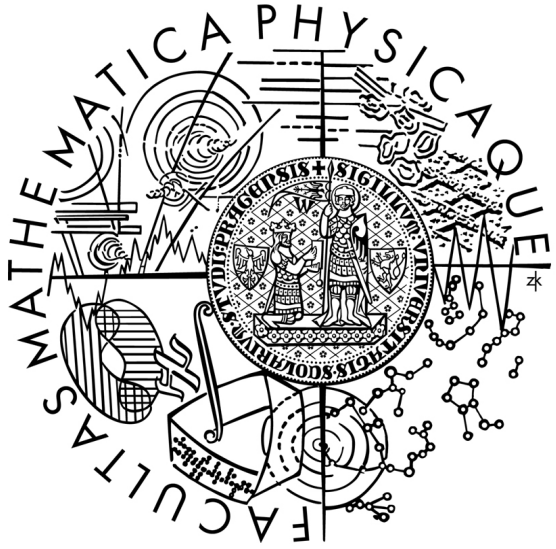
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Michal Tekeľ

FTP server

Katedra softwarového inženýrství

Vedoucí bakalářské práce: RNDr. Jana Štanclová

Studijní program: Informatika, Správa počítačových systémů

2006

Ďakujem RNDr. Jane Štanclovej za vedenie pri bakalárskej práci a za cenné rady a pripomienky.

Prehlasujem, že som svoju bakalársku prácu napísal samostatne a výhradne s použitím citovaných prameňov. Súhlasím s požičiavaním práce a jej zverejňovaním.

V Prahe dňa 11. augusta 2006 Michal Tekeľ

Obsah

Název práce: FTP server
Autor: Michal Tekeľ
Katedra: Katedra softwarového inženýrství
Vedoucí bakalářské práce: RNDr. Jana Štanclová
e-mail vedoucího: Jana.Stanclova@mff.cuni.cz

Abstrakt: Práca sa zaoberá implementáciou efektívneho FTP servra a programov potrebných na jeho správu, ktoré sú jej súčasťou. Dôležitým kritériom je výkon a schopnosť plne využívať dostupné prostriedky v prípade potreby. Server má podporovať všetky bežné FTP príkazy, menej používané umožňuje doplniť pomocou pluginov. Dielom práce je aj komunikačný protokol medzi servrom a administrátorskými programami, ktorý umožňuje správu aj pomocou cudzích programov.

Klíčová slova: FTP server, komunikačný protokol

Title: FTP server
Author: Michal Tekeľ
Department: Katedra softwarového inženýrství (Department of software engineering)
Supervisor: RNDr. Jana Štanclová
Supervisor's e-mail address: Jana.Stanclova@mff.cuni.cz

Abstract: This work deals with implementation of an effective FTP server and programs needed for its administration, which are also part of the work. Important factor is performance of the server alone and ability to fully utilize system resources when needed. Server should support all common FTP commands, ocassionaly used commands can be implemented using plugins. Communication protocol between server and administrative programs is also part of this work. This protocol permits use of third party programs for server management too.

Keywords: FTP server, communication protocol

1 Úvod

V dnešnej dobe je považovaná možnosť výmeny informácií medzi počítačmi za samozrejmosť. Počítače sú bežne prepájané do väčších sieti rôzneho druhu, rýchlosti a spoľahlivosti. Účel všetkých týchto sieti je ale rovnaký a je ním práve možnosť komunikácie, zdieľania a výmeny dát. K tomuto účelu sú ale potrebné isté prostriedky, aby mohli nájsť počítače a siete úplne odlišných architektúr „spoločnú reč“, je potrebná istá štandardizácia. Táto je dosiahnutá pomocou komunikácie prostredníctvom dostatočne presne definovaných a pre svoje špecifické potreby prispôsobených protokolov. V istom prenesenom zmysle slova sa dá povedať, že protokoly predstavujú reč, ktorou prebieha komunikácia. Komunikačný protokol nemusí byť iba dostatočne jednoznačný a akceptovaný, ale požaduje sa aj istá flexibilita ohľadom možných vylepšení a reakcií na zmeny používania do budúcna. Dôležitá je tiež výkonnosť samotného prenosu údajov cez daný protokol, a to jednak z pohľadu výpočtovej záťaže komunikujúcich strán, ale aj hľadiska pomeru užitočného množstva prenesených dát v pomere k celkovému objemu prenesených údajov. Zo známejších protokolov: HTTP sa používa pri prenose web stránok, ICQ pre instant messaging, SSH pre bezpečnú vzdialenú prácu. Nie všetky protokoly sa však stanú dostatočne široko akceptované, aby sa mohli považovať za štandard. FTP je ale protokolom, ktorý je vďaka svojím vlastnostiam už dlhšiu dobu jedným z najpoužívanejších pre prenos súborov.

1.1 FTP všeobecne

File **T**ransfer **P**rotocol (alebo protokol pre prenos súborov) vznikal spoločne s rozširovaním sietí, ako reakcia na potrebu štandardizácie výmeny dát medzi vtedy dosť odlišnými platformami [1]. Preto je jedným z najstarších protokolov určených pre prenos súborov. Napriek tomu sa postupne vyvíjal a prispôsoboval novým podmienkam (podpora národných znakov jednotlivých jazykov) a spôsobom využitia. O jeho kvalite svedčí aj to, že sa bežne používa dodnes a stále predstavuje pomerne dobrý prostriedok pre prenos a výmenu údajov uložených vo forme súborov. Samotný protokol je orientovaný viac na strojové spracovanie a preto o ňom možno tvrdiť, že nie je veľmi užívateľsky prívetivý. Prostredníctvom FTP komunikujú vždy dve strany – klient a server, prípadne dva servre, ktoré ovláda jeden klient. Klient je ten, ktorý požaduje od servra služby – napríklad výpis dostupných súborov, ich prenos ku klientovi, ale aj opačným smerom na server. Klient zadáva príkazy prostredníctvom protokolu telnet [2]. Tento protokol je používaný pre komunikáciu so servrom a cez telnet sa prenášajú FTP príkazy klienta a odpovede servra na ne. Zatiaľ čo klient pracuje naraz nanajvýš s dvoma servrami (kedy koordinuje prenos súborov medzi nimi), jeden server dokáže obslúžiť toľko klientov, koľko mu dovoľuje výkon a ostatné prostriedky počítača, kde beží. Preto je dôležité, aby bol práve server dostatočne efektívnym programom.

1.2 Ciele práce

Cieľom tejto práce je implementácia efektívneho FTP servra. Dôležitý bude predovšetkým vysoký výkon – a to v zmysle nízkej záťaže procesora (a teda pri rovnakom maximálnom dostupnom výkone zvládnutie viacerých klientov) ale aj rýchlosti samotného prenosu súborov prostredníctvom protokolu. Server nebude orientovaný na to, aby mal veľké množstvo rôznych funkcií (ktoré by mohli spomaľovať), ale bude obsahovať možnosť preniesť tieto funkcie na ďalšie tzv. správcovské programy. Tieto budú môcť nad nami definovaným rozhraním (tzv. ADM protokolom) implementovať svoje ďalšie funkcie podľa vlastnej ľubovôle. Toto rozhranie bude možné používať aj z obyčajných komunikačných programov, neprispôsobených špeciálne pre tento účel, ako napr. rôznych telnet klientov [P4]. Súčasťou práce budú aj tri správcovské aplikácie, ktoré budú umožňovať meniť nastavenia servra. Každá z nich bude ináč orientovaná a slúžia aj ako ukážka flexibility použitia ADM protokolu. Jedna grafická a viac užívateľsky prívetivá, druhá jednoduchšia v štýle telnet klientov s istými špecifickými pridanými funkciami a tretia najjednoduchšia bude slúžiť pre prácu z príkazového riadku. Táto bakalárska práca bude obsahovať aj návody na prácu so všetkými štyrmi aplikáciami. Samotný server, nazvaný FTPk, bude podporovať pluginy, teda možnosť užívateľa doprogramovať si k servru vlastné funkcie.

1.3 Štruktúra práce

V úvode práce sú objasnené princípy fungovania samotného FTP protokolu. V ďalšej kapitole sa budeme zaoberať porovnaním vybraných FTP servrov a ich vlastností. Implementovanému servru je venovaná štvrtá kapitola, kde sú rozoberané jeho funkcie a režimy činnosti, spôsoby ovládania a nastavovania. Práca obsahuje tiež popis vytvoreného ADM protokolu (kapitola 5), pomocou ktorého sa ovláda server a cez ktorý prebieha komunikácia so správcovskými programami. Týmto programom je venovaná kapitola 6, kde je popis ich ovládania a možností nastavovania servra, ktoré tieto programy umožňujú. Šiesta kapitola je tiež koncipovaná tak, aby mohla slúžiť aj ako užívateľská príručka pre prácu s týmito programami. V kapitole 7 sa rozoberá otvorenosť servra – modifikovateľnosť, možnosť využitia cudzích správcovských aplikácií a možnosti budúceho vývoja. Záver zhŕňa výsledky práce a porovnáva výsledný server s cieľmi stanovenými v úvode.

Prílohy práce obsahujú popis obsahu CD, kde sa nachádza programátorská a užívateľská dokumentácia k jednotlivým programom vytvoreným v rámci tejto práce. Na médiu sa nachádza tiež elektronická verzia bakalárskej práce a vývojovej dokumentácie, ktorá sa nachádza aj v prílohe.

2 FTP protokol

Táto kapitola je venovaná popisu samotného protokolu FTP, rozoberá príčiny jeho vzniku a možnosti využitia. Ďalej popisuje samotné princípy fungovania protokolu a pre lepšie pochopenie obsahuje aj názorne ukážky práce s protokolom. Jednotlivé podkapitoly sú zamerané na hlavné časti protokolu, kde popisujú jeho fungovanie podrobnejšie a do dostatočnej hĺbky potrebnej k pochopeniu fungovania protokolu.

2.1 Úvod do protokolu

Skratka FTP označuje tzv. File Transfer Protocol, teda protokol na prenos súborov. FTP sa ako internetový štandard postupne rozvíjal a zlepšoval cez radu RFC (request for comments) špecifikácí. Aktuálny štandard je popísaný v RFC 959 [1]. Novšie verzie špecifikácie obsahujú iba rôzne nepovinné vylepšenia ohľadom bezpečnosti. Venujú sa tiež možnosti použitia rôznych jazykov (pôvodne bolo podporované iba ASCII v telnete, čo je pre niektoré jazyky a ich národné znaky nedostatočné). Autormi a ľuďmi, ktorí sa najviac zaslúžili o vznik tohto (ale aj pár predchodcov tohto) RFC, sú J. Postel a J. Reynolds.

FTP je pomerne starý protokol. RFC 959 je z roku 1985, preto v ňom nájdeme rôzne, dnes už nepotrebné veci, ktoré napríklad zjednodušovali komunikáciu medzi mainframe počítačmi (ktoré mali rôzne veľkosti byteov), umožnovali preformátovať dokument do formátu pre tlač a pod. Účel FTP možno vystihnúť najlepšie, ak použijeme preklad z úvodu RFC 959 [1]. Úlohou FTP je

1. podporovať zdieľanie súborov
2. podnietiť nepriame alebo implicitné použitie vzdialených počítačov
3. ochrániť užívateľa od rozdielov v súborových systémoch medzi rôznymi počítačmi
4. prenášať údaje bezpečne a efektívne

Bezpečnosťou prenosu bolo myslené použitie štruktúry záznamu (STRU R) a blokového režimu prenosu (MODE B). Vysvetlenie týchto pojmov možno nájsť v nasledujúcich častiach tejto kapitoly. Bezpečným prenosom sa teda v 4. bode nemyslelo využitie enkrypcie. Nami implementovaný FTPk teda enkrypciu podporovať nebude, ale bude možné ju do FTPk pridať prostredníctvom pluginov.

