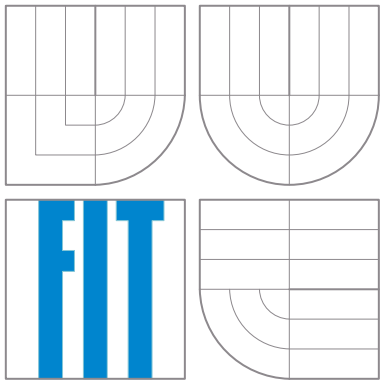

Dokumentace k projektu pro předmět ISA

TFTP server

28. listopadu 2014

Autor: Jan Kafka, xxkafka00@stud.fit.vutbr.cz
Fakulta Informačních Technologií
Vysoké Učení Technické v Brně

Obsah

1 Úvod **1**

2 Požadované vlastnosti dle RFC **1**
2.1 Typy paketů . . . 1
2.1.1 RRQ . . . 1
2.1.2 WRQ . . . 1
2.1.3 DATA . . . 1
2.1.4 ACK . . . 2
2.1.5 ERROR . . . 2
2.2 Rozšíření . . . 2
2.2.1 blocksize . . . 2
2.2.2 timeout . . . 2
2.2.3 transport size . . . 2

3 Popis implementace **3**
3.1 Koncept . . . 3
3.2 Zpracování parametrů . . . 3
3.3 Služby . . . 3
3.4 Obsluha klienta . . . 3
3.5 Operace se soubory . . . 4

4 Spuštění serveru **4**
4.1 Parametr -d . . . 4
4.2 Parametr -a . . . 4
4.3 Parametr -s . . . 4
4.4 Parametr -t . . . 4
4.5 Příklad spuštění . . . 4

1 Úvod

Úkolem bylo vytvořit konkurentní TFTPv2 server s rozšířeními dle specifikací RFC. Server má vhodnou formou uvědomovat uživatele o své činnosti, např. výpisem obsahující časové razítko, IP adresu klienta a rozbor zprávy. Forma výstupu nebyla přesně specifikována. Nebyla vyžadována podpora mail-módu a multicast option byl po čase změněn na bonusové rozšíření.

2 Požadované vlastnosti dle RFC

Základní protokol TFTP popisuje RFC 1350 [1](revize 2, původní verze je z roku 1980). Protokol je implementován nad UDP protokolem. Pomocí TFTP je možné číst soubory ze serveru, nebo zapisovat soubory na server. Na rozdíl od FTP postrádá některé funkce, např. neumožňuje výpis adresářů a nepodporuje ověření pravosti uživatelů.

2.1 Typy paketů

Klient a server spolu komunikují několika typy paketů. Pakety jsou rozlišovány podle čísla (opcode), které se nachází v prvních 2 bajtech. V paketech se mohou nacházet také další informace, které jsou odděleny nulovým bajtem.

2.1.1 RRQ

Paket s požadavkem na čtení (RRQ - read request) má opcode 1. Posílá jej klient, pokud chce číst nějaký soubor ze serveru. Paket obsahuje název souboru a mód přenosu (octet nebo netascii). V módu octet se uvažují veškerá data jako bajty a při přenosu nedojde k žádným změnám. V módu netascii se čtou data po ascii znacích. Některé escape sekvence se převádí na jiné a protistrana je převede na svůj formát. To umožňuje přenášet zejména textové soubory mezi různými platformami.

2.1.2 WRQ

Paket s požadavkem na zápis (WRQ - write request) má opcode 2. Posílá jej klient, pokud chce zapsat nějaký soubor na server. Paket obsahuje stejně jako v případě RRQ jméno souboru a mód přenosu.

2.1.3 DATA

Datový paket má opcode 3 a používá se k přenosu dat ze serveru nebo na server. Obsahuje číslo bloku a samotná data. Číslo je ve dvou bajtech a může být v rozmezí 1 až 65535. Zbytek paketu tvoří data. Ve standardní implementaci mají data délku max. 512 B. Soubor tedy musí být rozložen na bloky této velikosti. Pokud mají data menší délku, než 512 B, je to indikace, že datový paket je posledním.

2.1.4 ACK

Potvrzující paket (ACK - acknowledge) má opcode 4 a používá se k potvrzení přijetí datového paketu nebo k potvrzení zahájení přenosu. Obsahuje pouze opcode a číslo datového bloku ve 2 bajtech. Protože protokol UDP nezaručuje spolehlivý přenos paketů, potvrzuje se přijetí datového bloku před odesláním dalšího. V případě WRQ nejdříve server potvrdí žádost ACK paketem s číslem bolku 0 a poté začne klient zasílat datové pakety.

