VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

FTP SERVER PRO WINDOWS MOBILE 6

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE MARTIN BAJCAR
AUTHOR

BRNO 2009

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

FTP SERVER PRO WINDOWS MOBILE 6

FTP SERVER FOR WINDOWS MOBILE 6

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE MARTIN BAJCAR
AUTHOR

VEDOUCÍ PRÁCE ING. PET ČÁSTEK
SUPERVISOR

BRNO 2009

Abstrakt

Tato bakalářská práce se zabývá popisem protokolu FTP a jeho využitím v mobilních zařízeních využívajících operační systém Microsoft Windows Mobile. Dále popisuje možnosti tvorby aplikací pro tento systém v programovacím jazyce Java a shrnuje vlastnosti jeho mobilní verze J2ME. Tyto znalosti jsou poté využity při návrhu a implementaci aplikace FTP server.

Abstract

This bachelor thesis deals with describing protocol FTP and its use in mobile facilities which work with the operating system of Microsoft Windows Mobile. Furthermore, it handles with the possibility of making applications for this system in programming language Java and summarizes features of its mobile version J2ME. This knowledge is afterwards used to project and implement the application FTP server.

Klíčová slova

FTP protokol, server, Microsoft Windows Mobile, J2ME, RFC 959

Keywords

FTP protocol, server, Microsoft Windows Mobile, J2ME, RFC 959

Citace

Bajcar Martin: *FTP server pro Windows Mobile*, bakalářská práce, Brno, FIT VUT v Brně, 2009

FTP server pro Windows Mobile 6

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Petra Částka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Martin Bajcar
20.05 2009

Poděkování

Děkuji svému vedoucímu Ing. Petru Částkovi za cenné rady při vývoji aplikace. Také bych chtěl poděkovat svým rodičům za podporu při studiu a za vytvoření takových podmínek, které mi ulehčovali psaní této bakalářské práce.

© Martin Bajcar, 2009
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů..

Obsah

Obsah 1
1 Úvod 3
1.1 Protokol FTP 4
1.1.1 Řídící spojení 4
1.1.2 Datové spojení 4
1.1.3 Nevýhody FTP protokolu 4
1.2 FTP MODEL 5
1.3 FTP příkazy 6
1.3.1 Příkazy řízení přístupu 7
1.3.2 Příkazy nastavující parametry přenosu 8
1.3.3 Obsluhující příkazy 9
1.4 Návratové kódy příkazů a odpovědi 10
1.4.1 Význam 1. cifry 10
1.4.2 Význam 2. cifry 10
1.4.3 Význam 3. cifry 11
1.5 Aktivní versus pasivní mód serveru 11
1.6 Příklad komunikace serveru a klient 12
2 JAVA 13
2.1 Java pro mobilní zařízení – J2ME 13
2.1.1 Konfigurace: 14
2.1.2 Profily 14
3 Windows Mobile 6 15
3.1 Windows Mobile a Java 16
4 Vlastní implementace FTP serveru 17
4.1 Návrh tříd 17
4.2 Diagram tříd 17
4.3 Třída ServerGUI 18
4.4 Třída ServerSetting 19
4.5 Třída Server a ServerPI 20
4.6 Třída ServerDTP 20
5 Instalace a spuštění programu 21
5.1 Instalace Mysaifu JVM na Windows Mobile 6 21
6 Testování 22
6.1 Výsledky testování 22

7 Závěr 23
Příloha 1 26

1 Úvod

S narůstající mobilitou a výkonem elektronických zařízení narůstají také požadavky uživatelů na programové vybavení těchto zařízení. Zároveň je požadavek, aby bylo možné tyto zařízení připojit do sítě (LAN, internet) a být tak kdykoliv a kdekoliv online.

Tento požadavek se do jisté míry daří výrobcům plnit, avšak dle mého názoru opomíjejí do svých zařízení zakomponovat software, který by umožňoval plně využít možnosti být připojen do sítě. Tento problém se snaží řešit produkty třetích stran, které poskytují dodatečný software jako mobilní internetový prohlížeč, IM komunikátor a další. Stále mi však chybí možnost uživatelsky přívětivějšího sdílení dat uložených v zařízení. V době, kdy se kapacity pamětí mobilních zařízení pohybují v řádech gigabytů a z původně jednoúčelových zařízení se stávají multifunkční, se mi to jeví jako zásadní nedostatek. Možnost sdílení dat tak může hrát významnou roli při koupi mobilního zařízení. Právě praktická část této práce se zabývá vytvořením takového softwaru, který by umožňoval pro uživatele jednoduché a přitom plnohodnotné sdílení dat.

V první kapitole této bakalářské práce jsou popsány vlastnosti a využití protokolu FTP. Na modelu je vysvětlen princip komunikace mezi serverem a klientem. V závěru kapitoly je uveden a rozebrán příklad komunikace od jejího začátku až po konec.

V druhé kapitole je v krátkosti představen programovací jazyk Java a jeho rozdíly oproti jiným programovacím jazykům, zejména je vysvětlen důvod použití virtuálního stroje. Dále je popsána verze Javy J2ME, která je určena pro vývoj aplikací pro mobilní zařízení.

Tématem třetí kapitoly je popis operačního systému Microsoft Windows Mobile a popis a řešení problému spojeného s tvorbou aplikací v jazyce Java pro tento systém.

Čtvrtá kapitola popisuje vlastní implementaci FTP serveru. Čtenáři je vysvětleno, jak aplikace funguje a jak spolu komunikují jednotlivé moduly.

Předposlední kapitola se věnuje instalaci aplikace na mobilní telefon s operačním systémem Windows Mobile, instalaci virtuálního stroje a prvnímu spuštění serveru.

V poslední, šesté kapitole, jsou uvedeny mé vlastní poznatky při tvorbě této aplikace. Dále jsou zde nastíněny možná rozšíření a vylepšení.

1.1 Protokol FTP

FTP (File Transfer Protokol) je protokol, zaměřující se na přenos dat mezi počítači. Je specifikován v [1]. Jedná se o protokol pracující na aplikační vrstvě síťového modelu ISO OSI [2]. FTP tvoří mezi vrstvu mezi operačním systémem a uživatelem, Výhodou toho je, že umožňuje přenášet data mezi různými operačními systémy, které používají různé souborové systémy.

Komunikace dvou entit používající protokol FTP je odvozena od protokolu TELNET [3], tedy formát a podoba posílaných dat je velmi podobná. FTP stejně jako TELNET používá pro přenos zpráv zásadně textovou podobu zpráv ve formátu:

```
PŘÍKAZ<MEZERA>PARAMETRY
```

FTP používá pro svou činnost dvě spojení, která používají jiný port. Řídící spojení, které běží na portu 21 a datové spojení které běží buď na portu 20, nebo na takovém, na kterém se klient a server domluví.

1.1.1 Řídící spojení

Jeho úkolem je nejprve navázat spojení s druhou stranou (server nebo klient) a toto spojení pak řídit. FTP server zde hraje pouze pasivní roli a čeká, až se na něj připojí klient. Role klienta je inicializovat řídící spojení a řídit ho pomocí příkazů. Po celou dobu komunikace je vytvořeno pouze jedno řídící spojení, které je aktivní po celou dobu komunikace. S každým dalším připojeným klientem je vytvořeno další řídící spojení a server jejich příkazy zpracovává paralelně.

1.1.2 Datové spojení

Jakmile klient pošle požadavek na datový přenos, server i klient si vytvoří vlastní přenosový proces (DTP – Data Transfer Proces). Tyto procesy mezi sebou navážou spojení a pomocí něj je jsou přenášena data. Datové spojení, na rozdíl od řídícího spojení, existuje jen po dobu nutnou pro vlastní přenos dat. Po je dokončení jsou přenosové procesy na obou stranách ukončeny a jsou znova vytvořeny až při dalším požadavku o datový přenos.

