Západočeská univerzita v Plzni

Fakulta Aplikovaných věd

Katedra informatiky a výpočetní techniky

FTP Server

Samostatná práce z předmětu

Úvod do počítačových sítí

Leden 2008 Martin Sloup, A04372, msloup@students.zcu.cz

Martin Sloup, A04372
msloup@students.zcu.cz

Obsah

Martin Sloup, A04372
msloup@students.zcu.cz

1. Zadání semestrální práce

Napište program, který se bude chovat jako FTP server. Zvolte množinu příkazů z RFC pro FTP server, které naimplementujete. Server by měl podporovat minimáně pasivní režim a přihlašování na základě uživatelského jména a hesla. Server by měl umožnit obsloužit více klientů najednou. Aplikace bude primárně provozována na operačním systém Linux.

2. Analýza problému

2.1. Různé postřehy

Jako programovací jazyk jsem se rozhodl použít **Javu**, která o proti jazyku C umí lépe pracovat s textem. A taktéž mé znalosti Javy jsou na vyšší úrovni než je úroveň znalosti jazyka C.

Při zkoumání různých materiálů jsem narazil na stránku[1], kde je uvedeno, jaké aktuální příkazy, parametry příkazů a návratové kódy se aktuálně používají. FTP protokol je velmi starý. Je definován ve velkém množství **RFC**. Plno příkazů, či parametrů se již dávno nepoužívá, takže by asi nebylo od věci implementovat jen takovou sadu příkazů, která se aktuálně používá. Proto jsem se řídil výhradně touto stránkou a občas nahlédl také do RFC FTP protokolu.

Program je řešen multivláknově, tak aby mohl obsloužit několik klientů najednou. Samotný hlavní cyklus serveru (naslouchání) běží na hlavním vlákně programu. Pokud se připojí nový uživatel, tak je vytvořeno vlákno a na něm probíhá další komunikace. Pro datové spojení se používá další vlákno. To umožňuje, aby během komunikace mohl klient stále komunikovat na příkazovém spojení.

Dle zjištění, že příkaz **LIST** v FTP protokolu není standardizován jsem se rozhodl implementovat základní formát linuxového /bin/ls, tak jak ho má naimplmentována většína FTP serverů.

Potřeboval jsem nějakým způsobem vyřešit vypnutí serveru. A protože server běží v konzolovém režimu, nabízí se zkratka **CTRL+C**. Původně jsem si bláhově myslel, že se šikovně spusti Garbage Collector a já dokážu odchytit destrukci objektů a provést případné zavření. Jenže na tohle není spoleh. Proto jsem se rozhodl jít cestou, kdy vytvořím nové vlákno, které se spustí při odchycení **CTRL+C**. Toto vlákno postupně spustí u inicializovaných objektů funkci dispose(). Ta uzavře všechny použité prostředky u tříd. Celá tato „šaškárna“ byla nutná provést, protože jsem potřeboval do logu zanést, že se uzavřely všechny sockety a aplikace se ukončila.

Během implementace jsem se rozhodoval, jak moc FTP protokol implementovat. Použil jsem většinu FTP příkazů uvedené na stránce „**FTP: File Transfer Protocol**“[3]. Později jsem při testování doimplementoval další příkazy, které jsem uznal za vhodné(OPTS UTF8, MDTM, SIZE, FEAT, ...).

2.2. Režimy FTP přenosu

V základu jsem implementoval příkaz **PASV**, který je povinný dle RFC. Tento příkaz vrací připojenému klientovi informace o IP adrese a portu na serveru, na kterém bude navázána datová komunikace. Spojení bude tedy pro klienta pasivní.

1 D. J. Bernstein - FTP: File Transfer Protocol - http://cr.yp.to/ftp.html

Martin Sloup, A04372
msloup@students.zcu.cz

Ještě existuje druhý režim přenosu, aktivní. Klient pošle serveru příkaz **PORT** s parametry určující, na kterou IP adresu a port se ma server připojit k provedení datové komunikace.

Jeden z těchto dvou příkazu je nutné vykonat před příkazy **LIST**, **RETR**, **STOR**, **APPE**, atd, aby bylo otevřeno datové spojení, které používají tyto příkazy.