2.2 Model FTP

Schéma 1 prevzatá priamo z RFC znázorňuje princíp činnosti protokolu. FTP pracuje na princípe server-klient. Server na ľavej strane schémy prijíma FTP príkazy a posiela odpovede na ne cez **control connection** (spojenie na posielanie FTP príkazov). V prípade prenosu súboru sa vytvorí nové spojenie – **data connection**, cez ktoré sa prenesú údaje vo forme vopred dohodnutej cez control connection. FTP server teda pozostáva z dvoch častí:

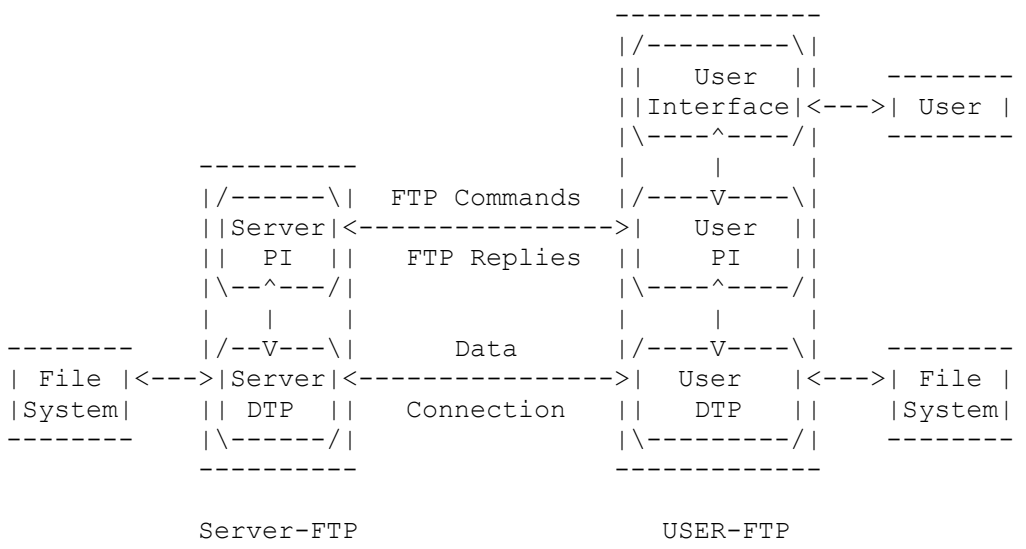


Schéma 1 [1]

1. **P**rotocol **I**nterpreter (skratka PI) prijíma FTP príkazy, posiela odpovede na ne, nastavuje parametre prenosu a iniciuje samotný prenos
2. **D**ata **T**ranfer **P**rocess (skrátene DTP) sa stará len o prenos údajov. Táto časť robí prípadne konverzie z vnútornej na dohodnutú reprezentáciu.

Takáto konverzia nastáva napr. pri stiahnutí plain text súboru z unixového stroja. DTP konvertuje konce riadkov z unixových LF (Line Feed – odriadkovanie) na štandard ASCII, teda CRLF (Carriage Return, Line Feed – návrat kurzora na začiatok, odriadkovanie), resp. inú cez PI dohodnutú reprezentáciu. Obdobne ako server, funguje aj klient. Ten má však okrem DTP a PI navyše rozhranie pre užívateľa, pretože FTP protokol nie je veľmi user friendly a je navrhnutý skôr na strojové spracovanie.

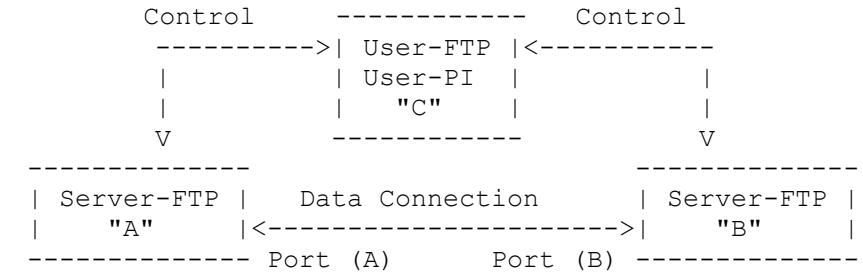


Schéma 2 [1]

Ako možno vidieť na schéme 2, FTP umožňuje aj prenos medzi servrami. V tomto prípade sa klient napojí na oba servre, cez control connection im nastaví parametre a cieľ prenosu, potom jednoducho spustí prenos – jednému servru dá uložiť súbor, druhému ho dá odoslať, ako cieľ pritom ale uvedie prvý server, ktorý už čaká na pripojenie a prenos.

2.3 Parametre prenosu údajov

Ako už bolo spomínané, údaje sa prenášajú cez data connection vo formáte nastavenom parametrami. Práve tu sa prejaví vlastnosť FTP protokolu, ktorá umožňuje bez problémov prenášať súbory medzi ináč úplne vnútorne nekompatibilnými počítačmi. Možno nastaviť tri parametre prenosu (v zátvorke je FTP príkaz):

1. **Typ (TYPE)** – sú podporované 4 typy údajov
   - ASCII, ktorý je aj základným nastavením a musí byť akceptovaný všetkými implementáciami protokolu. Dáta sa interpretujú ako 8 bitové byty, CRLF znamená koniec riadku.
   - EBCDIC typ predstavuje iné kódovanie znakov ako ASCII, je to pomerne starý štandard a dnes sa už nepoužíva (využívali ho predovšetkým staré veľké servre). Táto reprezentácia znakov má význam, najmä ak prenos nastane medzi dvoma počítačmi, z ktorých každý používa vnútorne tento typ, týmto im odpadá konverzia na iný typ a spätná do svojho formátu.
   - IMAGE typ znamená, že údaje sa majú reprezentovať ako prúd (stream) bitov. Používa sa pri prenose napr. obrázkov, zvuku, resp. binárnych súborov.
   - LOCAL typ umožňuje nastaviť vlastnú veľkosť byte prenášaných údajov. Napr. 36 bitový počítač chce poslať 36 bitové čísla na 32 bitový počítač, teda špecifikuje parameter príkazom TYPE L 36. 32 bitový počítač teraz vie, že ide o 36 bitové čísla a tak ich prekonvertuje napr. na 64 bitové, ktoré si uloží. Ale ak na tento 32 bitový počítač príde požiadavka odoslať údaje ako 36 bitové, tak tento ich z 64 bitovej reprezentácie prekonvertuje na 36 bitovú. Údaje sa prenášajú vždy ako 8 bitové byty, teda dve 36 bitové čísla ako 9B. Práve preto sa zavádza typ LOCAL, aby komunikujúce strany mohli špecifikovať, čo je vlastne obsahom prenášaných údajov.

Typy ASCII a EBCDIC majú aj druhý nepovinný parameter, ktorý umožní špecifikovať typ použitého formátovania. Dnes sa už veľmi nepoužíva (jedná sa o formát plaintext súborov). Podporované formáty sú Non Print – žiadne vertikálne formátovanie, Telnet – formátovanie špecifikované rovnako ako v protokole telnet a ASA – tzv. fortranovské formátovanie. Nami implementovaný FTPk bude podporovať typy ASCII a IMAGE. EBCDIC a LOCAL bude možné doplniť pomocou pluginov.

2. **Štruktúra (STRU)** – FTP obsahuje tri definované štruktúry súborov:
   - File – súbor sa má interpretovať ako kontinuálna sekvencia údajov typu špecifikovaného pomocou TYPE. Toto je defaultné nastavenie.
   - Record – súbor pozostáva z viacerých záznamov. Súvisí to najmä s tým, že niektoré počítače (rôzne mainframe) si ukladajú údaje nie ako súbory ale ako záznamy.
   - Page – toto má význam pri následnej tlači súborov. Dnes sa to už vôbec nepoužíva (lebo plaintext súbory väčšinou na stránky delené nie sú). Táto štruktúra špecifikuje napr. typ stránky (prvá, posledná a pod.), číslo stránky a iné.

Nami implementovaný FTPk bude podporovať štruktúru File. Record a Page bude možné doplniť pomocou pluginov.

3. **Režim prenosu (MODE)** – možno nastaviť tri režimy:
   - Stream – dáta sa prenášajú ako prúd bytov, uzavretie data connection znamená koniec súboru. Nie je to veľmi spoľahlivé, ale je to jednoduché a dnes sa v podstate ani nepoužíva nič iné.
   - Block – údaje sú rozdelené do blokov, kde každý blok má hlavičku, ktorá obsahuje popis typu bloku a jeho veľkosť. Tento typ je spoľahlivejší ako Stream, pretože koniec súboru je indikovaný v hlavičke bloku. Veľmi sa však nepoužíva, pretože stream plne postačuje.
   - Compressed – ako napovedá názov, údaje sú komprimované, bohužiaľ iba kompresiou typu RLE [3] (v podstate sa komprimuje iba ak sa opakuje niektorý byte za sebou, tak sa tam dá namiesto celej sekvencie iba tento byte a počet opakovaní). Vďaka slabej kompresii sa nepoužíva. Je vhodné radšej skomprimovať súbory ručne, napr. do formátu zip.

FTPk bude podporovať režim Stream. Block a Compressed bude možné doplniť prostredníctvom pluginov.

2.4 Vytváranie a manažment spojení, zotavovanie sa z chýb

Okrem nastavovania parametrov prenosu údajov, možno nastavovať a meniť aj parametre samotného spojenia. Štandardným nastavením je port 21 pre control connection a port o 1 nižší ako control connection (t.j. 20) pre data connection. Port pre data connection je však možno meniť. FTP môže pracovať v dvoch režimoch – aktívny a pasívny. V aktívnom režime sa server pripojí na klienta a potom začne prenos. V pasívnom režime je to naopak – server čaká, kým sa napojí klient. Medzi týmito dvoma režimami sa možno medzi prenosmi prepínať, slúžia na to príkazy PORT a PASV. Príkaz PORT prepne server do aktívneho režimu a povie mu IP adresu a port, na ktorý má napojiť data connection. Príkaz PORT patrí do minimálnej špecifikácie FTP, teda musí byť akceptovaný všetkými implementáciami. PASV povie servru, že má čakať na spojenie. Server ako odpoveď pošle IP adresu a port, na ktorom počúva. IP adresa sa posiela preto, aby bolo možné posielať údaje medzi dvoma servrami. Prenos prebieha tak, že klient cez control connection prepne jeden server do pasívneho režimu. Od tohto servra dostane IP adresu a port ako odpoveď. Túto dvojicu pošle ako parameter druhému servru, ktorý sa následne aktívne pripojí na prvý server (viz kapitolu 2.8, kde je názorná ukážka ovládania dvoch servrov). Všetky FTP implementácie musia akceptovať štandardné porty. Jedine cez control connection je možné nastavovať iné. Niekedy je to výhodné, napríklad v prípade parametru prenosu stream. Pri tomto režime prenosu musí byť koniec súboru indikovaný ako uzavretie spojenia a nie je možné hneď na tom istom porte otvoriť nové spojenie. TCP protokol totiž ešte chvíľu čaká za stratenými paketmi a pod. Toto spomalenie sa prejaví najmä pri prenose veľkého množstva malých súborov.

Čo sa týka zisťovania a zotavovania z chýb pri komunikácii, FTP sčasti spolieha na TCP a jeho služby pre spoľahlivé doručenie údajov a neobsahuje žiadne mechanizmy na detekciu a opravu chýb vzniknutých pri prenose. Poskytuje ale možnosť obnovenia zlyhaného prenosu (ktorý sa zistí napr. tak, že spadla celá sieť, alebo veľkosť súboru je iná ako vypisovaná cez LIST). Na toto obnovenie slúži príkaz REST, tento príkaz má parameter, ktorý určuje odkiaľ ma začať prenos, či už ide o číslo bloku alebo pozíciu v súbore. Za týmto príkazom by mal nasledovať príkaz na odoslanie, alebo prijatie

súboru (STOR alebo RETR). Tieto príkazy následne nezačnú prenos na začiatku súboru, ale v mieste špecifikovanom parametrom príkazu REST.

2.5 FTP príkazy

Príkazy a odpovede na ne sa prenášajú cez control connection vo formáte telnet, čo je skoro plain text, až na niektoré možnosti nastavenia ohľadom kódovania, mazania znakov atď., ktoré FTP však nepoužíva. Telnet sa používa preto, aby bolo možné jednoducho sa pripojiť na server a ručne mu zadávať príkazy. Syntax príkazov a aj odpovede však vôbec nie sú user friendly, ale skôr vo formáte, ktorý sa dobre strojovo spracováva: FTP príkazy majú 4 znaky, zopár iba 3, presne určenú syntax a počet parametrov. Nemá zmysel vymenovávať presnú syntax parametrov všetkých príkazov. Popísané budú len tie dôležitejšie, ktoré ešte neboli popisované skôr. Ostatné príkazy, ich syntax a všetko potrebné možno nájsť v RFC 959 [1], ktoré obsahuje aj prehľadné tabuľky syntaxe a podobné pomôcky pre ľahšiu implementáciu protokolu. Príkazy možno rozdeliť do skupín podľa toho čo vykonávajú. FTP obsahuje príkazy pre (toto sú v podstate skupiny):

- identifikáciu a prihlásenie užívateľa – USER, PASS, ACCT, QUIT, REIN, SMNT
- prácu s adresármi a súbormi (zmena adresára, mazanie, zmena aktuálneho adresára a pod.) – CWD, CDUP, RNFR, RNTO, DELE, MKD, RMD, PWD, LIST, NLST
- nastavovanie parametrov prenosu – PORT, PASV, TYPE, MODE, STRU
- príkazy pre prípravu, samotný prenos a príjem súborov – RETR, STOR, STOU, APPE, ALLO, REST, ABOR

A potom je tu ešte skupinka ostatných príkazov:

- SYST – odpoveďou by mal byť systém, na ktorom beží server. Toto je použiteľne pri príkaze LIST, pretože rôzne operačné systémy vypisujú zoznamy adresárov a súborov rôzne. FTP protokol neobsahuje žiadnu špecifikáciu tohto výpisu, preto ho možno použiť prakticky bez obmedzenia. Klient a server ak sú kompatibilné, tak poznajú svoje výpisy – dnes sa v praxi používa unixovský výpis presne zhodný s výstupom príkazu ls. Avšak niektoré staré mainframe počítače zo 70. a 80. rokov majú svoje úplne iné súborové systémy. FTP možno použiť teda aj na nich vďaka voľnosti formátu výpisu súborov.
- STAT vráti aktuálny stav (stav prenosu a nastavenie prenosových parametrov)
- HELP väčšinou vypíše zoznam podporovaných príkazov a nejaké informácie navyše (ako napríklad meno a verzia servra a pod.)
- SITE slúži pre špecifické služby daného servra. Poskytované služby (podobne ako pri LIST, ani tu FTP neobmedzuje) sa odporúčajú zistiť pomocou príkazu HELP SITE .
- NOOP nevykoná žiadnu operáciu. Slúži predovšetkým pre overenie spojenia so servrom.