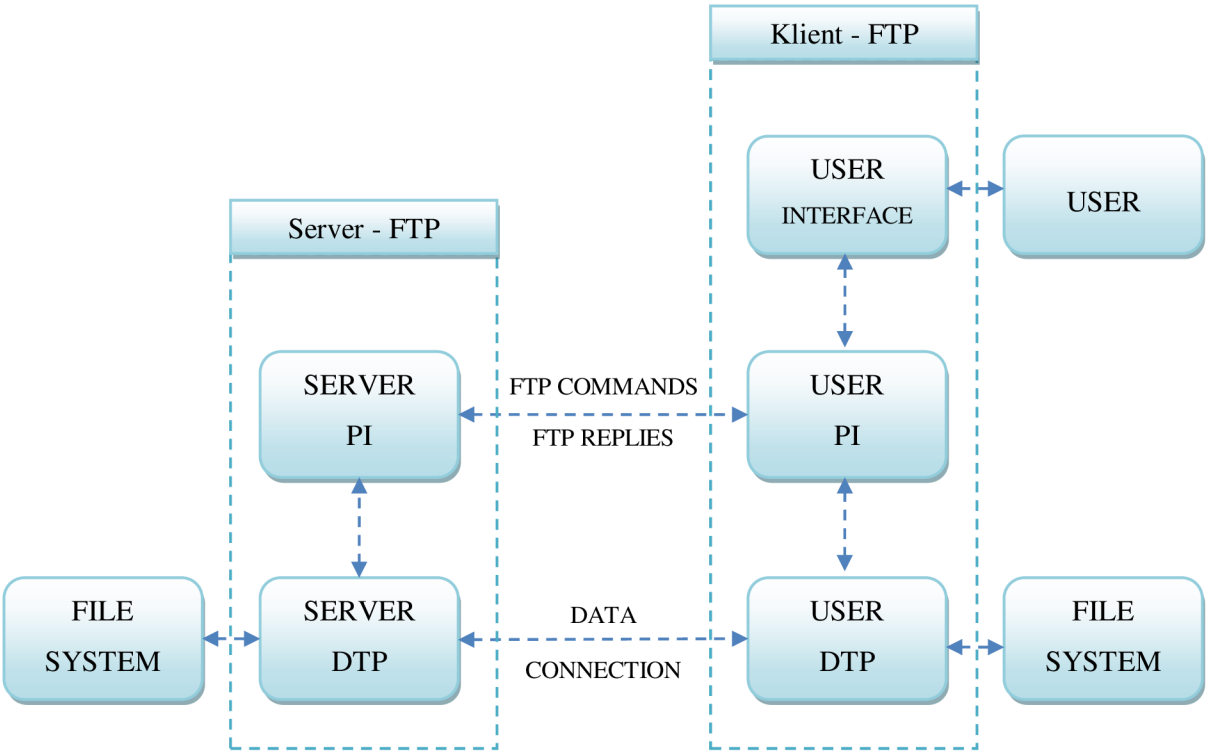
Řídící spojení inicializuje vždy klient. Datové spojení může inicializovat server i klient, podle toho, požaduje-li klient aktivní nebo pasivní režim přenosu. Rozdíl mezi aktivním a pasivním režim je popsán v kapitole 1.5.

1.1.3 Nevýhody FTP protokolu

Hlavní nevýhodou tohoto protokolu je jeho bezpečnost. Data i příkazy jsou posílány spolehlivou cestou přes TCP protokol, ale v otevřené textové podobě. Toho může útočník snadno využít a zjistit tak důvěrná data. Tuto nevýhodu lze odstranit nebo zásadně omezit SFTP (Secure FTP), které používá pro šifrování dat protokol SSH. Další nevýhodou je poměrně velká časová prodleva mezi stahováním jednotlivých souborů. Pokud se jedná o velmi malé soubory, může být režie komunikace (nastavení parametrů přenosu) delší než samotný přenos souboru.

1.2 FTP MODEL

Model použití FTP protokolu při komunikaci klient – server je znázorněn na následujícím obrázku:



Obr. 1: Model použití při komunikaci klient – server

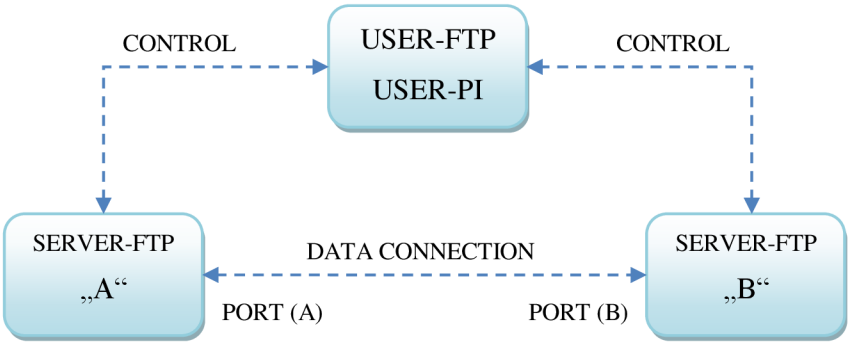
Vysvětlení pojmů:

- **File System** – souborový systém, např. FAT32, NTFS, EXT2.
- **PI** – internet protokolu FTP
- **DTP** – funkce serveru zajišťující přenos dat, může být aktivní nebo pasivní
- **Server PI** – interpret protokolu na straně serveru, ustanovuje a řídí spojení mezi sebou a klientem. Zároveň přijímá a zpracovává příkazy od klienta a posílá odpovědi. Je také tím, kdo řídí datové spojení.
- **Server DTP** – část serveru, která se stará o datové přenosy. V aktivním modu inicializuje spojení mezi serverem a „naslouchajícím“ klientem, v pasivním modu ustanovuje toto spojení.
- **FTP Commands** – množina příkazů, kterým rozumí server i klient. Jsou specifikovány v RFC 959 [2].
- **FTP Replies –** množina odpovědí na FTP příkazy.
- **Data Connection –** spojení, přes které probíhá přenos dat. Může být ustanoveno mezi ServerDTP a User DTP nebo mezi dvěma Server DTP.
- **User Interface** – rozhraní, přes které uživatel využívá služeb FTP. Může jít být internetový prohlížeč nebo specializovaný program
- **User PI** - interpret protokolu na straně klienta, inicializuje spojení mezi sebou a serverem, posílá příkazy a zpracovává odpovědi

- **User DTP -** část klienta, která se stará o datové přenosy. V aktivním modu ustanovuje spojení mezi serverem a klientem, v pasivním modu inicializuje toto spojení.
- **User** – uživatel

User PI inicializuje řídící spojení, přes které jsou zasílány příkazy. Server PI tyto příkazy přijme a zpracuje a podle typu příkazu a výsledku jeho provedení se odešle zpět odpověď. Odpověď se skládá z návratového kódu a popisu odpovědi. Možné návratové kódy jsou specifikovány v [1]. FTP příkazy definují parametry pro uskutečnění datového přenosu. Je to způsob přenosu, port, typ přenášených dat a struktura dat. Dále jsou posílány příkazy, které určují, co klient po serveru požaduje. To může být stažení, uložení nebo mazaní dat a další.
Pokud tedy klient požaduje přenos dat, pošle serveru příkaz a začne „naslouchat“ na předem dohodnutém portu. Server tento příkaz rozpozná a inicializuje spojení na daný port. Tento port však nesmí být stejný jako ten, přes který jsou posílány řídící příkazy, protože i v průběhu přenosu dat musí být klient schopen zasílat serveru příkazy (např. přerušení právě probíhajícího přenosu).