2.1.5 ERROR

Paket ohlašující chybu má opcode 5. Obsahuje kód chyby a textový řetězec. Může být zaslán po jakémkoliv paketu. Kód chyby má hodnotu od 0 do 7 a značí chybu konkrétního typu. Další podrobnosti o chybě obsahuje textový řetězec.

2.2 Rozšíření

V dokumentech RFC 1785, 2090, 2347, 2348, 2349 jsou popsána rozšíření, které je možné implementovat ke standardnímu TFTP serveru. Jde o volitelnou velikost datového bloku (blocksize), volitelný čas vypršení (timeout), oznámení/potvrzení velikosti souboru (transport size) a multicast. Poslední zmíněné rozšíření není v projektu implementováno. Každé rozšíření je potvrzeno dalším typem paketu OACK, který obsahuje informace o tom, jaká rozšíření bude server akceptovat.

2.2.1 blocksize

Klient může specifikovat, jak velký má být každý datový blok [4]. Standardní velikost je 512 B a klient může požádat o menší, nebo větší velikost. Minimální hodnota je 8 B a maximální 65464 B. Server může návrh akceptovat, případně hodnotu upravit a poslat klientovi potvrzující paket OACK. Velikosti požadované klientem nemusí být možné dosáhnout, např. kvůli omezení MTU rozhraní (maximum transmission unit). Nebo může být zadán parametr s velikostí bloku při spuštění serveru a ten ji považuje za maximální. Server proto může hodnotu upravit a klient se poté musí řídit výhradně parametry, které má od serveru potvrzeny.

2.2.2 timeout

Klient může specifikovat čas vypršení [5]. Server hodnotu potvrdí, nebo odmítne. Může se opět řídit podle parametru při spuštění nastavující maximální hodnotu timeoutu.

2.2.3 transport size

Klient se může prostřednictvím RRQ paketu a tohoto parametru dotázat na velikost souboru, který bude server posílat [5]. Může tak přenos stornovat ještě před počátkem (např. kvůli nedostatku místa). V případě WRQ paketu klient serveru oznámí velikost souboru, který chce zapsat a server může obdobným způsobem přenos odmítnout, nebo v OACK paketu poslat stejnou hodnotu, čímž potvrdí, že je schopen soubor přijmout.

3 Popis implementace

3.1 Koncept

Program je založen na nezávislé činnosti několika procesů. Procesy tvoří celkem tři úrovně. Obsluha klienta je proces na nejnižší úrovni, který vyřizuje aktuální požadavek klienta. Po vyřízení požadavku se ukončí. Proces služba běží na určitém síťovém rozhraní. Přijímá požadavky od klientů a vytváří nové obslužné procesy, které je poté obsluhují. Po vytvoření procesu obsluhy služba dále pokračuje v naslouchání dalším požadavkům. Hlavní proces je na nejvyšší úrovni a vytváří službu, nebo více služeb běžících na více rozhraní.
Program je napsán v jazyce C++ s využitím standardních systémových knihoven. Využívá objektově orientovaný návrh. Vzhledem ke konceptu (více procesů) se přímo nabízelo použití tříd a instancí objektů. Třídy jsou vytvořeny pro činnosti, které se mohou vícekrát opakovat a jejich instance mohou běžet zároveň jako samostatné procesy. Třídy jsou také využity pro případy, kde se jejich použití nabízí např. kvůli přehlednosti kódu.

3.2 Zpracování parametrů

Třída `options` řeší vše kolem načítání parametrů z příkazové řádky. Její metody např. kontrolují počet parametrů, řeší parsování adres, čísla portu, převod z řetězce na číslo a testují některé základní vlastnosti, např. zda zadaný adresář existuje. Po startu programu je zavolána metoda `load()`, která načte parametry a poté metoda `check()`, která kontroluje jejich vlastnosti. Pokud dojde k chybě, je program ukončen s chybovou hláškou.

3.3 Služby

Podle zadaných parametrů se spustí buď jedna, nebo více služeb na více rozhraních. Třída `service` představuje jednu službu. Obsahuje metodu `start()`. Té se předají potřebné parametry, vytvoří se nový socket s určitou adresou a portem. Poté přejde program do nekonečné smyčky, ve které naslouchá na rozhraní a příchozí požadavky předává obsluhám.
Pokud bylo zadáno více adres, vytvoří hlavní proces v cyklu více procesů pomocí funkce `fork()` a v každém procesu se vytvoří nová instance třídy `service`. Současně si uloží jejich PID pro pozdější korektní ukončení programu (uvolnění zdrojů, aby nezůstávaly v systému procesy ve stavu zombie).