Rozšírenia tohto RFC obsahujú ešte niektoré príkazy navyše, ktoré súvisia s nastavením znakových sád a pod., alebo nejako rozširujú niektoré FTP príkazy, sú to napr. XPWD, XCDW a iné. Pri príkazoch nezávisí na veľkých a malých písmenách, teda retr je to isté čo napr. rETr.

2.6 Odpovede na FTP príkazy

Odpovede sú reakciou servra na FTP príkazy zaslané užívateľom. Informujú o úspechu, alebo neúspechu vykonávania príkazu, alebo sú priamo výsledkom činnosti príkazu. Napr. odpoveďou na príkaz LIST je výpis súborov.

Odpovede začínajú trojciferným číselným kódom, medzerou a potom nasleduje nejaký jednoduchý popis odpovede. Pre funkčnosť FTP, ale tento popis nemusí byť prítomný, dôležitý je číselný kód. Text odpovede je určený pre užívateľa. Každá z troch cifier je číslo od 0 do 5. Pre jednoduchosť ale aj presnosť obsahuje RFC 959 pre každý príkaz zoznam možných odpovedí. Prvá cifra odpovede znamená:

- 1yz kladná predbežná odpoveď, server spracúva požiadavku, možno očakávať ďalšiu odpoveď o výsledku spracovania
- 2yz kladná konečná odpoveď, požadovaná akcia prebehla v poriadku
- 3yz kladná okamžitá odpoveď, príkaz bol prijatý, ale dokončenie akcie vyžaduje ďalšie príkazy (napr. RNFR a potom RNTO pre premenovanie súboru)
- 4yz prechodná negatívna odpoveď, dočasná chyba – napr. súbor je zamknutý na zápis (zapisuje iný užívateľ) a pod. Akciu možno opakovať
- 5yz permanentná negatívna odpoveď, napr. príkaz neimplementovaný

Druhá cifra znamená:

- x0z syntax – odpoveď súvisí so syntaxou (chyby, neimplementované parametre, nepodporované parametre)
- x1z informácie – odpoveď napr. na HELP, STAT
- x2z spojenie – odpovede ohľadom spojenia
- x3z autentifikácia a kontá
- x4z v RFC 959 zatiaľ nešpecifikované
- x5z súborový systém – odpoveď pre príkazy vykonávajúce činnosť so súbormi

Tretia cifra dáva ešte jemnejší význam odpovedi, zjemňuje druhú cifru. Tu treba skôr pozrieť na zoznam možných odpovedí daného príkazu a podľa toho vybrať správne číslo. Nové kódy odpovede môžu byť zavedené len ďalším rozšírením špecifikácie, rôzne implementácie protokolu na servroch by nemali pridávať nové vlastné kódy. V RFC 959 možno nájsť zoznam všetkých možných odpovedí zoradených podľa čísla, ale aj podľa skupín. Tiež ku každému FTP príkazu všetky možné odpovede a iné prehľadné tabuľky a výpisy.

2.7 Minimálna implementácia

Minimálna implementácia je potrebná pre servre, aby FTP mohlo fungovať. Všetky servre a klienti musia akceptovať a spĺňať aspoň túto minimálnu implementáciu:

Typ	– TYPE ASCII Non-print
Režim	– MODE Stream
Štruktúra	– STRU File, Record
Príkazy	– USER, QUIT, PORT, RETR, STOR, NOOP
	– TYPE, MODE, STRU pre základné nastavenia

FTPk bude okrem tejto minimálnej implementácie obsahovať takmer všetky FTP príkazy.

2.8 Záver, ukážka práce s FTP

FTP sa dnes bežne používa. Nevyužívajú sa však rôzne parametre a príkazy, ktoré boli robené hlavne pre mainframe a rôzne iné počítače. Dnešné počítače sú pomerne štandardné a tak dnes ľahko vystačíme s implementáciou protokolu, ktorá nepodporuje všetky možné príkazy a spôsoby prenosu.

Na záver ešte pár príkladov, toto je ukážka ovládania dvoch servrov cez control conneciton:

```
 User-PI - Server A                 User-PI - Server B
------------------                 ------------------

C->A : Connect                     C->B : Connect
C->A : PASV
A->C : 227 Entering Passive Mode.
       A1,A2,A3,A4,a1,a2
                                   C->B : PORT
                                          A1,A2,A3,A4,a1,a2
                                   B->C : 200 Okay
C->A : STOR                        C->B : RETR
           B->A : Connect to HOST-A, PORT-a
```

V tomto príklade C prezentuje klienta, A server A, B server B. Na začiatku sa klient pripojí na oba servre. Potom nastaví server A do pasívneho režimu použitím príkazu PASV. Server pošle kladnú odpoveď a tiež IP (A1,A2,A3,A4 sú 4B určujúce IP adresu) a port (a1,a2 sú 2B znamenajúce číslo portu), na ktorom očakáva prichádzajúce spojenie. Potom klient na servri B použije príkaz PORT, ktorého parametrom je IP a port, ktorý dostal od servra A ako odpoveď na PASV. Takto povie klient servru B, kam sa má napojiť v prípade, že dojde k prenosu súborov. Klient dostáva kladnú odpoveď. Nakoniec dá klient servru A príkaz pre uloženie súboru. Server A čaká na pripojenie. Potom klient dá servru B príkaz súbor odoslať, na čo sa server B pripojí na server A a prebehne prenos.

A ešte príklad typického priebehu jedného použitia FTP (viz nasledujúca strana):

Ľavý stĺpec ukážky obsahuje príkazy zadávané užívateľom svojmu FTP klientovi. Ten ich prekladá do FTP protokolu, posiela na server, spracúva odpovede a predkladá ich užívateľovi. Pravý stĺpec popisuje v skutočnosti klientovým programom vykonané činnosti. Používateľ pracuje na počítači U, server je S. ——> znamená príkazy odosielané z U na S, <—— odpovede servra odosielané užívateľovi. <CR> znamená odriadkovanie užívateľom (stlačenie <ENTER>).

Na začiatku sa používateľ pripojí k servru, odošle svoje prihlasovacie meno a potom aj heslo. Server pošle správu o úspešnom prihlásení. Potom užívateľ prenesie zo servra na svoj počítač U súbor. Použije pritom typ ASCII. Potom zmení typ na IMAGE a prenesie na server S súbor. Po tomto prenose ukončí prácu s programom a server ho odpojí.

Na tomto príklade si tiež možno všimnúť, že užívateľ používa na odriadkovanie iba <CR>, ale server používa v súlade s telnet protokolom <CRLF> [2].

```
LOCAL COMMANDS BY USER          ACTION INVOLVED

ftp (host) multics<CR>          Connect to host S, port L,
                                establishing control connections.
                                <---- 220 Service ready <CRLF>.
username Doe <CR>               USER Doe<CRLF>---->
                                <---- 331 User name ok,
                                          need password<CRLF>.
password mumble <CR>            PASS mumble<CRLF>---->
                                <---- 230 User logged in<CRLF>.
retrieve (local type) ASCII<CR>
(local pathname) test 1 <CR>    User-FTP opens local file in ASCII.
(for. pathname) test.pl1<CR>    RETR test.pl1<CRLF> ---->
                                <---- 150 File status okay;
                                      about to open data
                                      connection<CRLF>.
                                Server makes data connection
                                to port U.

                                <---- 226 Closing data connection,
                                    file transfer successful<CRLF>.
type Image<CR>                  TYPE I<CRLF> ---->
                                <---- 200 Command OK<CRLF>
store (local type) image<CR>
(local pathname) file dump<CR> User-FTP opens local file in Image.
(for.pathname) >udd>cn>fd<CR>   STOR >udd>cn>fd<CRLF> ---->
                                <---- 550 Access denied<CRLF>
terminate                       QUIT <CRLF> ---->
                                Server closes all
                                connections.
```

3 Súčasné FTP servre

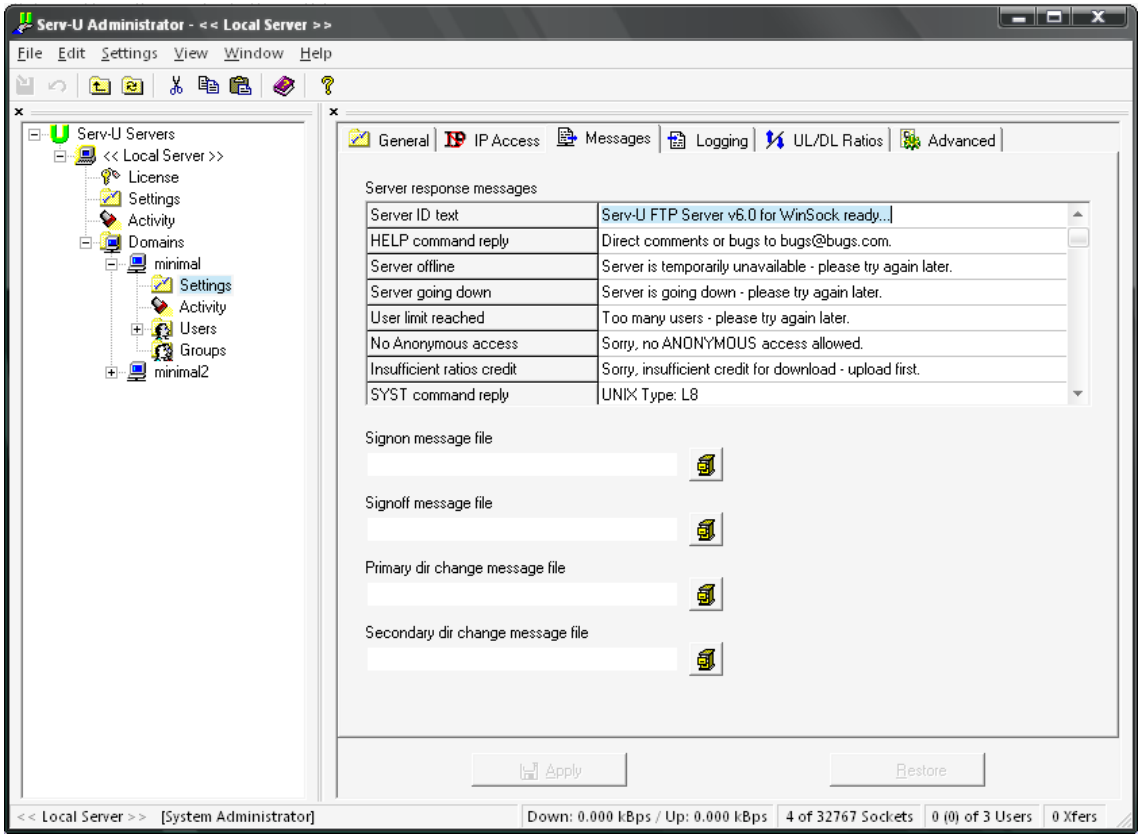
FTP je pomerne starý protokol a tak za čas jeho existencie stihlo vniknúť veľké množstvo klientov a servrov. Tiež vzniklo aj veľa programov, ktoré sú zamerané na nejakú úplne inú oblasť, ale na vnútornú komunikáciu používajú FTP protokol. Ten využívajú prostredníctvom rôznych FTP knižníc ako aj jednoduchších a na svoju špecifickú funkciu upravených servrov. Najrozšírenejšou skupinou, čo sa týka počtu inštalácií, sú obyčajné FTP servre orientované na užívateľa. Sú určené pre bežné použitie, prispôsobené pre funkčnosť a jednoduchosť. Obsahujú pomerne prepracované a pekné grafické nastavovacie programy. Väčšinou umožňujú správu vzdialených servrov, ale je potrebné použiť tento administrátorsky program dodávaný so samotným servrom. Mnoho z tých lepších FTP servrov patrí medzi komerčné, kde má užívateľ k dispozícii istú testovaciu dobu, počas ktorej funkcie nie sú obmedzované a po uplynutí tejto doby sa nasadia obmedzenia. Najčastejšími sú obmedzovanie počtu naraz pripojených klientov, rýchlosti prenosu, množstva klientských účtov a podobne. Medzi voľne šíriteľnými programami možno nájsť tiež veľmi kvalitné produkty. Veľmi špecializované servre sú skôr výnimkou, pretože tieto si dávajú pre svoje špecifické potreby buď priamo navrhovať, rozširovať alebo iba prispôsobovať rôzne spoločnosti. Takéto servre potom nachádzajú uplatnenie iba tam, kde možno využiť ich špecializovanosť, navyše často nie sú dostupné zdarma. V tejto kapitole budú rozoberané niektoré z bežných a veľmi rozšírených servrov, budú popísané ich dobré vlastnosti, ale aj oblasti v ktorých majú isté rezervy. Záver kapitoly (časť 3.4) obsahuje istý rozbor funkcií poskytovaných servrami, ich možné dopady na celkový výkon a posúdenie, ktoré z nich by bolo vhodné dorobiť do nami implementovaného FTPk.