Další možností je komunikace dvou serverů mezi sebou. Model komunikace znázorňuje následující obrázek:



Obr. 2: Model použití při komunikace server – server

Vysvětlení pojmů:

- **Server-FTP** – proces nebo skupina procesů zajištující činnost serveru. Řadí se sem Server DTP a Server PI.
- **User-FTP** – proces nebo skupina procesu zajišťující činnost klienta. Řadí se sem User DTP, User PI a User Interface

Tato situace může nastat, požaduje-li klient přenos dat mezi dvěma servery. Klient vytvoří řídící spojení mezi serverem A i mezi serverem B a nastaví parametry přenosu tak, aby se datové spojení vytvořilo mezi těmito servery.

1.3 FTP příkazy

Komunikace FTP serveru a FTP klienta je typu dotaz – odpověď. Klient odešle příkaz serveru a čeká, než server příkaz zpracuje a odešle odpověď. Podle přijaté odpovědi se klient rozhodne co dál dělat,

resp. který příkaz může následovat. Posloupnost příkazů je přesně daná FTP protokolem. Například při autentizaci klienta musí po příkazu USER bezprostředně následovat příkaz PASS.

Pro každý příkaz jsou definované stavy, ve kterých se může nacházet v době provádění a po ukončení provádění. Přechod z jednoho stavu do druhého (s výjimkou stavu Begin) se děje na základě výsledku provedení příkazu. Jednotlivé stavové diagramy a jejich popis je uveden v příloze A.

FTP příkazy můžeme rozdělit do tří skupin:

1. Access control commands - příkazy řízení přístupu
2. Transfer parametr commands - příkazy nastavující parametry přenosu
3. FTP service command - obsluhující příkazy

Pozn.: Seznam příkazů není úplný. Existují další příkazy, které upravují nebo nějak rozšiřují činnost serveru. Zde je uveden seznam běžně používaných FTP příkazů a těch, které jsem implementoval ve své práci.

1.3.1 Příkazy řízení přístupu

USER - user name

Syntax: `USER` *`username`*
Odesláním toho příkazu začíná proces přihlášení. *username* může být uživatelské jméno, které má povolen přístup na server nebo „anonymous“ v případě, že je na serveru povoleno anonymní přihlášení.

PASS – password

Syntax: `PASS` *`password`*
Po příkazu USER je vyžadováno zadání uživatelova hesla. Při zadání správného hesla je proces přihlášený úspěšně dokončen. V opačném případě je uživatel vyzván k zadání nového hesla. Pokud je na serveru povoleno anonymní přihlášení, je po uživateli požadováno zadání emailové adresy.

CWD – change working directory

Syntax: `CWD` *`remote-directory`*
Pomocí toho příkazu lze změnit právě prohlížený adresář za jiný, který je dán parametrem *remote-direcory*.

CDUP – change to parent directory

Syntax: `CDUP`
CDUP je podobný příkazu CWD. Pomocí tohoto příkazu lze ve stromové hierarchii adresářů postoupit o úroveň výše a zobrazit tak rodičovský adresář aktuálně prohlíženého adresáře.

REIN – reinitialize

Syntax: `REIN`
Tento příkaz zruší všechna nastavení provedená aktuálním uživatelem a nastaví je do defaultního nastavení. Po tomto příkazu by měl následovat příkaz USER pro přihlášení jiného uživatele.

QUIT – logout

Syntax: `QUIT`

Příkaz ukončí řídící spojení mezi serverem a klientem.

1.3.2 Příkazy nastavující parametry přenosu

PORT – data port

Syntax: `PORT` *`a1,a2,a3,a4,p1,p2`*

a1 .. a4 – čísla udávající adresu ve formátu IPv4

p1, p2 – čísla udávající číslo portu. Číslo získáme podle vzorce

`port = p1 * 256 + p2`

Parametr tohoto příkazu udává, na jakou adresu a jaký port se má server připojit, dojde-li k požadavku na datový přenos. V případě, že příkaz není zadán, jsou použity defaultní hodnoty.

PASV – Passive

Syntax: `PASV`

Tímto příkazem se nastaví server do pasivního modu. Server pak čeká, až se na něj připojí klient a můžou se tak přenášet data. Součástí odpovědi na tento příkaz je i adresa serveru a číslo portu, na kterém „naslouchá“.

TYPE – representation type

Syntax: `TYPE` *`type-character`*

Příkazem se nastaví typ souboru, který se bude přenášet. *type-character* může být:

- A – ASCII text
- E – EBCDIC text
- I – image (binární data)
- L – místní formát, který specifikuje, kolik bitů má jeden byte na daném systému
- Defaultně je nastaven parametr A.

STRU - file structure

Syntax: `STRU` *`structure-character`*

Příkazem se nastaví struktura přenášených dat. *structure-character* může být:

- F – file (soubor)
- R – record (záznam)
- P – page (stránka)

Defaultně je nastaven parametr F.

MODE – transfer mode

Syntax: `MODE` *`mode-character`*

Tímto příkazem se nastaví způsob přenášení dat. *Mode-chracter* může být:

- S – stream (proud dat)
- B – block (blok dat)
- C – compresed (komprimovaná data)

Defaultně je nastaven parametr S.

1.3.3 Obsluhující příkazy

RETR - retrieve

Syntax: `RETR` *`filename`*
Pošle-li klient tento příkaz, znamená to, že požaduje stažení souboru se jménem *filename*. Pokud je server v aktivním modu, server inicializuje spojení s klientem. V opačném případě čeká, než se připojí klient. Datový přenos se uskuteční pouze v případě, že požadovaný soubor na serveru existuje.

STOR – store

Syntax: `STOR` *`filename`*
Pošle-li klient tento příkaz, znamená to, že chce na server umístit soubor se jménem *filename*. Pokud je server v aktivním modu, server inicializuje spojení s klientem. V opačném případě čeká, než se připojí klient. Datový přenos nastane v případě, že na straně serveru neexistuje soubor se stejným jménem.

RNFR – rename from

Syntax: `RNFR` *`filename`*
Tento příkaz specifikuje soubor nebo adresář, který má být přejmenován. Musí být následován příkazem RNTO.

RNTO – rename to

Syntax: `RNTO` *`filename`*
Tímto příkazem se provede přejmenování souboru nebo adresáře, který byl dán příkazem RNFR na jméno *filename*.

ABOR – abort

Syntax: `ABOR`
Tento příkaz provede okamžité ukončení provádění předcházejícího příkazu. Pokud již byl předcházející příkaz dokončen, je tento příkaz bez efektu. Tímto příkazem se ukončuje datové spojení, řídící spojení zůstává aktivní.

DELE – delete

Syntax: `DELE` *`filename`*
Příkaz provede smazání souboru *filename*.

RMD – remove directory

Syntax: `RMD` *`filename`*
Příkaz provede smazání adresáře *filename*.

MKD – make directory

Syntax: `MKD` *`filename`*
Příkaz provede vytvoření nového adresáře *filename*.

PWD – print working directory

Syntax: `PWD`
Příkaz vrátí klientovi jméno aktuálního adresáře.

LIST – list

Syntax: `LIST filename`
Pokud je *filename* adresář, příkaz vrátí seznam adresářů a souborů a informace o nich v daném adresáři. Pokud je *filename* soubor, příkaz vrátí informace o daném souboru.

NLST – name list

Syntax: `NLST directory`
Podobný příkazu LIST. Příkaz NLST vrátí pouze seznam adresářů a souborů v *directory*.

SYST – system

Syntax: `SYST`
Příkaz slouží k identifikaci systému, na kterém běží server. Tato informace se může použít například k rozpoznání formátu odpovědi vrácené příkazem LIST.

NOOP – noop

Syntax: `NOOP`
Příkaz sám o sobě neprovádí nic. Může se použít na udržení spojení mezi serverem a klientem.