2.3. Přehled implementovaného FTP protokolu

2.3.1. Příkazy FTP protokolu

- **USER** – uživatelské jméno připojeného klienta
- **PASS** – heslo připojeného klienta, dokončí proces přihlášení
- **TYPE** – specifikuje režim serveru, textový nebo binární
- **STRU** – nastavuje strukturu přenosu v datovém spojení, aktuálně se používá jen s prametrem F – file (bez struktury), jiné parametry se již nepoužívají
- **MODE** – nastavuje mod přenosu, parametr S – stream (proud dat), ještě se používá Z[1] – komprimovaný přenos pomocí zlib, ale ten jsem neimplementoval
- **CWD**, **XCWD** – změna aktuálního pracovního adresáře
- **PWD**, **XPWD** – vrátí aktuální pracovní adresář
- **CDUP**, **XCUP** – vratí se o adresář víšše
- **PASV** – nastavuje pasivní režim přenosu, server vrátí IP a port, kde naslouchá
- **PORT** – nastavuje aktivní režim přenosu, klient posílá IP a port kam se má server připojit
- **RETR** – v datovém spojení vrací zadaný soubor
- **REST** – nastavuje pozici v soboru, od které se bude u příkazu **RETR** vracet data, u **STOR** nastavuje, od jaké pozice se budou data v souboru nahrávat.
- **LIST** – vrací v datovém spojení výpis aktuálního pracovního adresáře nebo adresáře zadaného přes parametr
- **NLST** – vrací v datovém spojení seznam adresářů v aktuálním pracovním adresáři nebo v adresáři zadaného přes parametr, Formát odpovědi je název položky následovaný novým řádkem
- **QUIT** – ukončuje spojení klienta
- **SYST** – vrací identifikaci serveru
- **NOOP** – příkaz udržující spojení, neprovádí prakticky nic
- **STOR** – přes datové spojení nahrává zadaný soubor na server
- **APPE** – přes datové spojení nahrává zadaný soubor na server na konec již existujiciho souboru

1 Návod, jak naimplementovat MODE Z: http://www.g6ftpserver.com/en/modez

Martin Sloup, A04372
msloup@students.zcu.cz

- **MKD**, **XMKD** – vytváří zadaný adresář
- **RMD**, **XRMD** – maže zadaný adresář
- **DELE** – maže zadaný soubor
- **RNFR** – nastavuje objekt, který se bude přejmenovávat
- **RNTO** – přejmenuje nastavený objekt (příkaz **RNFR**) na nový název zadaný přes parametr
- **CLNT** – zaznamená User Agent připojeného klienta
- **SIZE** – vrátí velikost zadaného souboru
- **MDTM** – vrátí nebo nastaví poslední čas modifikace souboru
- **FEAT** – vrátí seznam nadstandardních příkazů
- **OPTS UTF8** – aktivuje nebo deaktivoje kódovou stránku příkazového spojení na UTF8

2.3.2. Návratové kódy FTP protokolu

- **150** – otevřeno datové spojení
- **200** – příkaz úspěšně vykonán
- **202** – příkaz není naimplementován
- **213** – informace o souboru (velikost, datum, atd.)
- **215** – typ sytému
- **221** – úspěšné odhlášení uživatele
- **226** – uzavřeno datové spojení, přenos byl úspěšný
- **227** – úspěšně vytvořeno pasivní spojení, vrací IP a port na serveru, kde probíhá naslouchání datového spojení
- **230** – klient úspěšně přihlášen
- **331** – uživatelské jméno přijato, čeká se na heslo
- **250** – akce, prováděná v souborovém systému úspěšně dokončena (vytvoření a mazání adresáře, přejmenování, atd.)
- **257** – vrací aktuální pracovní adresář
- **350** – informace o souboru přijata, čeká se na další akci pro dokončení (příkazy **REST**, **RNFR**, ...)
- **425** – nepovedlo se otevřít datové spojení
- **426** – datové spojení bylo náhle ukončeno, přenos byl přerušen
- **451** – akce se souborovým systémem se nezdařila, problém na serveru
- **503** – špatné pořadí příkazů
- **504** – příkaz není naimpementován pro zadaný parametr

- **530** – uživatel nepřihlášen
- **550** – zadaná akce se nezdařila, soubor nebyl nalezen

3. Programátorská příručka

Aplikaci jsem tvořil metodou programuj a opravuj. Nejprve jsem naimplementoval základní příkazy a systém práv. Otestoval jsem funkčnost implementace. Pokud kód nepracoval správně, opravoval jsem ho do té doby, dokud nezačal fungovat. Až potom jsem doimplementovával další funkce do FTP serveru.