3.1 Serv-U ftp server [P1]

Serv-U je komerčný server, ktorý poskytuje tridsaťdňovú testovaciu dobu a tak si možno veľmi dobre odskúšať všetky jeho vlastnosti a ponúkané funkcie. Server má vlastnú integrovanú správcovskú aplikáciu, ktorá umožňuje spravovať aj vzdialené servre. Ponúkané nastavenia sú veľmi podrobné a rozsiahle. Možno nastavovať dokonca jednotlivé buffre používane pri čítaní, zápise a prenose. Server podporuje virtuálne cesty. Administrátorský program vie zobraziť čo sa práve na servri deje, v prípade nejakých podozrení možno jednoducho užívateľov odpojiť, zakázať, alebo im len spomaliť rýchlosť.

Medzi nevýhody tohto servra patrí to, že niekedy nedokáže využiť maximálnu rýchlosť spojenia. Vďaka veľkému množstvu nastavení je tento server trocha viac náročný na procesor, napriek tomu ale táto záťaž procesora nie je limitujúcim faktorom výkonu, pretože je stále dostatočne nízka a limitom výkonu sa najčastejšie stáva rýchlosť disku. V pamäti server zaberá približne 4MB. Nastavovacia aplikácia asi 9MB. Program neumožňuje kompresiu log súborov. Celkovo sa jedná o pomerne dobrý program. Najlacnejšia verzia stojí $50. Personal edition verzia je zdarma, ale podporuje

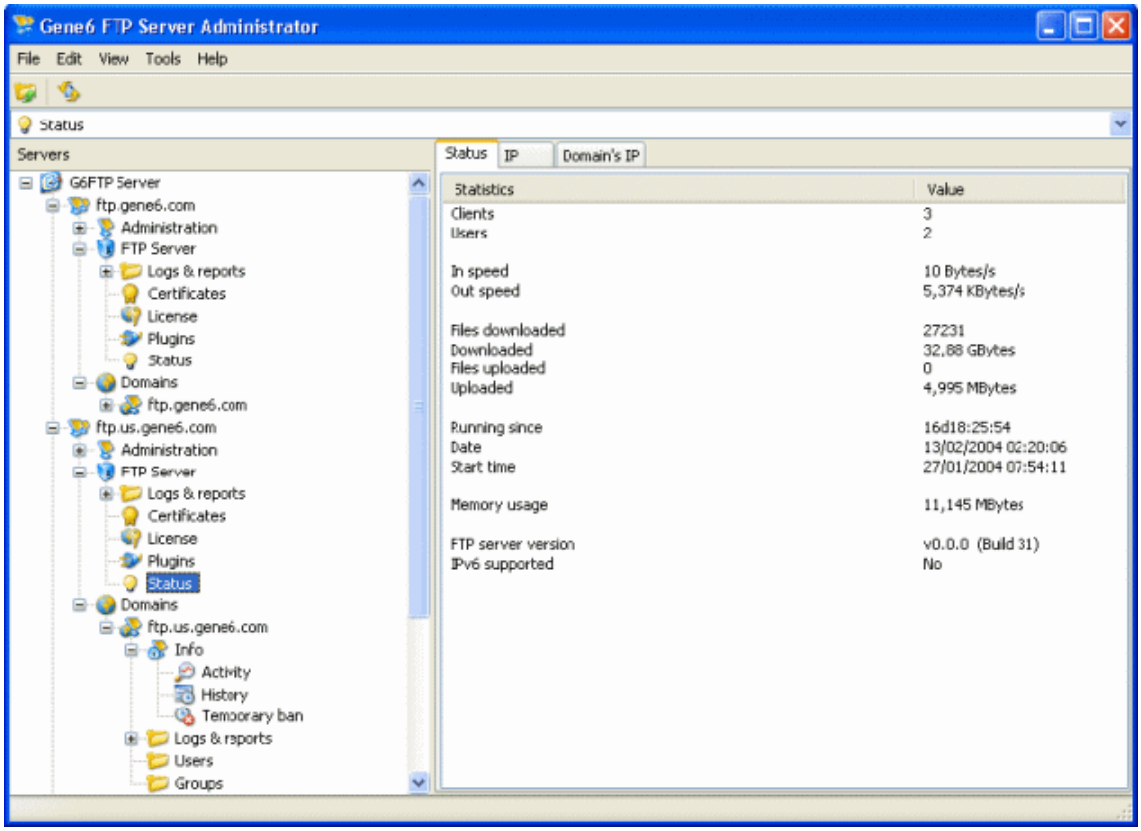
iba 5 uživateľských kónt, naraz maximálne dve súčasné pripojenia a skrátená je tiež o kvóty a virtuálne cesty. Nepodporuje ani vzdialenú správu.



Obrázok 1 – Serv-U administrátorský program

3.2 Gene 6 ftp server [P2]

Gene6 alebo G6 FTP server je tiež komerčným programom a obdobne ako Serv-U ho možno testovať 30 dní. Ponúka omnoho vyšší rozsah nastavení ako Serv-U, odhadom asi tak dva – tri krát väčší. Má veľmi prepracovanú prácu s užívateľmi a kontami a veľmi dobré bezpečnostné funkcie, kde vie sám odhaliť aj niekoľko jednoduchších typov útokov. Tento server má tiež integrovanú správcovskú aplikáciu. Oproti Serv-U je rýchlejší – vie naplno využiť rýchlosť spojenia, zaťažuje menej procesor. V pamäti ale zaberá viac – 16MB. Limitom rýchlosti je teda opäť výkon diskov. Server tiež neumožňuje kompresiu log súborov, ktoré môžu v tomto prípade narásť do veľmi veľkých rozmerov, keďže si možno nechať logovať veľmi rozsiahle množstvo informácií. Gene6 server by bol aj pomerne vhodný pre nasadenie na nejaký špecializovaný počítač s veľmi rýchlymi diskami, ktorý by takto zvládal obsluhovanie veľkého množstva klientov. Server podporuje aj jednoduché skriptovanie. Zaujímavou vlastnosťou servra je, že umožňuje využitie rôznych správcovských aplikácií. Celkovo sa jedná o veľmi dobrý, robusný a výkonný server, ktorý má skutočne široké možnosti nastavovania. Standard edition stojí podobne ako Serv-U $50. Po uplynutí testovacej doby ale server nemožno používať.



Obrázok 2 – Gene 6 administrátorský program

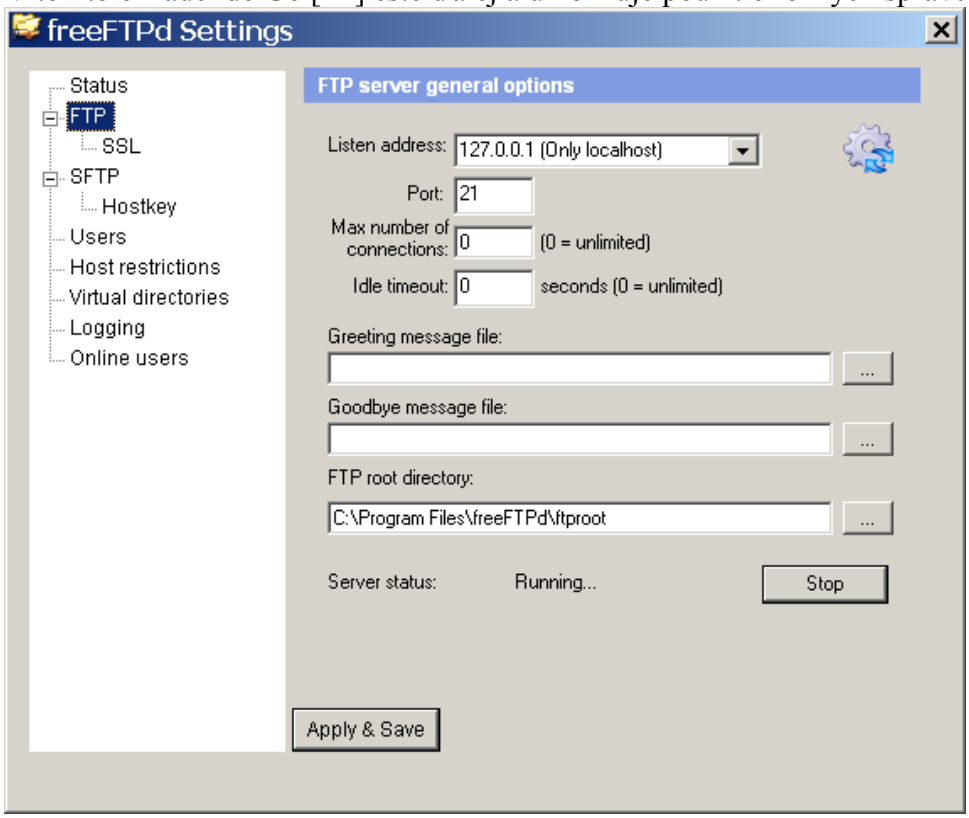
3.3 freeFTPd server [P3]

freeFTPd server je na rozdiel od dvoch predchádzajúcich zdarma. Neobsahuje toľko funkcií. Je to pomerne jednoduchý server. Neumožňuje vzdialenú správu. Na druhej strane, zaberá 11MB v pamäti. To je vzhľadom na množstvo poskytovaných funkcií pomerne veľa. Treba ale brať do úvahy že väčšinu tejto pamäte pravdepodobne zaberá grafický nastavovací program. Server umožňuje vytvárať virtuálne cesty, logovať do súboru. Ostatné nastavenia sú pomerne jednoduché. U užívateľov sa nedajú nastaviť ani prístupové práva. Na druhej strane, server podporuje secure FTP.

Z tohto popisu sa môže zdať, že sa jedná o úplne jednoznačné nevhodný server, treba si ale uvedomiť, aký je jeho účel. A ten je čo najjednoduchšie a najrýchlejšie sprístupniť údaje cez FTP. To server bez problémov zvláda – má jednoduchú a rýchlu inštaláciu a prehľadné nastavovacie rozhranie. Tiež ho nemožno porovnávať s platenými servrami. Vhodnejšie by bolo skôr porovnanie s týmito servrami, keď uplynie testovacia doba. Vtedy tieto servre umožnia len obmedzený počet pripojení, naproti tomu freeFTPd bude naďalej fungovať bez obmedzení.

3.4 Zhrnutie vlastností servrov

Všetky tri spomínané servre podporujú secure FTP. Podpora tejto vlastnosti vyžaduje, aby program zvládal komunikáciu prostredníctvom SSH. K implementácii SSH je potrebné použiť kryptovanie, a to je mimo oblasť tejto práce, ktorá sa chce zaoberať servrom samotným. Zaujímavou vlastnosťou je možnosť vzdialenej správy. V tomto ohľade ide G6 [P2] ešte ďalej a umožňuje použitie rôznych správcovských



Obrázok 3 – freeFTPd server nastavovací program

programov. Na druhej strane, ak sa užívateľ nerozhodne zaplatiť po uplynutí testovacej doby, dôjde k výraznému obmedzeniu funkčnosti servrov.

Preto sme sa rozhodli, že nami implementovaný FTPk server, by mal podporovať základné nastavenia, aké ponúkajú ostatné servre. Jedná sa najmä o podporu užívateľských kónt s právami a obmedzeniami prenosu. Ďalej bude podporovaná zmena nastavení servra ohľadom FTP protokolu – port control connection rozhrania, hello a system message servra. Server bude tiež poskytovať možnosť logovania aktuálneho diania do súboru a aj na obrazovku (ale v jednoduchšej forme). Server nebude obsahovať žiadne obmedzenia na maximálny počet pripojených klientov. Bude obmedzovaný iba systémovými prostriedkami.

Keďže sa bude jednať o efektívny server, tieto systémové prostriedky bude vedieť využiť úplne a nenechá žiadne rezervy. Dôležitá nebude len možnosť úplného využitia, ale tiež aby boli prostriedky využívané efektívne. Teda sa budeme snažiť, aby server nezaberal veľa v pamäti a tiež, aby veľmi nezaťažoval procesor pri spracovávaní FTP príkazov. Od tohto kritéria efektivity sa bude odvíjať aj množstvo a druh ďalších funkcií ponúkaných servrom.