1.4 Návratové kódy příkazů a odpovědi

Klient posílá na server příkazy. Aby se dozvěděl o tom, jak příkaz dopadl, server odesílá zpět odpověď. Ta se skládá z tří ciferného čísla a textové části. Podle čísla klient identifikuje výsledek a příkazu a může správně reagovat. Textová část slouží k popisu chyby a její podoba není přesně daná. První cifra určuje, zda příkaz dopadl dobře, špatně nebo jeho provádění nebylo dokončeno.

1.4.1 Význam 1. cifry

První cifra návratového kódu indikuje úspěch nebo neúspěch příkazu, nebo zda přijatý příkaz je kompletní nebo nekompletní.

- **1yz** *Positive Preliminary reply* **-** požadovaná akce se zahajuje. Další odpověď bude zaslána před ukončením akce. Používá se k indikování, že příkaz byl přijat a klient má dávat pozor na datové spojení.
- **2yz** *Positive Completion reply* - požadovaná akce byla úspěšně ukončena. A další může následovat.
- **3yz** *Positive Intermediate reply* - příkaz byl přijat, ale požadovaná akce byla odložena kvůli očekávání dalších informací. Uživatel by měl poslat další příkaz specifikující tuto informaci. Používá se ve skupinových příkazech.
- **4yz** *Transient Negative Completion reply* - příkaz nebyl přijat a požadovaná akce nebyla provedena, ale chybový stav je jen dočasný a akce může být zopakována.
- **5yz** *Permanent Negative Completion reply* - příkaz nebyl přijat a požadovaná akce nebyla provedena. Chybový stav je trvalý a uživatel by neměl akci opakovat.

1.4.2 Význam 2. cifry

Druhá cifra rozděluje návratové kódy podle jejich funkcí.

- **x0z** *Syntax* - syntaktická chyba

- **x1z** *Information* - odpověď na požadovanou informaci
- **x2z** *Connections* - odpovědi týkající se řídícího a datového spojení
- **x3z** *Authentication and accounting* - odpovědi týkající se přihlašovacího procesu
- **x4z** *Unspecified as yet* - zatím nespecifikováno
- **x5z** *File system* - odpovědi udávající status file systému

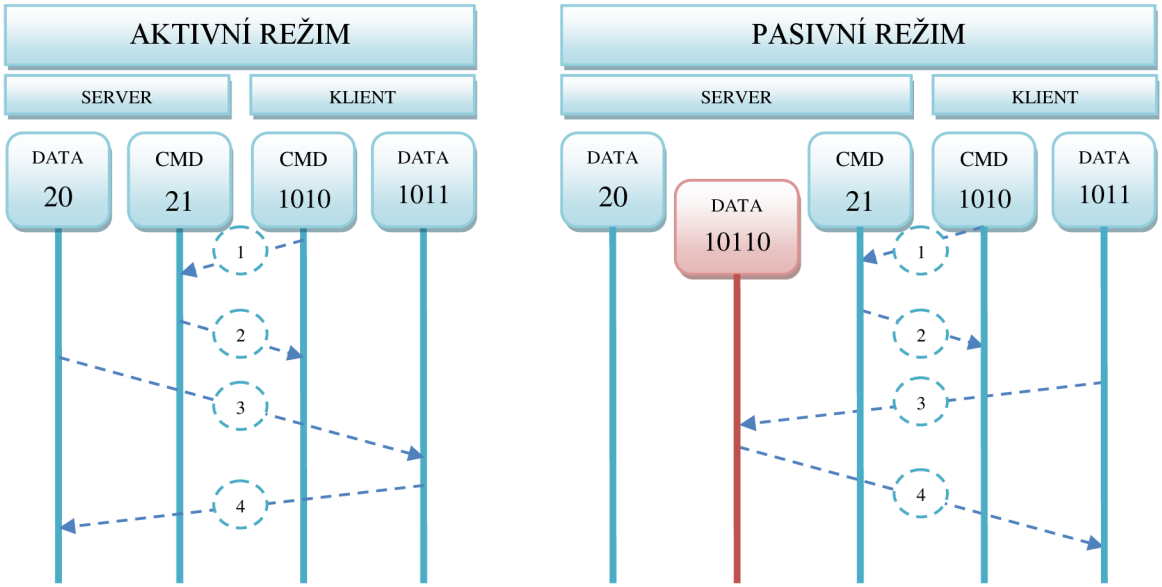
1.4.3 Význam 3. cifry

Třetí cifra udává další upřesnění odpovědi podle kategorie příkazu. U každé kategorie může být význam odlišný.

Úplný popis návratových kódů je uveden v [1].

1.5 Aktivní versus pasivní mód serveru

Někdy se může stát, že se klientovi podaří připojit na server, ale pak přestane server odpovídat. Může to být způsobeno tím, že klient je na síti umístěn za firewallem a server se na něj nemůže připojit a vytvořit tak datové spojení. Řešení spočívá právě v tom, nastavit serveru do pasivního modu. Rozdíl mezi aktivním a pasivním modem je znázorněn na následujícím obrázku.



Obr. 3: Aktivní versus pasivní režim

Jak je vidět, rozdíl mezi aktivním a pasivním režimem je ten, že v aktivním režimu se server připojuje na klienta. V pasivním režimu se server pošle informace o IP adrese a portu, na kterém „naslouchá“ a čeká, až se klient připojí. Takto se ustanoví datové spojení a mohou se přenášet data.

1.6 Příklad komunikace serveru a klient

Uvedený příklad je zobrazuje úplný záznam komunikace serveru a klienta od připojení klienta až po jeho odpojení. Ze záznamu je vidět, že serveru „běží“ na adrese 192.168.1.25 a je na něm povoleno anonymní přihlášení. V celé komunikace není vidět příkaz `PASV`, čili se jedná a komunikaci v aktivním režimu. Klient s adresou 192.168.1.15 požaduje po serveru stažení souboru `kapela.mp3`, který je umístěn v adresáři `/pisnicky`. Jedná se o binární soubor, proto klient nastaví přenos do binárního módu. Po dokončení přenosu je komunikace ukončena příkazem `QUIT`

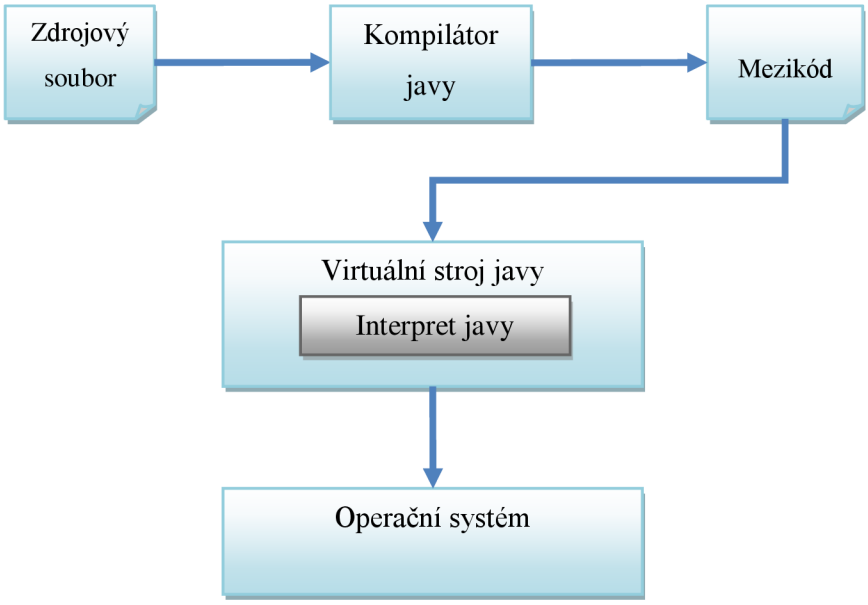
```
Connect to: (06.05.2009 19:55:36)
hostname=192.168.1.25
username=anonymous
startdir=
220 Microsoft FTP Service
USER anonymous
331 Anonymous access allowed, send identity (e-mail name) as password.
PASS ***********
230 Anonymous user logged in.
SYST
215 Windows_NT
Connect ok!
PWD
257 "/" is current directory.
Složka
TYPE A
200 Type set to A.
PORT 192,168,1,15,10,73
200 PORT command successful.
LIST
150 Opening ASCII mode data connection for /bin/ls.
Stahuje se
Čekáme na server...
226 Transfer complete.
CWD pisnicky
250 CWD command successful.
PWD
257 "/pisnicky" is current directory.
Složka
PORT 192,168,1,15,10,75
200 PORT command successful.
LIST
150 Opening ASCII mode data connection for /bin/ls.
Stahuje se
Čekáme na server...
226 Transfer complete.
TYPE I
200 Type set to I.
PORT 192,168,1,15,10,76
200 PORT command successful.
RETR kapela.mp3
150 Opening BINARY mode data connection
Stahuje se
Čekáme na server...
226 Transfer complete.
QUIT
221
```