Do aplikace jsem naprogramoval jednoduchou správu práv. U každého uživatele se může naspecifikovat cesta a práva k ní. Můžeme tak určit, co v jaké cestě uživatel může dělat, např. mazat.

3.1. Popis tříd

3.1.1. Balíček cz.zcu.ups.arcaoftpd

ArcaoFTPd – hlavní třída aplikace, obsahuje cyklus serveru, kde se čeká na připojení uživatele

3.1.2. Balíček cz.zcu.ups.arcaoftpd.client

- **ClientHandler** – obsluhuje veškerou další komunikaci s klientem po jeho připojení, vykonává zadané příkazy od klienta
- **ClientParameters** – obsahuje veškeré hodnoty od klienta (pracovní adresář, otevřená spojení, atd.)

3.1.3. Balíček cz.zcu.ups.arcaoftpd.connection

- **ActiveConnection** – obsluhuje aktivní datové spojení a spouští datovou akci
- **Connection** – abstraktní třída pro spojeni, od ní dědí ActiveConnection a PassiveConnection
- **PassiveConnection** – obsluhuje pasivní datové spojení a spouští datovou akci

3.1.4. Balíček cz.zcu.ups.arcaoftpd.dataaction

- **DataAction** – abstraktní třída definující datovou akci
- **ListDataAction** – vrací po datovém spojení výpis adresáře
- **NlstDataAction** – vrací po datovém spojení výpis adresářů v zadaném adresáři ve formátu název adresáře a nová řádka
- **RetriveDataAction** – vrací obsah souboru po datovém spojení
- **StoreDataAction** – ukládá soubor přijatý po datovém spojení

3.1.5. Balíček cz.zcu.ups.arcaoftpd.users

- **Map** – seskupuje potřebné informace o namapovaném adresáři a jeho právech
- **User** – udržuje informace o uživateli a jeho právech k souborovému systému
- **Users** – spravuje veškeré uživatele a informace o nich

3.1.6. Balíček cz.zcu.ups.arcaoftpd.util

- **Logger** – třída sloužící k logování spojení a chybových stavů

3.2. Poznatky o síťové komunikaci

Na stránce, podle které jsem se řídil při vytváření FTP serveru jsem se dozvěděl, že je nutné být striktní při odesílání výsledků klientovi, ale zase na druhou stranu být tolerantní na dotazy od klienta. Takže při detekování nových řádek se nehledí, zda obsahuje sekvenci **CRLF** nebo jen **LF**. U příkazů se třeba nehledí na na velikosti písmen příkazu, ale na velikost písmen u parametrů příkayu se již hledí.

To co vrací server klientovi musí být už podle striktních pravidel. U příkazů jsou pevně dány povolené návratové kódy, výsledné odpovědi musí být ukončeny novým řádkem **LF**. Více řádkové odpovědi musí být správně formátovány[1].

4. Uživatelská příručka

4.1. Popis softwaru

Tento program slouží jako ukázka implementace FTP Serveru. Plno věcí není ještě plně dotaženo k dokonalosti, tak aby se mohl FTP Server použít v ostrém provozu. Server podporuje jak stahování tak nahrávání souborů, operace se soubory a adresáři. Obsahuje zjednodušenou správu práv na soubory a adresáře. Program funguje v příkazovém řádku, tedy server lze jen spustit a vypnout.

4.2. Přeložení zdrojových souborů

Program lze přeložit například pomocí **ANTu**. Pro přeložení jsem použil poslední verzi programu **ANT**, který jsem pustil v kořenovém adresáři projektu:

```
> ant
Buildfile: build.xml

javadoc:
   [delete] Deleting directory ArcaoFTPd\doc
    [mkdir] Created dir: ArcaoFTPd\doc
  [javadoc] Generating Javadoc
...

compile:
```

1 Requests, verbs, parameters, responses, and codes - http://cr.yp.to/ftp/request.html#response

```
   [delete] Deleting directory ArcaoFTPd\bin
    [mkdir] Created dir: ArcaoFTPd\bin
    [javac] Compiling 16 source files to ArcaoFTPd\bin

jar:
   [delete] Deleting: ArcaoFTPd\ArcaoFTPd.jar
      [jar] Building jar: ArcaoFTPd\ArcaoFTPd.jar

default:

BUILD SUCCESSFUL
```