V oblasti administrácie pôjde FTPk server ešte o krok ďalej ako Gene6 [P2]. Bude totižto umožňovať použitie aj cudzích programov pre administráciu. Aby bolo toto možné, navrhneme ADM protokol, pomocou ktorého bude prebiehať komunikácia so servrom. K servru doprogramujeme aspoň jeden vlastný administrátorský program,

aby bolo možné nastavovať server aj bez znalosti ADM protokolu. Pre ukážku možností samotného ADM protokolu implementujeme ešte aspoň jednu správcovskú aplikáciu.

FTPk server bude samozrejme zdarma a nebude obsahovať žiadnu testovaciu dobu, ani iné podmienky, ktoré by obmedzovali jeho použitie. Pre možnosti rozširovania bude podporovať pluginy.

4 FTPk server

Dielom tejto práce je predovšetkým samotný FTP server nazvaný FTPk, ale aj správcovské aplikácie určené pre zmenu jeho nastavení. V tejto časti práce budú popísane vlastnosti nami naprogramovaného FTPk servra, podporované funkcie, implementované FTP príkazy. Tiež sa popisujú možnosti správy a nastavovania, používania pluginov. Záver kapitoly obsahuje popis inštalácie, používania a minimálne požiadavky pre beh servra. Súčasťou textu tejto kapitoly je aj popis princípov fungovania servra.

4.1 Funkcie servra

FTPk je orientovaný viac na efektivitu ako veľké množstvo implementovaných funkcií. Podpora samotných FTP príkazov sa rozoberá v nasledujúcej podkapitole. Pre možnosti správy a zmeny nastavení podporuje FTPk protokol ADM, ktorý sme tiež navrhli. Jeho popis možno nájsť v kapitole 5. Pre rozširovanie funkcií možno tiež použiť pluginy (viz 4.6). Z funkcií zabudovaných priamo do servra možno spomenúť podporu logovania do súboru a do okna, podporu vzdialenej správy (s využitím ADM protokolu). Server umožňuje meniť HELLO a SYSTEM správy FTP protokolu, port na ktorom beží FTP rozhranie. Používateľ si môže tiež určiť port, na ktorom pobeží administrátorské rozhranie. Server beží iba na jednej doméne a túto umožňuje nastaviť. Server podporuje užívateľské kontá, pre každé konto možno nastaviť prihlasovacie meno, heslo, domovský adresár, prístupové práva a obmedzenia prenosu. Server podporuje iba jeden administrátorský účet, ktorý používa autentifikáciu prostredníctvom hesla. Tento účet slúži len pre používanie ADM protokolu, pre prístup k FTP treba vytvoriť samostatné užívateľské kontá. Pre detailné informácie o postupe zmeny nastavení viz kapitoly 5 a 6.

4.2 Princípy fungovania servra

FTPk je viacvláknovou [5] aplikáciou. Samotný program má vždy aspoň tri vlákna – hlavné vlákno programu, vlákno administrátorského rozhrania a vlákno čakajúce na pripojenia k FTP rozhraniu. Na každého pripojeného klienta sa spustí samostatný protocol interpreter thread, ktorý prijíma, vykonáva a posiela odpovede na klientove FTP príkazy. V prípade prenosu súborov sa spustí prenosový thread, ktorý prenesie daný súbor a potom sa skončí. Toto je potrebné z dôvodu nutnosti súčasne prenášať súbory ale aj spracovávať FTP príkazy. Po odpojení klienta sa protocol interpreter thread končí, aby uvoľnil systémové prostriedky.

Inou možnosťou konceptu programu by bola simulácia súčasného spracovania FTP príkazov viacerých napojených klientov pomocou ich ukladania do fronty a následného vykonávania. V takomto prípade by ale mohlo dôjsť k istému spomaleniu

reakcií. Všetci klienti, ktorý by boli vo fronte za klientom zadávajúcim dlhú dobu trvajúci FTP príkaz, by museli počkať. Navyše pri spadnutí spojenia by došlo k ďalšiemu zablokovaniu, pretože TCP/IP protokol nie je spoľahlivý a tak rozhranie čaká ešte istú dobu za stratenými paketmi a pod (viz [4]). Ak v prípade mnohovláknovej aplikácie dojde k nejakým problémom so spojením, zablokovaný ostane iba jeden protocol interpreter thread. Ostatní klienti môžu bez problémov a bez čakania pracovať ďalej. Informácia o cudzích problémoch sa k ním tiež nemá ako dostať, čo je v poriadku.

Ďalšou výhodou mnohovláknového konceptu je využiteľnosť viacerých procesorov. FTPk dokáže využiť toľko procesorov, koľko mu dá operačný systém k dispozícii. Samozrejme, na plné využitie bude potrebných mnoho súčasne pripojených klientov. Táto vlastnosť sa v dnešnej dobe viacprocesorových počítačov a viacjadrových procesorov javí ako veľmi dobrá. Naproti tomu koncept so simuláciou súčasného vykonávania FTP príkazov pomocou fronty by mal problém využiť v podstate už len druhý procesor. Pre ďalšie informácie o multiprocessingu viz [6].

4.3 Podpora FTP príkazov

FTPk server podporuje takmer všetky FTP príkazy špeficikované v RFC 959 [1] s výnimkou niektorých dnes už nepoužívaných režimov prenosu a typov súborov. Tieto ale možno k servru doplniť prostredníctvom pluginov (viz 4.6). Rovnako možno doplniť ľubovoľné ďalšie FTP, ale aj vlastné príkazy.

Následuje zoznam bez pluginov podporovaných príkazov:

USER PASS CWD CDUP QUIT PORT TYPE STRU REST RETR STOR MODE PWD STAT APPE STOU RNFR MKD PWD DELE RMD LIST ABOR RNTO NLST SYST HELP PASV NOOP SITE

Všetky príkazy sú podporované pre základné parametre.

4.4 Režimy behu servra

FTPk môže mať pri behu vlastné okno, do ktorého vypisuje aktuálne informácie o dianí na servri. Toto slúži však iba pre informácie a ukončovanie programu – zatvorením okna. Okno neumožňuje scrollovanie, po určitom množstve vpísaných znakov sa premaže. Preto sa odporúča pre potrebu záznamu používať možnosť logovania do súboru, kde sa zapíšu rovnaké informácie ako do okna. Dôležité je ubezpečiť sa, že na disku, kde je log súbor je dostatok miesta, v opačnom prípade po zaplnení disku FTPk odmietne ďalšiu prácu s tým, že nemá kam zapisovať log informácie.

Pre šetrenie pamäťou je možné program spúšťať aj bez okna. Ušetrí sa takto približne 1MB pamäte. Tento režim je výhodné použiť tiež tam, kde by nám stále prítomné logovacie okno aj po minimalizácii prekážalo. Pre vypnutie servra bez okna je nutné použiť task manager, alebo iný program, ktorý vie vypínať bežiace procesy.

4.5 Nastavovanie a možnosti správy

Samotný server, keďže je neinteraktívny, neumožňuje žiadnu zmenu nastavení. Akékoľvek zmeny sa vykonávajú prostredníctvom administrátorských programov, ktoré sú súčasťou diela tejto práce. Pre komunikáciu s týmito programami bol vytvorený ADM protokol, ktorý vďaka zverejnenej špecifikácii (viz kapitola 5) nemusia využívať iba dodané správcovské programy, ale ľubovoľné programy, ktoré tento protokol podporujú. Pre lokálne zmeny možno použiť ešte program USER, ktorý ale pracuje iba z príkazového riadku a dá sa povedať že predstavuje núdzové riešenie. Nevyžaduje však aby server bežal. Naopak, nemožno ho použiť, ak je server už spustený, pretože by mohlo došlo k poškodeniu súborov s nastaveniami, zákazmi a užívateľmi. Pre pohodlnejšiu prácu sa odporúča grafický program ADM (gui).

4.6 Podpora pluginov

FTPk podporuje pluginy. Načítavajú sa pri štarte a sú uložené v dynamickej knižnici plugins.dll. Táto knižnica momentálne neobsahuje žiadnú funkcionalitu, pretože tá bola naprogramovaná priamo do servra. Pluginy slúžia predovšetkým ako možnosť užívateľa doprogramovať si k už hotovému servru svoje ďalšie funkcie, a to aj bez toho aby mal k dispozícii zdrojový kód FTPk. Všetko, čo je potrebné pre písanie pluginov, je špecifikácia rozhrania pre komunikáciu s programom. Zdrojový kód plugins.dll už obsahuje predpripravené fukncie – ich názvy a deklarácia musí ostať rovnaká, zdrojový kód si k nim môže dopísať užívateľ ľubovoľný. Pre ďalšie podrobnosti a technické detaily viz príloha – programátorská dokumentácia k FTPk.

Pluginy boli pôvodne mienené pre využitie ako náhrady pre chýbajúce režimy prenosu (compressed) a typov (EBCIDC), navyše pribudla možnosť pluginu pre doplnenie fukncie spúšťanej pri štarte programu. Pomocou pluginov tiež možno doplniť FTP príkaz ALLO, resp. ľubovoľné ďalšie príkazy. Ako ukážka je implementovaná funkcia spúšťajúca sa pri načítani pluginov, ktorá vypíše text TEST do logovacieho okna FTPk.

4.7 Inštalácia a používanie

Program nevyžaduje žiadnu špeciálnu inštaláciu, stačí ho spustiť. Pre svoj beh ale vyžaduje, aby bol spustený v rovnakom adresári, ako sa nachádzajú súbory s nastavením (main.dat), zákazmi (ban.dat) a užívateľskými kontami (user.dat). Program priložený na CD obsahuje základné nastavenia. Odporúča sa tieto nastavenia hneď po prvom spustení zmeniť a to predovšetkým administrátorské heslo, práva a domáce adresáre klientských kónt, prípadne nastaviť zákazy pre vybrané IP adresy. Pre zmenu nastavení je požadovaná možnosť zápisu, program teda možno spustiť napr. z CD, ale nemožno potom meniť žiadné nastavenia, navyše pri pokuse o zmenu dojde k prerušeniu fuknčnosti servra.

4.8 Minimálne požiadavky pre beh FTPk

Server požaduje operačný systém MS Windows 2000, alebo lepší. Možno by fungoval aj na nižších verziách, ale nemožno zaručiť, že by program fungoval vo všetkých prípadoch správne. Čo sa týka požiadaviek na hardware počítača, postačia minimálne požiadavky pre beh operačného systému. FTPk podporuje všetky súborové

systémy, ktoré podporuje aj operačný systém na ktorom beží. Teda NTFS aj FAT, a to všetky verzie, ktoré podporuje operačný systém. Server zaberá v pamäti od 5MB, v závislosti od počtu pripojených užívateľov lineárne narastá spotreba pamäte – približne o 20 KB na jedného pripojeného užívateľa.


FTPk log window
WM_CREATE: Vytvara sa logovacie okno...
Uspesna inicializacia winsock2
Uspesne vytvorenie main socketu
Uspesne nabindovanie main socketu na port
Uspesne vytvorenie admin socketu
Uspesne nabindovanie admin socketu na port
Uspesne vytvorenie data socketu
Uspesne nabindovanie data socketu na port
Spustam mainTHREAD...
Spustam adminTHREAD...
mainTHREAD na porte 21 spusteny...
adminTHREAD na porte 10000 spusteny...
_MAX_PATH je 260B
Pokus o pripojenie na admin thread
Akceptovane pripojenie na admin porte 10000
Pripaja sa 127.0.0.1:1052
Zacinam spracuvavat prichadzajuce ADM prikazy...
heslo
Bytes Recv: 5
user L
Bytes Recv: 6
MAIN G
Bytes Recv: 6
USER L
Bytes Recv: 6
BAN L
Bytes Recv: 5
MAIN G
Bytes Recv: 6
USER L
Bytes Recv: 6
USER L
Bytes Recv: 6
USER A THX
C:\users\THX
ftp
0
0
1
0
1
0
Bytes Recv: 49
USER L
Bytes Recv: 6

5 ADM protokol

V tejto časti je popísaný ADM protokol, ktorý sme tiež vytvorili v rámci tejto práce. Sú tu rozoberané dôvody vzniku, príčiny ktoré formovali protokol do aktuálnej formy, princípy fungovania. Jednotlivé príkazy protokolu sú rozoberané v samostatnej podkapitole. Túto kapitolu možno považovať za dostatočne presné zverejnenie špecifikácie protokolu.

5.1 Úvod k protokolu

Pre možnosť vzdialenej správy servra sme museli vyvinúť nejaký spôsob komunikácie servra a programu pre správu. Istou inšpiráciou pri vývoji a pre aktuálnu podobu protokolu bol samotný FTP protokol a jeho spôsob fungovania. Takáto forma komunikácie so sebou prináša isté výhody ako sú prístupnosť protokolu užívateľovi a možnosť priamej práce prostredníctvom protokolu. Ale tiež aj nevýhody, medzi ktoré jednoznačne patrí vyššia náročnosť strojového spracovania a s tým spojená vyššia zložitosť samotného servra. Každopádne, výhody prevážili nad nevýhodami aj preto, lebo tento protokol, ak sa zverejní jeho špecifikácia, budú môcť využívať aj cudzie programy pre správu servra. Takýto program si môže každý upraviť pre svoje potreby, môže si doň implementovať svoje veľmi špecifické funkcie, ktoré by inak ťažko (alebo dokonca vôbec ne-) vykonával prostredníctvom funkcií dostupných v existujúcich programov. Takto nie je problém urobiť si skript, ktorý z nejakého súboru (napr. zoznam študentov a ich oborov) vygeneruje zodpovedajúce kontá na servri a nastaví príslušné obmedzenia prístupových práv. Možno niekto bude potrebovať pre všetkých používateľov raz za mesiac generovať nové heslá. Alebo po zmene prihlasovacieho hesla, zmeniť heslo pre FTP prístup na rovnaké. Alebo pri nedostatku miesta na disku dočasne zaviesť kvóty pre upload a pod. Možnosti je skutočne veľa, ale vďaka tomu, že špecifikácia je zverejnená, nie je problém realizovať aj zložitejšie operácie. Administrátorom servra sa takto dostáva viac slobody a priestoru pre správu.