2 JAVA

Java je moderní programovací jazyk, který se používá obvykle pro programy, které mají pracovat na více platformách.

Základy toho jazyka lze nalézt v projektu Oak, který vznikl ve firmě SUN na počátku 90. let pro řízení elektronických výrobků. V roce 1994 byl přenesen jako programovací jazyk do prostředí počítačů pod názvem JAVA[4].

Výhoda jazyka Java je jeho multiplatformost. Zdrojový kód jazyka je přeložen do spustitelného kódu (bytecode), který lze pak spouštět pomocí nainstalovaného běhového prostředí (JRE – Java Runtime Edition) na virtuálních strojích (JVM - Java Virtual Machine) přímo na různých typech zařízení. Výhodou tohoto způsobu spouštění je právě zmiňovaná multiplatformost. Nevýhodou je však vyšší spotřeba systémových zdrojů (využití procesoru a paměti) a pomalejší běh výsledné aplikace. Tuto nevýhodu je možné odstranit použitím specializovaných překladačů na cílovém prostředí (JIT – Java just In Time). Postup zkompilování a spuštění aplikace je znázorněn na následujícím obrázku.



Obr. 4: Postup vytvoření java aplikace

2.1 Java pro mobilní zařízení – J2ME

J2ME je verze produktu Java určená pro trh s mobilními zařízeními, jakými jsou mobilní telefony, přehrávače, PDA a další. J2ME poskytuje kompletní podporu moderních síťových aplikací pro malá zařízení. Specifikace J2ME definuje tyto komponenty [5]:

- sérii javovských virtuálních strojů

- skupiny knihoven a API, které zle spustit na každém VM – nazývají se konfigurace a komponenty
- nástroje pro vývoj a nastavení

2.1.1 Konfigurace:

Konfigurace definují členění produktů podle množství paměti a výkonu procesoru. V současnosti existují tyto dvě konfigurace:

2.1.1.1 CDC (Connected Device Configuration)

- určena pro výkonná zařízení, která obsahují plnou verzi virtuálního stroje podobného tomu, který se používá ve verzi J2SE (Java Standard Edition)
- zařízení musí být řízena 32bitovým procesorem a musí mít 2MB a více pro prostředí Java

2.1.1.2 CLDC (Connected Limited Device Configuration)

- udává mnohem menší nároky na zařízení než CDC
- zařízení může mít celkem 160 – 512 KB paměti pro prostředí Java včetně paměti RAM, paměti FLASH a ROM

2.1.2 Profily

Profil je sada programových rozhraní (API) tvořící nadstavbu konfigurace. Nabízí tak programu přístup ke specifickým vlastnostem konkrétního zařízení. Profilů je celá řada, nejznámější a nejpoužívanější jsou:

2.1.2.1 MIDP

- je navržen pro prácí s CLDC a poskytuje sadu programových rozhraní pro práci s mobilními zařízeními, jako jsou mobilní telefony a pagery
- obsahuje pro uživatelské rozhraní, povoluje ukládání a práci v síti
- aplikace, které běží pod MDP, se nazývají midlety

2.1.2.2 Základní profil (Foundation profile)

- rozšiřuje rozhraní CDC ale nedodává žádné API pro uživatelské rozhraní
- slouží jako základ pro další profily

2.1.2.3 Osobní profil (Personal profile)

- rozšiřuje základní profil a grafické uživatelské rozhraní
- používá balíček java.awt

2.1.2.4 RMI profil

- přidává k základnímu profilu vzdálené volání metod kompatibilní s rozhraním standardní edice J2SE (knihovna `java.rmi`)

2.1.2.5 Herní profil (Game profile)

- není ještě zcela specifikován
- určen primárně k vývoji her

3 Windows Mobile 6

Windows Mobile je operační systém firmy Microsoft určený pro mobilní zařízení. Záměrem Microsoftu bylo, aby se tento systém co nejvíce podobal desktopové verzi systému Windows. Svědčí o tom přítomnost nabídky Start, která je pro desktopové verze typická. Dále obsahuje několik aplikací jako Office Mobile, Windows Messenger, Outlook Mobile, Internet Explorer, Media player, které jsou v podstatě upravené desktopové aplikace.

Windows Mobile je určen pro širokou škálu zařízení. Muže být nasazen v mobilních telefonech, přehrávačích muziky či videa nebo i zařízeních určených pro automobily. Proto existuje několik verzí tohoto systému, které se liší programovým vybavením, různou podporou paměťových karet a rozlišením.

Platforma Windows Mobile zahrnuje podporu USB portů, bezdrátových technologií GSM, IrDA, bluetooth, WIFI, několik typů paměťových karet, možnost ovládat mobilní telefon přes dotykový display nebo hlasem.

Aktuální verze systému je 6.1 a v brzké době se očekává nová verze 6.5, která má obsahovat mnoho vylepšení jako nový plastický vzhled, podpora technologie Silverligt a další[6].

Obr. 5:Základní obrazovka WM 6 a Internet Explorer Mobile

3.1 Windows Mobile a Java

Hned na začátku je potřeba říct, že operační systém Windows Mobile nemá zabudovanou podporu Javy. Je až záležitostí výrobce zařízení, zda a jakou verzi Javy zakomponuje do svého zařízení. Proto není neobvyklé, že aplikace na jednom zařízení bez problému fungují a na druhém je nelze spustit. V poslední době se však tato situace vylepšuje a mnoho výrobců si uvědomuje, že bez podpory Javy ztrácejí konkurenceschopnost na trhu s mobilními zařízeními.

Přesto však existuje několik projektů, které poskytují JVM (Java Virtual Machine) právě pro mobilní zařízení. Jejich snahou je poskytnou uživateli stejné prostředí pro běh programů jako je na osobních počítačích. Jsou to např. Esmertec Jbed CDC Java Virtual Machine , IBM WebSphere Everyplace Micro Environment, Mysaifu JVM a existují i další [7]. První dva uvedené jsou bohužel placené. Mysaifu JVM je šířen pod licencí GPLv2, proto jsem se rozhodl jej využít ve své práci.

