na server

Odosielanie súboru začína zistením veľkosti súboru, ktorý bude odosielať na server pomocou funkcie `Calculate_size()`. Veľkosť súboru je potrebná pre `tsize` option, ktorý má byť prítomný v každej žiadosti. Ďalej program pokračuje vytvorením WRQ. Na vytvorenie žiadosti slúži funkcia `Create_request()`, ktorá postupne vytvorí paket. Následne postupne do paketu ukladá potrebné hodnoty a v prípade zadania parametrov `-s`, `-t` aj príslušné options a ich hodnoty. Následne je paket odoslaný pomocou `sendto()` na server.

### 2.6.1 Odosielanie súboru s options

Program očakáva ACK paket a po jeho obdržaní začína prenos súboru. V prípade, že bol zadán mód prenosu `netascii` sú dáta upravené podľa normy RFC 854. Zakaždým odoslaným dátovým paketom očakáva potvrdzovací (ACK) paket. V prípade, že program neodbrží potvrdzovací paket nastane timeout na 3 sekundy. Po uplynutí časového intervalu sa program pokúsi znova odoslať dáta. Proces sa opakuje trikrát a ak ani po poslednom pokuse neodbrží správny potvrdzovací paket, prenos sa ukončí chybou.

### 2.6.2 Odosielanie súboru bez options

Program očakáva OACK paket, ktorý následne spracuje vo funkcii `Handle_OACK()`. V prípade, že nenastala chyba klient začína odosielať dáta rovnakým spôsobom ako pri odosielaní súboru bez options.

## 2.7 Stratégia spracovania options

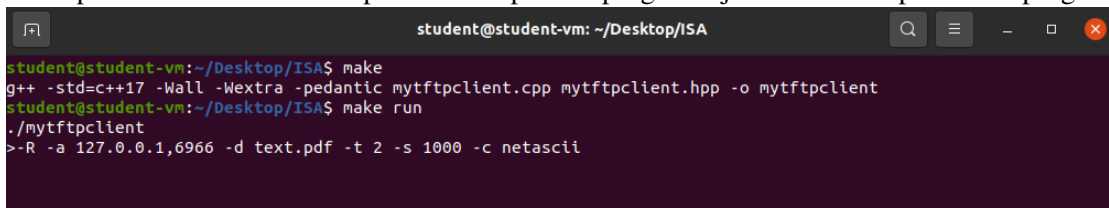
Program podporuje 3 typy options a to `transfer size`, `block size` a `timeout`. Transfer size je prítomný v každej žiadosti. V prípade, že server odpovedal na tento option je jeho hodnota použitá pri výpise informácii o prenose. Ak server neprijal tento option je vo výpise použité X.

Block size option sa nastavuje pomocou parametra `-s`. Povolené hodnoty sú od 8 do 65464. V prípade, že užívateľ zadal väčšiu hodnotu ako je MTU tak sa použije hodnota MTU. Ak server odpovedal na žiadosť skontroluje sa hodnota odpovede serveru. Ak server odoslal rovnakú hodnotu ako klient tak prenos súboru prebieha normálne. Ak server navrhol novú hodnotu použije sa táto nová hodnota.

Timeout option sa nastavuje pomocou parametra `-t`. V prípade, že server prijal hodnotu prenos pokračuje s touto hodnotou. V prípade, že server neprijal option pokračuje prenos v základnom nastavení. Ak server odoslal novú hodnotu timeout option tak sa prenos súboru ukončí chybou.

## 3 Preklad a spustenie programu

Program sa prekladá pomocou Makefile. Pre preklad slúži príkaz `make`. Spustenie programu je možné pomocou príkazu `make run`. Po preložení a spustení programu je nutné zadať parametre programu.



```
student@student-vm: ~/Desktop/ISA
student@student-vm:~/Desktop/ISA$ make
g++ -std=c++17 -Wall -Wextra -pedantic mytftpclient.cpp mytftpclient.hpp -o mytftpclient
student@student-vm:~/Desktop/ISA$ make run
./mytftpclient
>-R -a 127.0.0.1,6966 -d text.pdf -t 2 -s 1000 -c netascii
```

Príklad prekladu a spustenia programu

## 4 Záver

Projekt bol mojím prvým stretnutím s programovaním sieťových aplikácií v jazyku C++. Bol to taktiež môj prvý veľký projekt v tomto jazyku. Dozvedel som sa mnoho nových informácií nielen o práci s paketom, ale aj o tvorbe socketu a vytvorení aplikácie, ktorá funguje aj pre IPv6. Naučil som sa ako funguje TFTP protokol a a jeho možnú implementáciu. Podarilo sa mi vypracovať všetky options okrem multicast option. Pri testovaní som nenarazil na žiadne chybné chovanie. Projekt som testoval na viacerých implementáciach TFTP serveru metódou porovnávania vstupu a výstupu ako aj kontrola packetu v programe Wireshark.

## 5 Použitá literatúra

Pri tvorbe programu som pracoval s nasledujúcimi zdrojmi :

1. [RFC0854] Postel, J. and J. Reynolds, "Telnet Protocol Specification", STD 8, RFC 854, DOI 10.17487/RFC0854, May 1983, <<https://www.rfc-editor.org/info/rfc854>>.
2. [RFC1350] Sollins, K., "The TFTP Protocol (Revision 2)", STD 33, RFC 1350, DOI 10.17487/RFC1350, July 1992, <<https://www.rfc-editor.org/info/rfc1350>>.
3. [RFC2347] Malkin, G. and A. Harkin, "TFTP Option Extension", RFC 2347, DOI 10.17487/RFC2347, May 1998, <<https://www.rfc-editor.org/info/rfc2347>>.
4. [RFC2348] Malkin, G. and A. Harkin, "TFTP Blocksize Option", RFC 2348, DOI 10.17487/RFC2348, May 1998, <<https://www.rfc-editor.org/info/rfc2348>>.
5. [RFC2349] Malkin, G. and A. Harkin, "TFTP Timeout Interval and Transfer Size Options", RFC 2349, DOI 10.17487/RFC2349, May 1998, <<https://www.rfc-editor.org/info/rfc2349>>.
6. Jorgensen, H. B. "B. (2009). Beej's Guide to Network Programming: Using Internet sockets. Beej's Guide to Network Programming. Retrieved November 14, 2021, from <https://beej.us/guide/bgnet/html/>.
7. C++ split string by space into vector. SlayStudy. (2020, June 20). Retrieved November 14, 2021, Dostupné z: <https://slaystudy.com/c-split-string-by-space-into-vector/>.
8. How to get the size of a file in C++ - using file handling. CodeSpeedy. (2019, June 24). Retrieved November 14, 2021, from <https://www.codespeedy.com/cpp-program-to-get-the-size-of-a-file/>.