5.2 Princípy fungovania

Nami navrhnutý ADM protokol funguje na podobných princípoch ako FTP. Samotný protokol pre svoj prenos nevyžaduje žiaden ďalší podprotokol, ako je telnet u FTP, ale prenáša sa priamo po spojení. Iniciátorom spojenia je vždy klient. Po pripojení na servrov administrátorský port začne prebiehať komunikácia. Ako prvé pošle klient servru administrátorské heslo. Všetok prenos po spojení je plain text – príkazy klienta, ako aj odpovede servra. Pre ukončovanie príkazov pred odoslaním servru (nie jednotlivých príkazov, ale kompletného príkazu aj so všetkými parametrami) možno použiť CRLF (teda jednoducho povedané <ENTER>), ale nie je to nutné.

Administrátorské heslo sa posiela tiež ako text. Toto nie je veľmi bezpečné, ale cieľom tejto práce nebola implementácia vysokej bezpečnosti a s ňou spojeného šifrovania, ale skôr vyššej efektivity. Toto je tiež priestor pre budúce zlepšovanie servra. Heslá by sa mohli posielať ako MD5 alebo SHA1 hashe a celé spojenie prebiehať cez SSH. Takto by sa dosiahla pomerne vysoká úroveň bezpečnosti, navyše tieto funkcie

možno prostredníctvom externých knižníc pomerne rýchlo a jednoducho doplniť do programu. Potom by ale výkonnosť programu závisela aj od rýchlosti týchto knižníc. Navyše by bolo nutné spoľahnúť sa na to, že tieto knižnice fungujú skutočne tak ako majú. Alebo ich naprogramovať, ale to je mimo rámec tejto práce. Ak bolo heslo správne, server o tom oboznámi klienta (`Password OK, logged in`) Alebo v prípade nesprávneho hesla pošle správu `Wrong password, disconnecting`, po ktorej nasleduje odpojenie.

Po prihlásení sa môže začať komunikácia prostredníctvom ADM protokolu. Tá prebieha tak, že klient pošle príkaz (vo forme plain textu s, či už bez odriadkovania na konci) a dostane od servra odpoveď, ktorá je tiež vo forme plain textu ale vždy s odriadkovaním na konci. Formát ADM príkazov a aj odpovedí servra je istým kompromisom medzi užívateľskou prívetivosťou a možnosťou efektívnejšieho strojového spracovania. A to jednak na strane servra, ktorý rozpoznáva klientove príkazy a ich parametre, ako aj na strane klienta, kde môže bežať nejaký program, ktorý odpoveď servra napríklad graficky znázorní a preto ju musí tiež rozpoznávať. Syntax jednotlivých príkazov a odpovedí na ne je popísaná v podkapitole 5.3. Pri ukončovaní komunikácie dôjde k zrušeniu spojenia zo strany servra. Príkazy nie sú case sensitive, parametre áno. Ak by pri písaní príkazu došlo napr. k preklepu, alebo by bol napísaný nepodporovaný príkaz, server odpovie `Command not recognized.`

Komunikovať pomocou ADM protokolu možno nie len cez spojenie na administrátorský port servra, ale aj cez samotne FTP control connection pomocou FTP príkazu SITE. Za príkazom nasleduje medzera a príkaz ADM protokolu. Komunikácia, syntax, príkazy a odpovede servra sú rovnaké ako v prípade napojenia na administrátorský port. Výnimkou je prípad nesprávne zadaného hesla, kedy nedôjde k odpojeniu klienta, pretože tento používa FTP control connection a odpojenie by znamenalo zrušenie možnosti práce s FTP. Podobne pri ukončovaní práce s ADM protokolom tiež nedôjde k prerušeniu spojenia.

5.3 Popis príkazov

Táto podkapitola obsahuje zoznam v momentálnej verzii protokolu podporovaných ADM príkazov, spolu s ich popisom a formátom odpovede na ne. Prvé slovo je príkaz, nie je case sensitive. Všetko, čo nasleduje za medzerou po príkaze, je parametrom, ktorý ale case sensitive je. Pre vyššiu prehľadnosť sa pri popise rôznych parametrov toho istého príkazu bude ako príkaz uvádzať celé znenie príkazu aj s parametrom. Text v <zátvorkách> znamená nahradenie odpovedajúcim reťazcom:

`<username>` meno užívateľa, nanajvýš 200 znakov

`<password>` prihlasovacie heslo užívateľa (plaintext), nie je case sensitive, nanajvýš 200 znakov

`<ip>` IP adresa, zapísaná vo forme textu, nanajvýš 20 znakov

`<rwxw>` určuje prístupové práva užívateľa. Ak má užívateľ dané právo, uvedie sa v reťazci dané písmeno. Ak nemá, tak - .

Prvé písmeno je pre právo čítať súbory, druhé modifikovať, vytvárať a mazať súbory, tretie vstup do adresára (výpis súborov v adresári), posledné modifikovať vytvárať a mazať adresáre.

r-x- teda znamená právo čítať súbory a vstupovať do adresárov, rwxw znamená plne práva, ---- žiadne práva

<homedir> domáci adresár užívateľa na lokálnom disku, vždy úplná cesta, nanajvýš 260 znakov

<#number> celé kladné číslo (32 bitový integer)

<userinfo> <username> <homedir> <password> <rwxw> <#uploadlimit> <#downloadlimit>

userinfo je reťazec popisujúci kompletné informácie o užívateľovi. Obsahuje polia oddelené medzerou. Posledné dve polia sú nepovinné (server ich v odpovedi ale vypisuje stále), znamenajú upload a download limit v B. Limit 0B znamená žiadne obmedzenia

Nasleduje zoznam príkazov a odpovedí servra na ne:

Príkaz
->Odpoveď servra
a popis príkazu.

BYE
->Ending session, disconnecting...
Príkaz slúži pre ukončenie práce. Ak je klient napojený na administrátorský port servra, dôjde k zrušeniu spojenia zo strany servra.

PING
->PONG
Jedná sa o obdobu FTP príkazu NOOP, ktorý tiež nevykonáva nič a slúži predovšetkým pre overenie dostupnosti servra. Odpoveďou servra je vždy PONG.

USER L
-><Name: <username> PW: <password> Home: <homedir> r:<#> w:<#> X:<#> W:<#> up:<#>B down:<#>B>
Príkaz vypíše zoznam všetkých užívateľov, pričom na každom riadku sa nachádzajú informácie o jednom užívateľovi. Pri právach 0 znamená, že užívateľ dané právo nemá, 1 znamená že užívateľ právo má.

USER R <username>
->User not found!
Príkaz slúži na odstránenie užívateľa s menom <username>.
Toto je odpoveď, ak neexistuje užívateľ s daným menom.
Žiadna odpoveď znamená úspešné odstránenie užívateľa.

USER A <userinfo>
->User already exists!
Príkaz slúži na pridanie užívateľa s menom <username>.
Toto je odpoveď, ak neexistuje užívateľ s daným menom.
Žiadna odpoveď znamená úspešné pridanie užívateľa.

```
BAN L
-><ip>
```
Príkaz vypíše zoznam zakázaných IP adries.
Na každom riadku odpovede sa nachádza jedná IP adresa, ktorá ma zakázané pripájanie k servru.

```
BAN R <ip>
->IP not found!
```
Príkaz odstraňuje zákaz pripojenia pre adresu <ip>.
Toto je odpoveď, ak neexistuje daná IP adresa v zozname zakázaných
Žiadna odpoveď od servra znamená úspešné odstránenie užívateľa.

```
BAN A <ip>
```
Príkaz pridáva zákaz pre danú ip adresu.
```
->Ip to be banned already exists in the banlist!
```
Ak už existuje daná IP adresa v zozname zakázaných.
```
->Added ip into the ban list
```
Ak prebehlo pridanie adresy do zoznamu zakázaných úspešne.

```
MAIN G
-><  <adminPassword>
     <hellomessage>
     <serverIP>
     <logtofilepath>
     <systemmessage>
     <#adminport>
     <#log?>
     <#logtofile?>
     <#port> >
```
Táto odpoveď obsahuje popis hlavných nastavení servra. Jednotlivé položky nastavenia nie sú oddelené medzerami, ale každá je na samostatnom riadku. U <#log?> a <#logtofile?> 1 znamená, že sa daná činnosť vykonáva, 0 že sa nevykonáva. Maximálne dĺžky pre jednotlivé polia sú 200 znakov pre adminPassword, 200 znakov pre hellomessage, 20 znakov pre serverIP, 260 znakov pre logtofilepath, 200 znakov pre systemmessage, port a adminport je číslo od 1 do 65535, pre log a logtofile možno použiť iba 1 a 0.

```
MAIN S <  <hellomessage>
          <systemmessage>
          <serverIP>
          <logtofilepath>
          <adminPassword>
          <#port>
          <#log?>
          <#logtofile?>
          <#adminport> >
```
Server neposiela žiadnu odpoveď, len nastaví požadované údaje.
Pre maximálne veľkosti jednotlivých polí viz popis predchádzajúceho príkazu.

6 Administrátorské programy

Táto časť práce je venovaná administrátorským programom, ktoré sme vytvorili tiež v rámci tejto bakalárskej práce. Čiastočne slúži aj ako náhrada užívateľskej príručky. Jednotlivé programy majú popísané svoje funkcie, postupy inštalácie, spúšťania a práce s nimi. Samotný FTP server nevyžaduje nejaký konkrétny administrátorský program. Stačí, ak program komunikuje pomocou ADM protokolu. V tejto kapitole ale budú rozoberané iba programy, ktoré sú súčasťou tejto práce, možnosť používania ostatných programov sa rozoberá v kapitole 7.

6.1 Úvod

Dôvodom pre naprogramovanie týchto troch administrátorských programov bolo ukázať využitie možností ponúkaných ADM protokolom, a tiež aby sa server spolu s týmito programami stal kompletnejším a plnohodnotnejším dielom. Programy reprezentujú dva rôzne koncepty. Prvý z nich je grafický a viac užívateľsky prívetivý (ADM gui), obsahuje však aj konzolovú časť, kde možno sledovať a zasahovať do komunikácie. Druhý predstavuje konzolovú aplikáciu (ADM konzolový), ktorá nemá veľa funkcií a používateľ musí prostredníctvom ADM protokolu komunikovať samostatne, bez asistencie programu. Výhodou takéhoto programu je jeho jednoduchosť, podpora všetkých možných budúcich vylepšení a zmien ADM protokolu, a tiež jeho veľkosť. Posledný program USER pracuje iba lokálne a z príkazového riadku, nevyžaduje ale beh servra.

6.2 ADM (gui)

Program ADM (gui) sme vytvorili na administráciu nášho FTP servra FTPk, na ktorý sa napojí a posiela mu príkazy a spracováva odpovede servra na tieto príkazy. Požiadavky pre beh programu sú rovnaké ako požiadavky FTPk, teda postačí minimálna konfigurácia pre systém Windows 2000, ktorý je požadovaný tiež. ADM (gui) nie je potrebné inštalovať, stačí ho rovno spustiť. Program pozostáva z jednotlivých záložiek, pričom každá z nich slúži na nastavovanie nejakých iných parametrov servra – viz konkrétny popis ku každej záložke.