4 Vlastní implementace FTP serveru

4.1 Návrh tříd

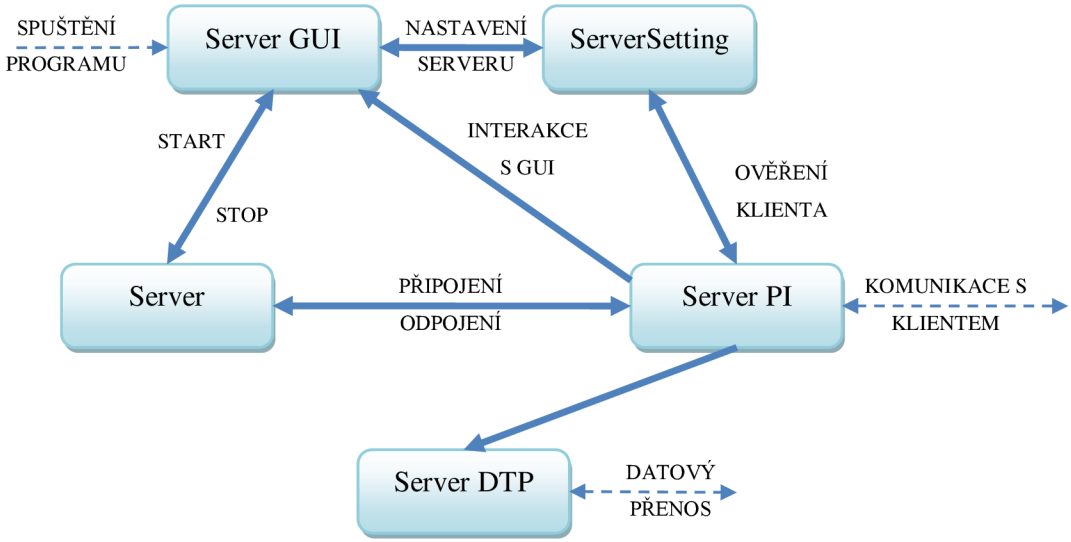
Výsledkem návrhu má být aplikace typu server, která splňuje specifikace protokolu FTP.

Uživatel musí mít možnost server kdykoliv spustit, zastavit a měnit jeho nastavení. Server musí umožňovat obsluhu více klientů najednou, tudíž každý klient musí být obsloužen ve vlastním vlákně (procesu) a jedno vlákno bude sloužit pro příjem spojení.

Výsledkem výše jmenovaných vlastností, které server musí splňovat je rozdělení jednotlivých úloh do 5 tříd:

1. ServerGUI
2. ServerSetting
3. Server
4. ServerPI
5. ServerDTP

4.2 Diagram tříd

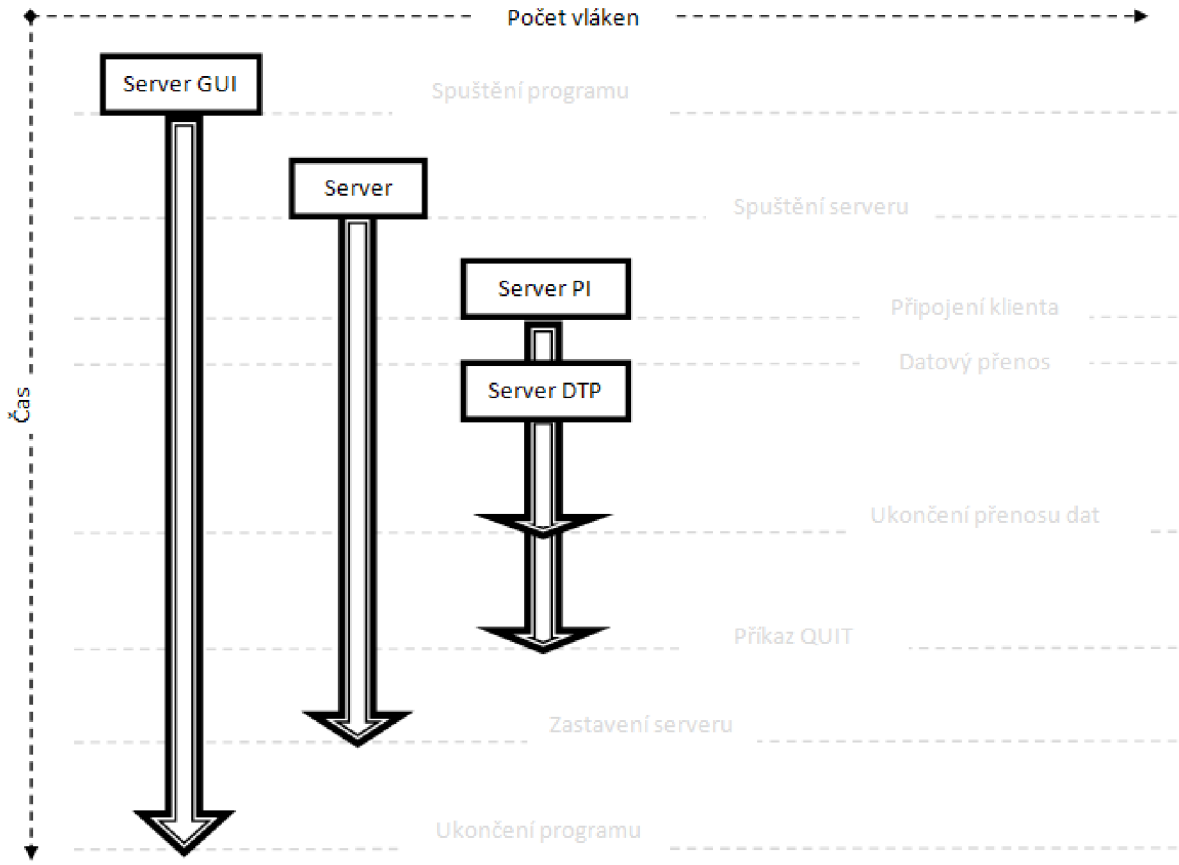


Obr. 6: Diagram tříd

Spuštěním programu se vytvoří instance třídy ServerGUI a zobrazí se okno programu. Nyní program čeká na pokyny od uživatele. Ten má možnost buď spustit server, nebo změnit nastavení serveru. Pokud se uživatel rozhodne spustit server, vytvoří se nové vlákno s instancí třídy Server. Druhá

možnost je, že uživatel bude chtít před spuštěním například změnit heslo určitému klientovi. Pokud se tak rozhodne, vytvoří se nová instance třídy ServerSetting a zobrazí se nové okno s možnostmi nastavení.

Úkolem třídy Server je přijímat příchozí spojení od klientů, do té doby, než uživatel server zastaví. Jakmile se připojí klient, vytvoří se nové vlákno s instancí třídy ServerPI. Ta zajišťuje vlastní komunikace s klientem. Pokud se klientovi nepodaří autorizovat, je vlákno automaticky ukončeno. V opačném případě se aktualizuje seznam a počet připojených klientů (šipka ze ServerPI do ServerGUI). Pokud klient pošle příkaz požadující datový přenos (LIST,STOR,RETR) vytvoří se objekt třídy ServerDTP. Již nevytváří další vlákno. Provede se přenos dat a objekt je okamžitě zrušen. Pro úplnou funkčnost server je minimální možný počet současně běžících vláken 3. Maximální možný počet současně běžících vláken je dán schopností JVM a maximální možný počet současně připojených klientů je JVM_Thread_Max – 2 (max. počet vláken – vlákno třídy ServerGUI a vlákno třídy Server). Pošle-li klient příkaz QUIT, znamená to, že se chce odpojit. ServerPI provede aktualizaci seznamu a počtu připojených klientů a ukončí se. Časový digram života vláken je zobrazen na Obr. 7.



