Vysoké učení technické v Brně
Fakulta informačních technologií

SÍŤOVÉ APLIKACE A SPRÁVA SÍTÍ

2021/2022

Projekt ISA
Varianta termínu – TFTP Klient

Tomáš Milostný (xmilos02)

Obsah

Úvod

Projekt je implementovaný v jazyce **C++**. Výsledný program je implementací klienta protokolu TFTP[1] s podporou *Option Extension*[2]. Oproti původnímu zadání nebyla implementována multicastová komunikace.

Popis implementace

Jádrem projektu je hlavní program v souboru `mytftpclient.cpp`. Program začíná spuštěním vlastní příkazové řádky, která uživateli umožňuje zadat příkazy programu na standardním vstupu. Následně je tento vstup od uživatele zpracován třídou `ArgumentParser` předáním zadaného řetězce jejímu konstruktoru.

ArgumentParser může vyhazovat výjimky, které jsou tedy ošetřeny v try-catch bloku.
V tomto případě je na standardní chybový výstup vypsána zpráva výjimky a příkazová řádka je znovu spuštěna bez pokus o zahájení TFTP přenosu.

Pokud jsou všechny argumenty příkazové řádky v pořádku, pokračuje se vytvořením instance třídy `Tftp` předáním argumentů jejímu konstruktoru. Zde dojde k pokusu o vytvoření souboru (v případě čtení ze serveru) nebo otevření souboru pro čtení (v případě zápisu na server). Pokud se nepodaří soubor otevřít, dojde k vyhození výjimky, jejíž chyba je opět vypsána na standardní chybový výstup a dojde k opětovnému navrácení k příkazové řádce bez zahájení přenosu.

V případě úspěšného pokusu o otevření souboru, je zahájen přenos voláním metody `Transfer`. Po dokončení přenosu dojde k uvolnění paměti používané objekty typu `ArgumentParser` a `Tftp` a opětovnému návratu k příkazové řádce. Oproti původnímu zadání byly přidány příkaz **quit** a **exit** pro ukončení programu a **help** pro zobrazení nápovědy k argumentům podporovaných tímto projektem.

ArgumentParser

Třída, která slouží ke zpracování argumentů příkazové řádky zadané uživatelem programu a jejich uložení v atributech této třídy. Pro zpracování argumentů je použita funkce **getopt**, která počítá se standardním vstupním bodem jazyka C/C++ a s argumenty programu uložených v poli **argv** a jejich počet v proměnné **argc**. Proto je řetězec ze standardního vstupu ještě předzpracován do tohoto formátu.

V případě chyby od getopt, neplatné hodnoty pro daný argument nebo chybějící povinný argument má za následek vyvolání výjimky `std::invalid_argument` s odpovídající chybovou zprávou, která je vypsána na vyšší vrstvě během zachytávání této výjimky.

Program pracuje s následujícími argumenty:

- **-R**: Nastavit režim čtení ze serveru. (povinný, nelze kombinovat s argumentem -W)
- **-W**: Nastavit režim zápisu na server. (povinný, nelze kombinovat s argumentem -R)
- **-d <filename>**: Cílový soubor, ze kterého se čte nebo je zapisován na server. (povinný)
- **-t <timeout>**: Časový limit v sekundách.
- **-s <size>:** Velikost bloku v datovém paketu TFTP. (výchozí: 512 B)
- **-m:** Přepnutí do režimu multicastu.
- **-c <mode>:** Mód přenosu (**octet**/**binary** nebo **ascii**/**netascii**, výchozí: octet)
- **-a <address>,<port>:** IPv4 nebo IPv6 adresa a port serveru. (výchozí: 127.0.0.1,69)

Tftp

Třída implementující klientskou komunikaci protokolem TFTP[2]. Parametrem konstruktoru je instance třídy `ArgumentParser`, která obsahuje atributy nastavení přenosu.