Skript obsahuje další použitelné akce:

- **compile** – zkompiluje zdrojový kód do spustitelných class souborů
- **jar** – zabalí class soubory do spustitelného balíku společně se zdrojovými kódy a dokumentací
- **javadoc** – vygeneruje dokumentaci tříd
- **clean** – ostraní veškeré soubory vygenerované soubory víšše uvedených akcí

4.3. Konfigurace a spuštění programu

Konfigurace připojených uživatelů se provadí prostřednictvím souboru users.xml:

```
<?xml version="1.0" encoding="utf-8" ?>
<users>
  <user name="anonymous">
    <password req="0"/>
    <maps>
      <!-- access: r=read, w=write, l=list, a=append file, u=delete file,
d=delete direcory, m=make directory  -->
      <map source="/ftp" destination="/" access="rl"/>
      <map source="/ftp/upload" destination="/upload/" access="rwlmdu"/>
    </maps>
  </user>
  <user name="admin">
    <password>nimda</password>
    <maps>
      <map source="/ftp" destination="/" access="rwlaudm"/>
    </maps>
  </user>
</users>
```

Každý uživatel se definuje tagem **user**. V atributu **name** je pak jeho uživatelské jméno. Tag **user** pak obsahuje další tagy: **password**, **maps**. Tag **password** obsahuje heslo uživatele, pokud ale na heslu nezáleží nastaví se atribut **req** na „0“. Tag **maps** pak obsahuje seznam namapovaných adresářů. Každý adresář je zapsaný prostřednictvím tagu **map**. Ten obsahuje atributy **source** (co budeme mapovat), **destination** (absolutni cesta na ftp, kam budeme mapovat) a **access** (kombinace

práv: „r“ pro čtení, „w“ pro zápis, „l“ pro výpis adresáře, „a“ pro navazování souborů, „u“ pro mazání souborů, „d“ pro mazání adresářů a „m“ pro vytváření adresářů).

Spuštění programu provedeme pomocí:

```
> java -jar ArcaoFTPd.jar [port]
```

Pokud parametr port neuvedeme, FTP server bude naslouchat na výchozím portu 2121.

4.4. Ukončení programu

Program je možné ukončit z konzole pomocí kombinace kláves **CTRL+C**. Tím dojde k úspěšnému ukončení aplikace.

5. Závěr

Práce na této semestrální práci byla pro mě přínosná. Během vývoje programu jsem se seznámil se sockety v Javě a tvorbou serverových aplikací. Také jsem se dozvěděl, více informací ohledně tvorby vláken v Javě, a jak vnitřně funguje protokol FTP. Naučil jsem se tvořit

Původně jsem myslel, že tvorba FTP serveru bude jednoduchou záležitostí, ale zabralo mi to poměrně více času, než jsem doufal, proto jsem nedokázal implementovat úplně celou funkčnost, jakou jsem předpokládal.

Co se týče ostrého nasazení, tak program není pro to určen. Muselo by se ještě plno věcí doprogramovat. Bylo by například nutné upravit vnitřní implementaci souborového systému.

5.1. Nedostatky programu

Bohužel jsem nestačil implementovat časovač pro ukončení klienta po čase nepřítomnosti. To by bylo určitě potřeba doplnit, aby byl FTP server použitelný.

FTP server jsem sice testoval na pár nejrozšířenějších FTP klientech. Sice jsem poustupoval podle doporučení, ale nezaručuji, že bude stoprocentně fungovat úpně na všech clientech.

6. Použité prostředky

6.1. Písemné materiály

Java SDK dokumentace	http://java.sun.com/j2se/1.5.0/docs/api/
FTP: File Transfer Protocol	http://cr.yp.to/ftp.html
RFC 542	specifikace FTP protokolu
RFC 765	novější specifikace FTP protokolu

Martin Sloup, A04372
msloup@students.zcu.cz

6.2. Použitý software

Programovací jazyk Java	http://java.sun.com/
Easy Eclipse Desktop	vývojové prostředí postavené na Eclipsu
Apache ANT	sestavovací prostředek
OpenOffice	svobodný kancelářský balík, použit k napsání této dokumentace
Mozilla Firefox	prohlížeč WWW – test FTP serveru
Total Commander	souborový manažér – test FTP serveru