Obr. 7: Život vláken

4.3 Třída ServerGUI

Třída ServerGUI implementuje grafické uživatelské rozhraní. Při spuštění programu se vytvoří instance této třídy a uživateli se zobrazí úvodní okno aplikace. Toto okno obsahuje IP adresu, kterou má zařízení přiděleno z DHCP serveru nebo ji uživatel ručně nastavil. Pod IP adresou je zobrazen

stav serveru. Uživatel tak při prvním pohledu ví, zda je server spuštěn nebo zastaven. Zaškrtávací políčko umožňuje rychlé přepínání nastavení serveru, které povoluje nebo zakazuje anonymní přístup na server. Dále na této obrazovce uživatel vidí jména přihlášených klientů a okamžitě tak vidí kdo je připojen na server. Ve spodní části jsou umístěna tlačítka pro spuštění a zastavení serveru. Následující obrázek zobrazuje popisované okno programu.

Obr. 8: Třída ServerGUI - úvodní okno programu

4.4 Třída ServerSetting

Jak název třídy napovídá, tato třída implementuje možnosti nastavení serveru. Toto nastavení spočívá v přidávání nebo odebrání klientů, kterým je na server povolen přístup. Nastavení se načítá ze souboru `FTPAccount.txt`, který je vytvořen při prvním spuštění v kořenovém adresáři. Formát dat uložených v souboru je následující

```
jméno_klienta;heslo
```

kde `jméno_klienta` je přístupové jméno na server a heslo je `heslo` přístupové heslo klienta. Okno s nastavením zobrazuje následující obrázek.

Obr. 9:Třída ServerSetting - okno nastavení

4.5 Třída Server a ServerPI

Úkolem třídy Server je přijímat řídící spojení od klientů a předávat jej třídě ServerPI. Tato třída implementuje vlastnosti protokolu FTP, tedy provádění příkazů. Každý příkaz, který klient odešle je objektem této třídy přijat a následně zpracován. Podle výsledku zpracování je zpět odeslána zpráva s návratovým kódem odpovídající výsledku příkazu.

Každý nový proces vytvořený touto třídou má nastavený časový interval 5 minut, po který může být v nečinnosti (klient neposílá žádné příkazy). Pokud klient neposílá žádné příkazy, je zbytečné udržovat s ním spojení, proto je toto spojení ukončeno.

Pošle-li klient nějaký příkaz, který není rozpoznán, je mu vrácena odpověď, značící, že server tento příkaz nepodporuje.

4.6 Třída ServerDTP

Tato třída implementuje procesy spojené s přenosem dat. Jelikož se přistupuje k datům, je potřeba přístupy synchronizovat, aby nedocházelo ke konfliktům, které by znamenaly buď poškození dat, nebo špatné načtení dat. Synchronizace se v jazyce Java provádí pomocí konstrukce

```
synchronized(objekt){...}
```

kde `object` může být název metody. Takto jsou synchronizovány metody, provádějící příkazy LIST,STORE a APPE.

Tato třída má na starosti i implantaci příkazu LIST. Tento příkaz vrací klientovi seznam souborů a adresářů v aktuálním adresáři. Každý řádek seznamu musí být v určitém formátu, aby jej klient rozpoznal. Nejrozšířenější formát, který podporuje každý klient je formát `bin/ls`[8]:

```
-rw-r--r-- 1 owner group           213 Aug 26 16:31 README
```

Řádek obsahuje:

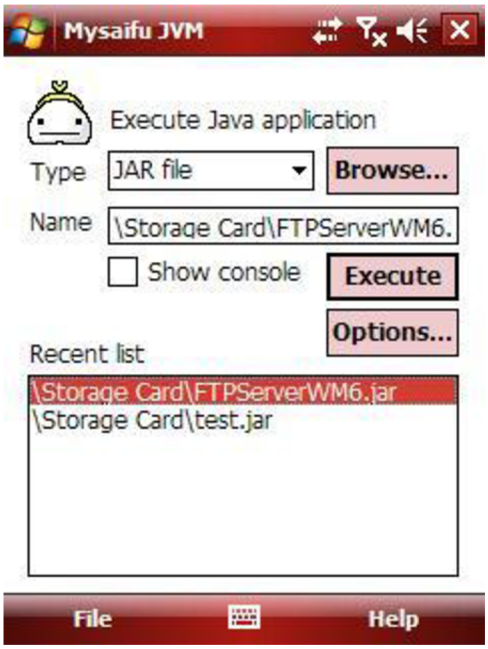
1. `-` pro soubor nebo `d pro adresář`
2. řetězec `rw-r--r-- 1 owner group` pro soubor `nebo rwxr-xr-x 1 owner group` pro adresář
3. velikost souboru
4. třípísmenné označení měsíce
5. dne
6. hodina
7. minuta
8. název souboru s příponou

5 Instalace a spuštění programu

Proto, aby program fungoval a mohl být spuštěn, je nutné, aby na cílovém zařízení byl instalován virtuální stroj Javy. Mysaifu JVM , dostupný z [9], je volně dostupný ke stažení, proto jsem se rozhodl využít právě jeho. Po instalaci virtuální stroje stačí na zařízení nakopírovat soubor FTPServer.jar který lze spustit ve virtuálním stroji.

5.1 Instalace Mysaifu JVM na Windows Mobile 6

Mysaifu JVM je distribuován v podobě `cab` balíčku. Tento balíček stačí nakopírovat na zařízení s Windows Mobile a spustit. Po dokončení instalace se v Menu – Programy objeví ikona Mysaifu JVM.



Obr. 10: Mysaifu JVM - spustění aplikace

Na obrázku 10 je zobrazeno menu programu. V tomto menu stačí již vybrat požadovaný program a kliknout na *Execute*. Pokud je vše v pořádku, spustí se program. V opačném případě ke spuštění programu nedojde a v konzoli, kterou je možné zobrazit z menu *File*, bude zobrazena chyba. Další nastavení programu není nutné.

6 Testování

Testování aplikace jsem prováděl pomocí emulátoru telefonního přístroje s operačním systémem Windows Mobile. Tento emulátor je volně dostupný na oficiálních stránkách firmy Microsoft [10]. Emulátor je program, který emuluju (simuluje) chování nějakého zařízení a chová se stejně jako dané zařízení.

Činnost aplikace jsem testoval a ověřoval pomocí FTP klietů:

1. Integrovaný ftp klient v programu TotalCommander 7.5
2. WinSCP 4.2.1
3. Internet Explorer 7

6.1 Výsledky testování

Testováním aplikace se mi podařilo opravit několik chyb. Důležité bylo provést vždy stejné úkony ve všech testovacích aplikacích a porovnat, zda jsou výsledky shodné a hlavně správné.

Při testování jsem narazil na to, že klienti posílají i některé příkazy, které nejsou specifikovány v RFC 959. Tyto příkazy můj program nereflektoval a vykonávání programu skončilo chybou. Tuto chybu se mi podařilo opravit a nyní program odpoví návratovým kódem 501, který informuje klienta o tom, že daný příkaz není na serveru podporován.

Po odstranění nalezených chyb program fungoval správně se všemi třemi klienty. Proto předpokládám, že funkčnost programu nebude nijak omezena ani v dalších FTP klientech, kteří splňují specifikaci RFC 959.

7 Závěr

Cílem této bakalářské práce byla tvorba aplikace pro platformu Windows Mobile v jazyce Java, konkrétně FTP serveru. Při tvorbě této aplikace jsem uplatnil znalosti získané v průběhu studia, zejména znalosti získané v předmětech zabývajících se síťovou tematikou. Při řešení této práce jsem narazil na problémy spojené s tvorbou Java aplikací pro platformu Windows Mobile, které se mi však podařilo odstranit a výsledkem je aplikace FTP Server splňující specifikaci FTP protokolu. Díky nutnosti použití virtuálního stroje je aplikace spustitelná pouze na těch zařízeních a platformách, pro které je dostupný virtuální stroj.