3.4 Obsluha klienta

Pokud klient pošle požadavek službě, ta vytvoří nový proces opět pomocí funkce `fork()`, který bude požadavek obsluhovat. Třída `Client` a její metody slouží pro obsluhu jednoho klienta. V novém procesu se vytvoří nová instance této třídy a poté se vytvoří nový socket na novém portu. Na tomto socketu bude nadále klient komunikovat se serverem. Služba běžící v rodičovském procesu mezitím dále naslouchá dalším požadavkům. Po ukončení obsluhy se proces ukončí. Rodičovský proces může během své činnosti vytvořit nespočet nových procesů (obsluh). Jejich PID si pamatuje a při ukončování programu (zaslání signálu SIGINT hlavním procesem) postupně projde jejich seznam a uvolní jejich zdroje.

3.5 Operace se soubory

Třída `File` a její metody řeší veškeré operace spojené s čtením, nebo zápisem dat do souboru. Např. funkce pro načítání bloku dat určité velikosti a od určité pozice, funkce pro převod z/na formát netascii, funkce pro ukládání bloku dat do souboru, funkce pro zjištění velikosti souboru apod. Instanci této třídy si vytváří nezávisle každý proces obsluhy.

4 Spuštění serveru

Server se spouští s několika parametry, které nastavují jeho vlastnosti.

4.1 Parametr -d

Parametr `-d` nastavuje adresář, ze kterého bude server soubory číst, nebo do kterého je bude zapisovat. Adresář může být zadán jako absolutní, nebo relativní cesta. Např. `-d /home/jan/sources` nebo `-d ./sources`. Parametr je povinný. Pokud není zadán, server vypíše chybu. Pokud zadán je, nejdříve se zkontroluje jeho existence a zda má k němu server přístup. Pokud není něco v pořádku, je rovněž vypsána chyba.

4.2 Parametr -a

Parametr `-a` nastavuje adresu a port, nebo více adres a portů, na kterých server poběží. Adresa a port se zapisuje jako dvojice a jsou odděleny čárkou. Dvojice lze zapisovat za sebe a oddělovat mřížkou. Např. `-a 127.0.0.1,1300` nebo `127.0.0.1,1300#192.168.1.5,1400`. Parametr není povinný. Pokud není zadán, běží pouze jedna instance serveru na adrese 127.0.0.1 a portu 69.

4.3 Parametr -s

Parametr `-s` nastavuje maximální velikost bloku, kterou bude server od klienta akceptovat. Hodnota se zadává v rozmezí 8 - 65464 bajtů, což jsou i krajní hodnoty definované v RFC. Parametr není povinný. Pokud není zadán, je buď uvažována standardní velikost bloku 512 B, nebo v případě žádosti od klienta je uvažována velikost MTU rozhraní - 32 B.

4.4 Parametr -t

Parametr `-t` nastavuje čas vypršení (timeout) v sekundách, který bude server od klienta akceptovat. Parametr není povinný. Pokud není zadán, je uvažována výchozí hodnota 3 s.

4.5 Příklad spuštění

Příklad, jak může vypadat spuštění serveru, který poběží na dvou rozhraních, bude číst/zapisovat do aktuálního adresáře a bude akceptovat datový blok o velikosti 1024 B a timeout 10 s.

```
./tftpserver -a 127.0.0.1,1300#192.168.1.5,1400 -d ./ -s 1024 -t 10
```

Reference

[1] SOLLINS, K. THE TFTP PROTOCOL (REVISION 2). *RFC-Editor Webpage* [online]. 1992 [cit. 2014-11-27]. Dostupné z: https://tools.ietf.org/html/rfc1350

[2] MALKIN, G. a A. HARKIN. TFTP Option Negotiation Analysis. *RFC-Editor Webpage* [online]. 1995 [cit. 2014-11-27]. Dostupné z: https://tools.ietf.org/html/rfc1785

[3] MALKIN, G. a A. HARKIN. TFTP Option Extension. *RFC-Editor Webpage* [online]. 1998 [cit. 2014-11-27]. Dostupné z: https://tools.ietf.org/html/rfc2347

[4] MALKIN, G. a A. HARKIN. TFTP Blocksize Option. *RFC-Editor Webpage* [online]. 1998 [cit. 2014-11-27]. Dostupné z: https://tools.ietf.org/html/rfc2348

[5] MALKIN, G. a A. HARKIN. TFTP Timeout Interval and Transfer Size Options. *RFC-Editor Webpage* [online]. 1998 [cit. 2014-11-27]. Dostupné z: https://tools.ietf.org/html/rfc2349