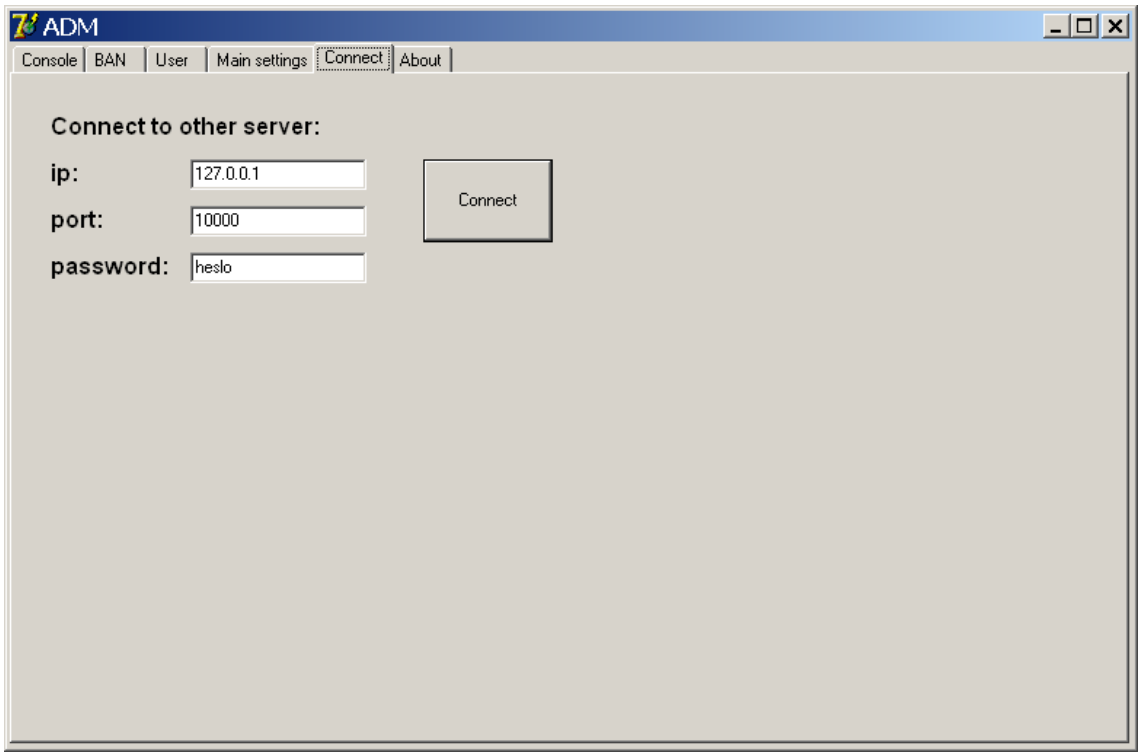
Jednotlivé vyplnené položky buď predstavujú štandardné nastavenia, alebo aktuálne nastavenia servra – podrobnosti pri popise každej záložky. Pred začatím nastavovania servra, je nutné sa naň napojiť – pomocou záložky CONNECT. Program umožňuje spravovať ľubovoľný FTPk, na ktorý je schopný sa napojiť, t.j. aj vzdialené servre. ADM (gui) je vďaka svojmu princípu a princípu fungovania administrátorského rozhrania FTPk univerzálny pre všetky (aj budúce – ak budú mať stále rovnaký princíp fungovania admin rozhrania) verzie FTPk. Táto čiastočná kompatibilita je zabezpečená záložkou CONSOLE, kde môže užívateľ priamo zadávať príkazy a čítať odpovede servra na ne. Samozrejme, staršie verzie ADM (gui) nebudú vedieť zobraziť tieto odpovede nových ADM príkazov do jednotlivých záložiek, ktoré možno ani nebudú obsahovať. ADM (gui) sa pripája na administrátorský port servra a neumožňuje fungovanie cez FTP control connection prostredníctvom použitia príkazu SITE.

Následuje popis jednotlivých záložiek programu:

ABOUT

V tejto záložke možno nájsť nejaké informácie o programe a jeho verziu. Momentálna verzia 1 obsahuje iba odkaz na túto dokumentáciu. V tejto záložke sa nič nenastavuje a týka sa iba programu ADM (gui).

CONNECT



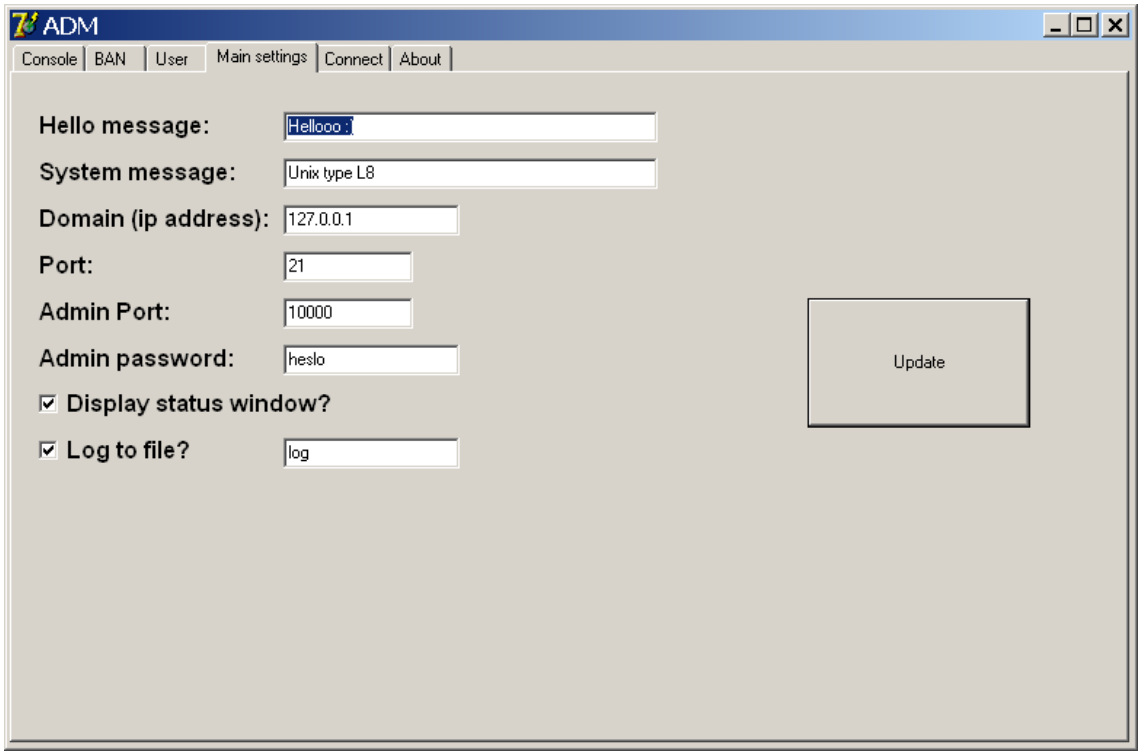
Obrázok 6 – ADM (gui) Connect záložka

Tu sa pripája na server, ktorý chceme spravovať. V príslušných poliach sa zadá IP adresa servra, port administrátorského rozhrania a heslo. Polia majú predvyplnené štandardné hodnoty. Samotné pripojenie nastane po kliknutí na tlačidlo connect. Výsledok pokusu o pripojenie si môžeme pozrieť v hlavnej konzole, kde by mala byť odpoveď od servra o úspešnom pripojení. V prípade neúspechu sa odporúča skontrolovať správnosť vyplnených údajov, nastavenie firewallu a pod., resp. opakovať pokus o pripojenie. Táto časť programu nepoužíva ADM protokol a po pripojení na server (a samozrejme ani pred pripojením) neumožňuje meniť žiadne nastavenia servra.

MAIN SETTINGS

Táto záložka umožňuje meniť hlavné nastavenia servra, a to port, doménu, port a heslo administrátorského rozhrania, HELLO a SYSTEM message servra, možnosti zobrazovania okna samotného servra a nastavenie logovania do súboru. Vždy pri kliknutí na túto položku sa predvyplnené informácie aktualizujú podľa súčasného nastavenia servra. Po kliknutí na update sa nové nastavenia odošlú na server a ten si ich uloží. Nedojde avšak k okamžitému použitiu, pretože server beží. Preto nie je chybou ak po kliknutí na update, odchodom do inej záložky a opätovnom návrate do main settings

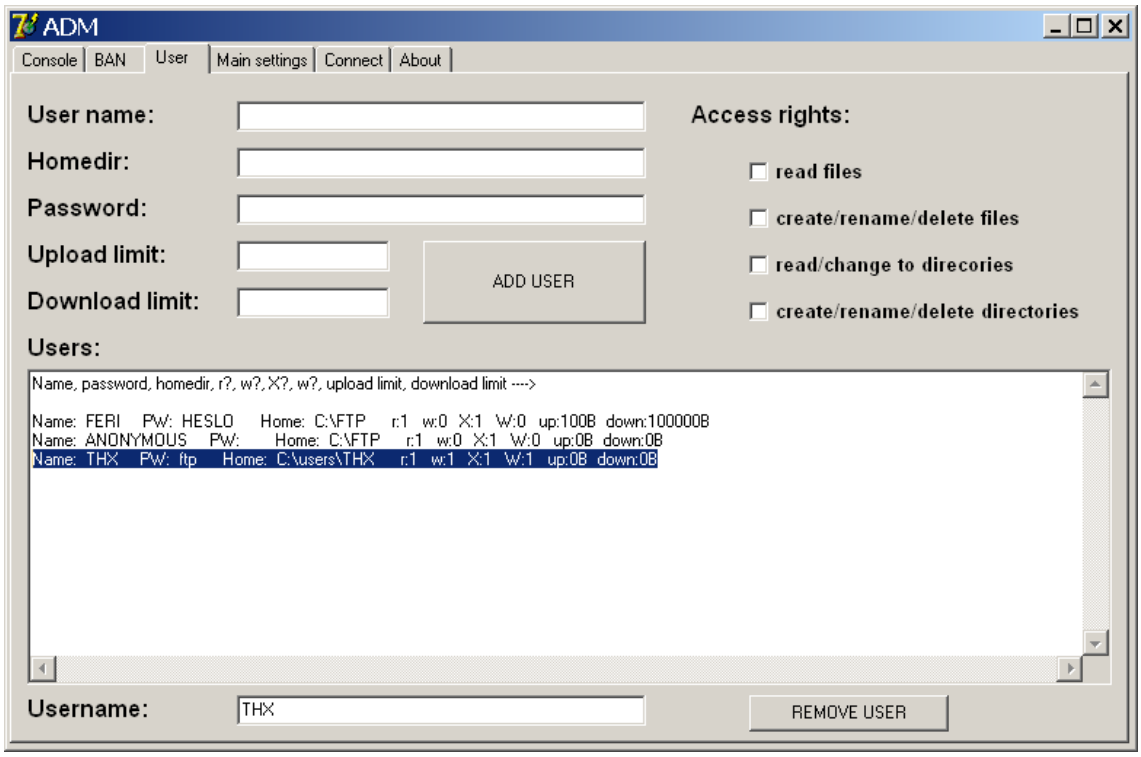
sa predvyplnia staré hodnoty namiesto tých ktoré boli odoslané kliknutím na update. Tieto nové nastavenia sa uplatnia až po reštarte servra. Aktuálne nastavenie sa počas behu servra nemení. Všetky nastavenia ktoré možno meniť v tejto záložke si server ukladá do súboru main.dat. Dĺžky hello, system messageov, admin hesla sú obmedzené na 190 znakov. Dĺžka cesty k log súboru na 255 znakov. ADM príkazy, ktoré sú používané touto záložkou, sú MAIN G – pre získanie informácií o nastavení servra. Tento príkaz sa pošle vždy po vstupe do tejto záložky. Program ADM (gui) potom spracuje odpoveď servra a podľa nej nastaví hodnoty príslušných polí záložky. Naopak pri odoslaní informácií na server sa vyplnené informácie tejto záložky transformujú na parameter príkazu MAIN S , ktorý je následne poslaný servru.



Obrázok 7 – ADM (gui) Main settings záložka

USER

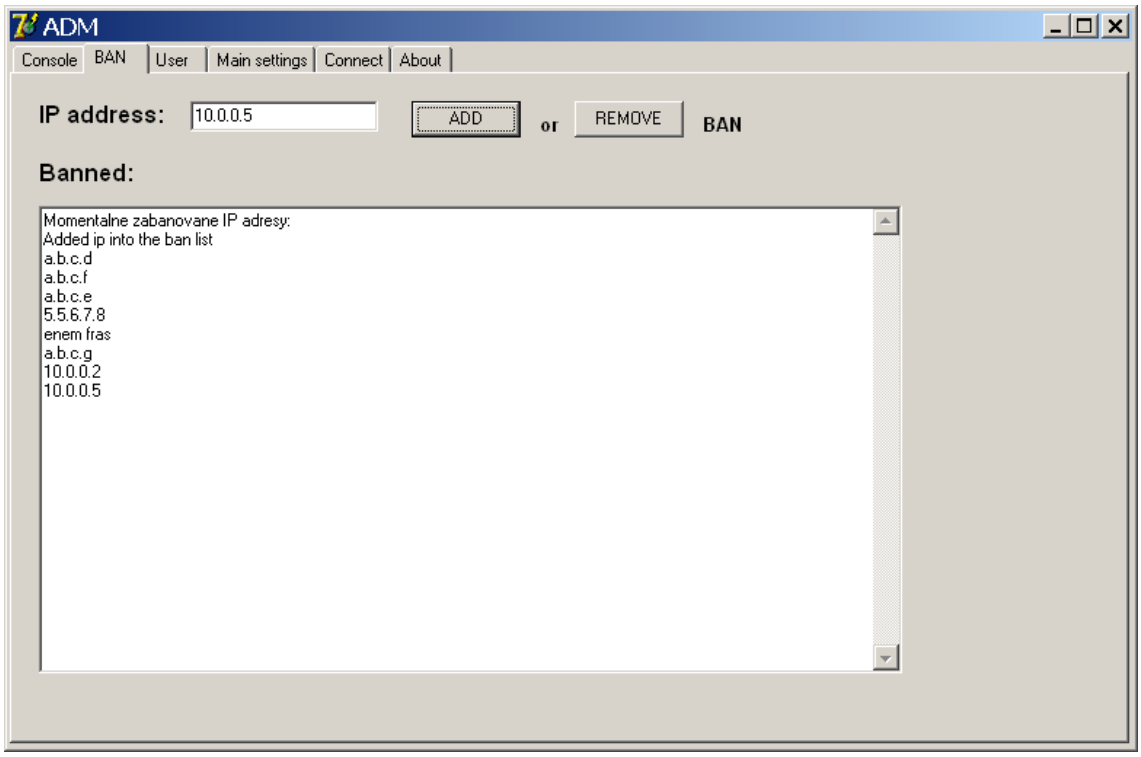
Záložka USER slúži na správu užívateľov. Nastavujú sa tu všetky parametre pre užívateľov – meno, heslo, domáci adresár, kvóty a práva prístupu k súborom a adresárom. Užívateľov možno pridávať, ako aj odoberať. Username a Password majú obmedzenie dĺžky reťazca na 190 znakov, homedir na 255. Vždy pri zmene záložky na USER sa aktualizuje zoznam užívateľov – pomocou príkazu USER L, zoznam užívateľov, spolu s nastaveniami pre každého z nich možno vidieť v spodnom textovom poli záložky. Pri pridávaní užívateľa sa vyplnené polia transformujú do parametru ADM príkazu MAIN A. Pri odoberaní je poslané meno užívateľa ako parameter príkazu MAIN R.