Proto námětem dalšího rozšíření může být odstranění tohoto nedostatku a tím zvýšení počtu možných zařízení, využívající jiné operační systémy (Symbian, Android), na kterých by aplikace fungovala. Dalším možným rozšířením aplikace je zvýšení bezpečnosti. Jelikož se jedná o aplikace pro mobilní zařízení a komunikace probíhá především pomocí bezdrátových technologií, není pro případného útočníka složité odposlechnout komunikaci. Implementací protokolu SSH by se výrazně zamezilo případnému zneužití odposlechnutých dat.

Literatura

[1] POSTEL J., REYNOLDS, J. *RFC959 - File Transfer Protocol*. 1985 [cit. 2009-05-15]. Dostupný z WWW: <http://www.faqs.org/rfcs/rfc959.html>.

[2] KABELOVÁ, Alena, DOSTÁLEK, Libor. *Velký průvodce protokoly TCP/IP a systémemDNS*. 5. vyd. [s.l.] : Computer press, 2008. 488 s. ISBN 978-80-251-2236-.

[3] POSTEL J., REYNOLDS, J. *RFC854 Telnet Protocol Specification*. 1983 [cit. 2009-05-15]. Dostupný z WWW: <http://www.faqs.org/rfcs/rfc854.html>.

[4] HATINA, Petr . *Programování v jazyku Java*, 9.7.2004 [cit. 2009-05-15]. Dostupný z WWW: <http://www.linuxsoft.cz/article.php?id_article=244>.

[5] TOPLEY, Kim. *J2ME v kostce: Pohotová referenční příručka*. 2004. vyd. [s.l.] : Grada, 2004. 536 s. ISBN 80-247-0426-9.

[6] MIKUDÍK, Radek, VOKÁČ, Luděk. *Windows Mobile 6.5 ve znamení změn* [online]. 17. února 2009 [cit. 2009-05-15]. Dostupný z WWW: <http://palmare.idnes.cz/windows-mobile-6-5-ve-znameni-zmen-d4j-/tech-a-trendy.asp?c=A090216_184937_tech-a-trendy_ram>.

[7] Wikipedie. *List of Java virtual machina.* 11.května 2009 [cit. 2009 – 05- 17]. Dostupný z WWW: <http://en.wikipedia.org/wiki/List_of_Java_virtual_machines>

[8] BERNSTEIN D. J., *Internet publication. FTP: File Transfer Protocol. /bin/ls format.* Dostupný z WWW: < http://cr.yp.to/ftp/list/binls.html>

[9] Mysaifu JVM. Dostupný z WWW: <http://www2s.biglobe.ne.jp/~dat/java/project/jvm/index_en.html>

[10] Windows Mobile Emulator. Dostupný z WWW: < http://www.microsoft.com/downloads/details.aspx?FamilyId=1A7A6B52-F89E-4354-84CE-5D19C204498A&displaylang=en>

Seznam příloh

Příloha 1. Stavové diagramy jednotlivých FTP příkazů
Příloha 2. DVD

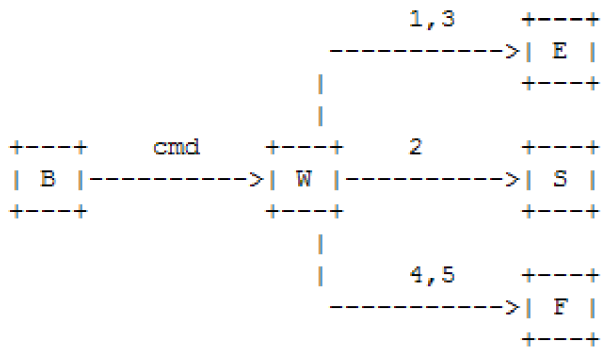
Příloha 1

Stavové diagramy zpracování příkazů

Jednotlivé příkazy jsou rozděleny do několika skupin, pro které platí stejné stavové digramy. Změna stavu je závislá pouze na první číslici odpovědi serveru. Další čísla v tomto případě nemají význam. V diagramech jsou znázorněny tyto stavy:

- B - Begin - počáteční stav
- W - Wait For Replay - čekání na odpověď
- S - Success -úspěšné provedení
- F – Failure - selhání
- E – Error - chyba

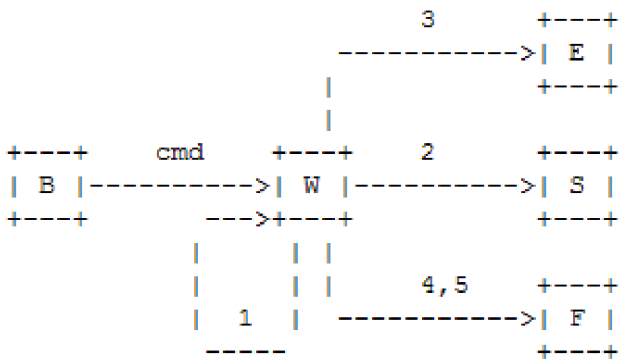
První diagram reprezentuje největší skupinu příkazů.



Obr. 1: Diagram 1

Platí pro tyto příkazy:
ABOR, ALLO, DELE, CWD, CDUP, SMNT, HELP, MODE, NOOP, PASV, QUIT, SITE, PORT, SYST, STAT, RMD, MKD, PWD, STRU a TYPE.

Další diagram je velmi podobný prvnímu.



Obr. 2: Diagram 2

Tento model je možné použít i pro první skupinu příkazů, jediný rozdíl je, že po přijetí odpovědi ze skupiny 1yz je očekávána další odpověď. V prvním modelu odpověď 1yz znamená chybu.
Platí pro tyto příkazy:
APPE, LIST, NLST, REIN, RETR, STOR a STOU.

Další model representuje příkazy pro přejmenování souboru – RNFR a RNTO.

```
+---+   RNFR    +---+    1,2    +---+
| B |---------->| W |---------->| E |
+---+           +---+        -->+---+
                 | |         |
          3      | | 4,5     |
   --------------  ------    |
  |                       |  |   +---+
  |                ------------->| S |
  |                |   1,3 |  |   +---+
  |              2|  --------
  |                | |     |
  V                | |     |
+---+   RNTO    +---+ 4,5 ----->+---+
|   |---------->| W |---------->| F |
+---+           +---+           +---+
```

Obr. 3: Diagram 3

Následující model representuje diagram pro použití příkazu REST a RETR,APPE a STOR.

```
+---+   REST    +---+    1,2    +---+
| B |---------->| W |---------->| E |
+---+           +---+        -->+---+
                 | |         |
          3      | | 4,5     |
   --------------  ------    |
  |                       |  |   +---+
  |                ------------->| S |
  |                |   3   |  |   +---+
  |              2|  --------
  |                | |     |
  V                | |     |
+---+   cmd     +---+ 4,5 ----->+---+
|   |---------->| W |---------->| F |
+---+        -->+---+           +---+
            |      |
            |  1   |
             ------
```

Obr. 4: Diagram 4

„cmd“ může být jeden z příkazů RETR, STOR a APPE.

Poslední diagram je ze všech nejsložitější a representuje proces přihlašování.

```
                        1
+---+   USER    +---+-------------->+---+
| B |---------->| W | 2       ---->| E |
+---+           +---+------  |  -->+---+
                 | |       | | |
               3 | | 4,5   | | |
   --------------   -----  | | |
  |                      | | | |
  |                      | | | |
  |                 ---------  |
  |               1|     | |   |
  V                |     | |   |
+---+   PASS    +---+ 2  |  ------>+---+
|   |---------->| W |------------->| S |
+---+           +---+   ---------->+---+
                 | |   | |     |
               3 | |4,5| |     |
   --------------   --------   |
  |                    | |  |  |
  |                    | |  |  |
  |                 -----------
  |             1,3|   | |  |
  V                |  2| |  |
+---+   ACCT    +---+--  |   ----->+---+
|   |---------->| W | 4,5 -------->| F |
+---+           +---+------------->+---+
```

Obr. 5: Diagram 5