Veřejnou metodou `Tranfer` je po inicializaci BSD schránky zahájena komunikace se serverem voláním metody `Request`. Pokud je nastaven režim zápisu na server, je předtím soubor otevřen ke čtení, pokud neexistuje, skončí se s výjimkou před zahájením komunikace. Jinak je soubor pro zápis dat ze serveru otevřen po zahájení komunikace, server mohl v chybovém packetu ohlásit, že daný soubor neexistuje, a `Request` skončit vyhozením výjimky. Pokud bylo zahájení komunikace úspěšné, následuje buď příjem dat od serveru, nebo odesílání dat od klienta.

`Request` začíná dynamickým vytvořením TFTP Request packetu, kam jsou na konec nakopírována potřebná zadaná nastavení. Možnost **tsize**[3] je zadána pouze pro binární přenosy, **timeout**[3] a **blksize**[4], pokud jsou nastaveny dané `ArgumentParser` atributy *Timeout* a *Size*. Následuje příjem odpovědi od serveru. V případě přijetí chybového packetu, je vyhozena výjimka se zprávou od serveru. Jinak je pro binární přenos načtena hodnota **tsize**, která je návratovou hodnotou této metody.

Odesílání dat probíhá v režimu zápisu na server voláním metody `SendDataBlock` a příjem dat probíhá v režimu čtení ze serveru voláním metody `ReceiveData`. Obě s parametrem celkové velikosti souboru, získané z odpovědi serveru při zahájení komunikace.

Hlavní smyčka metody `SendDataBlock` začíná inkrementací aktuálního bloku dat k odeslání a dynamickým vytvořením datového packetu k odeslání. Následně přečte část zdrojového souboru, podle velikosti dat k odeslání a odešle na server. Nakonec čeká na ACK packet v odpovědi od serveru. Při čekání na odpověď může dojít k vypršení časového limitu, pokud je nastaven. Zde také může nastat situace přijetí packetu s jiným operačním kódem než ACK, v tomto případě končí přenos s chybou. Když nenastala žádná chyba, přenos končí, pokud byla klientem odeslána všechna data.

Metoda `ReceiveData` začíná vytvořením bufferu dle zadané velikosti, 4 B pro operační kód a číslo bloku + velikost v B nastavená v `ArgumentParser` atributu *Size* (výchozí 512 B). Poté začíná svou hlavní smyčku odesláním ACK packetu s číslem aktuálně přijatého bloku dat (na začátku přenosu 0 značí žádost o první blok dat). Poté probíhá samotný příjem dat, kde je zkontrolováno číslo přijatého bloku dat, pokud se toto číslo nerovná očekávanému číslu bloku, číslo aktuálního bloku se dekrementuje (značí opětovné potvrzení posledních přijatých dat) a vrací se k odeslání ACK packetu. Jestliže je přijatý packet v pořádku, jsou obsažená data zapsána do cílového souboru. Přenos končí v momentě, kdy velikost přijatého packetu je menší než velikost bufferu. Příjem posledních dat je před návratem z volání potvrzen odesláním ACK packetu.

StampMessagePrinter

Pomocná třída se statickými metodami pro výpisy s časovým razítkem (`Print`, pro výpis na standardní výstup, a `PrintError`, pro výpis na standardní chybový výstup). Uvnitř těchto metod je volána privátní metoda `PrintWithTimeStamp`, která přijímá krom samotné zprávy i parametr výstup, na kterém má být vypsána.

Seznam literatury

[1] *RFC 1350: THE TFTP PROTOCOL (REVISION 2)* [online]. [cit. 2021-11-15]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc1350

[2] *RFC 2347: TFTP Option Extension* [online]. [cit. 2021-11-15]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc2347

[3] *RFC 2349: TFTP Timeout Interval and Transfer Size Options* [online]. [cit. 2021-11-15]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc2349

[4] *RFC 2348: TFTP Blocksize Option* [online]. [cit. 2021-11-15]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc2348