Obrázok 8 – ADM (gui) User záložka

BAN

Slúži na zakázanie prístupu na server. Vždy pri zmene na záložku BAN sa aktualizuje zoznam zakázaných IP adries podľa aktuálneho stavu. Pri zadávaní adries na zákaz, alebo zrušenie zákazu sa neberie ohľad na to, či je IP adresa správna. Ak je IP adresa nesprávna, tak nebude mať zákaz žiaden účinok, pretože sa nikdy nikto nepripojí s takouto chybne zadanou adresou, a teda mu ani nebude môcť byť obmedzený prístup. V budúcnosti bude možno tento zoznam použiť aj pre protokol Ipv6, ktorý má dlhšie adresy ako Ipv4. IP adresa, ktorá sa zadáva na zákaz, alebo zrušenie zákazu, má obmedzenie na maximálnu dĺžku 16 znakov. Obdobne ako pri iných záložkách, pre získavanie informácií sa používa ADM príkaz BAN L. Pre pridávanie a odoberanie zákazov sa používa BAN A a BAN R, kde program transformuje text vyplnených polí záložky na parameter týchto príkazov, ktorý odošle servru spolu s príslušnými ADM príkazmi.



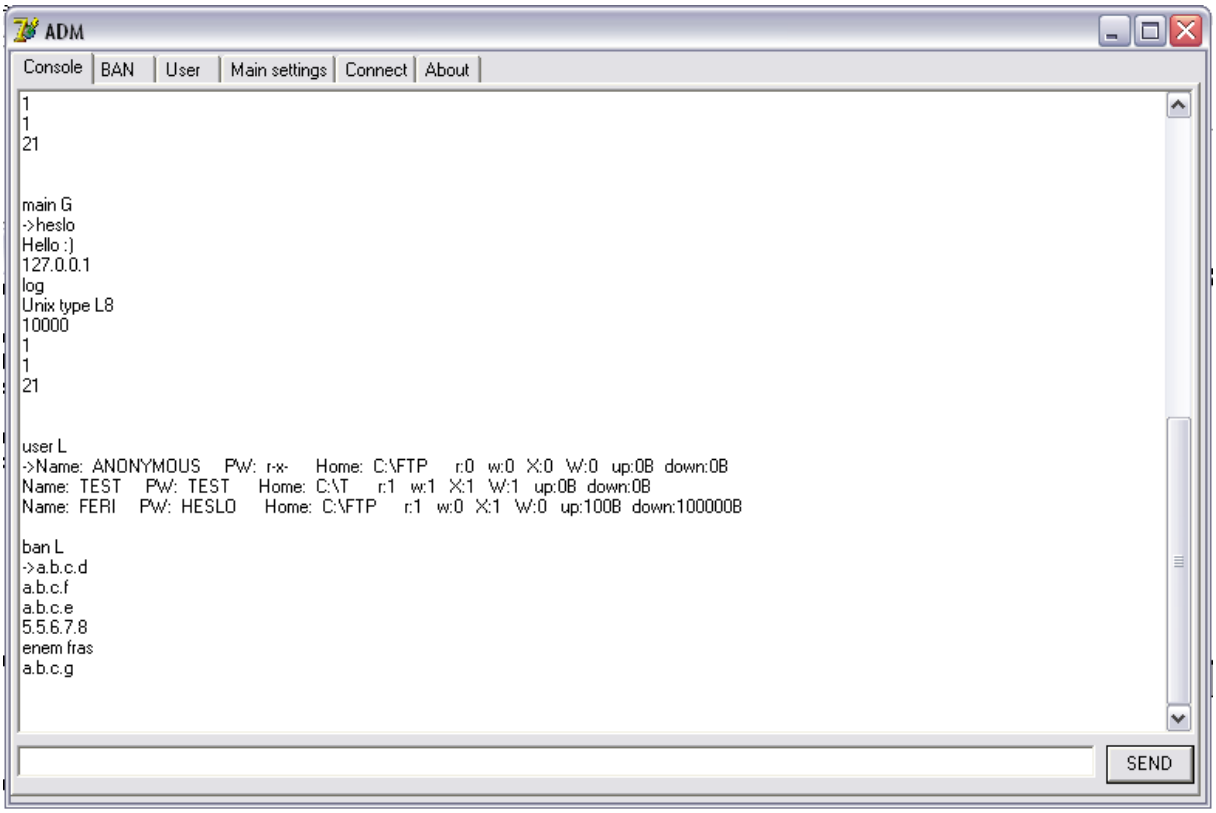
Obrázok 9 – ADM (gui) BAN záložka

CONSOLE

Console predstavuje najzaujímavejšiu časť celého programu. Ide síce v podstate iba o jednoduché rozhranie. Vďaka systému pracovania rozhrania ADM programu FTPk, možno ale z konzoly robiť všetky nastavenia, ktoré umožňujú ostatné záložky. Dokonca (ako už bolo spomínané v úvode) možno cez konzolu ovládať aj budúce verzie programu FTPk, ktoré budú podporovať ADM rozhranie a protokol. Príkazy konzoly sú priamo príkazmi ADM protokolu. Tiež aj odpovede servra, ktoré sa zobrazujú v konzole, predstavujú priame a nemodifikované odpovede podľa ADM protokolu. Pre popis príkazov a odpovedí ADM protokolu viz kapitolu 5. Užívateľ môže kedykoľvek začať cez konzolu pracovať priamo využívajúc ADM protokol. Musí ale rátať s tým, že keď vstúpi do záložky main settings, príkaz tejto záložky sa okamžite objavia v konzole. Podobne pri pridávaní užívateľa sa do konzoly dostane príslušný príkaz ADM protokolu. Takto možno v podstate kontrolovať, že program ADM (gui) robí skutočne to, čo má, a prípadne identifikovať problémy. Ak napr. chýba odpoveď servra, pravdepodobne došlo k prerušeniu spojenia – dá sa to ľahko overiť pomocou ADM príkazu PING. Záložky BAN a USER ale svoju prácu nevypisujú do konzoly, pretože v prípade dlhého výpisu užívateľov a zákazov by to bolo užívateľovi pracujúcemu cez konzolu na obtiaž.

Postupne s vývojom jadra FTPk môžu pribúdať ďalšie príkazy, pre ich jednoduché využitie bude potrebná nová verzia programu ADM (gui), kde budú musieť byť upravené (a prípadne pridané nové) záložky, avšak cez konzolu pôjdu použiť už z prvej verzie.

Obrázok 10 zobrazuje prácu s konzolou, všimnite si, že príkazy sú písané malým písmom, teda ich zadával užívateľ, ADM (gui) používa vždy iba veľké písmená.



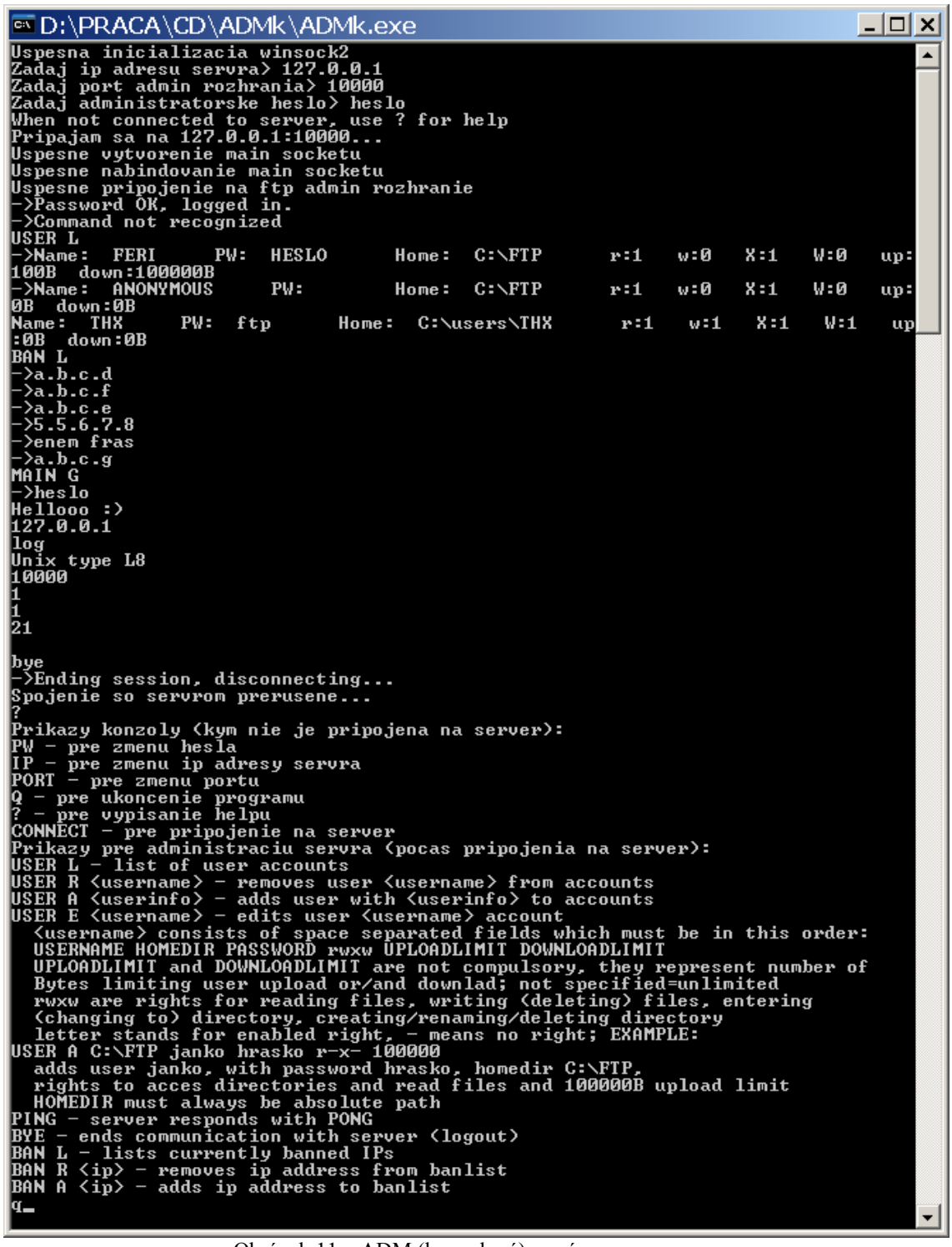
Obrázok 10 – ADM (gui) Console záložka

6.3 ADM (konzolový)

V porovnaní s ADM (gui) je ADM (konzolový) jednoduchším programom. Obsahuje v podstate iba konzolu, prostredníctvom ktorej môže užívateľ komunikovať so servrom cez ADM protokol. Program sa podobne ako ADM (gui) pripája na administrátorský port servra a neumožňuje prácu cez FTP spojenie pomocou použitia príkazu SITE. Konzolový ADM beží v textovom režime a nevyžaduje žiadnu inštaláciu, program stačí iba spustiť.

Konzola programu má dva režimy. Prvý, lokálny, je aktívny keď nie je program pripojený na server, teda tiež hneď po spustení programu. Po pripojení sa konzola prepne do ADM režimu, kde používateľ komunikuje priamo so servrom cez ADM protokol, môže teda využívať všetky ADM príkazy, ktoré boli popísané v kapitole 5. Následuje popis lokálnych konzolových príkazov – príkazy sa tak ako ADM protokolovské odosielajú stlačením <enter>:

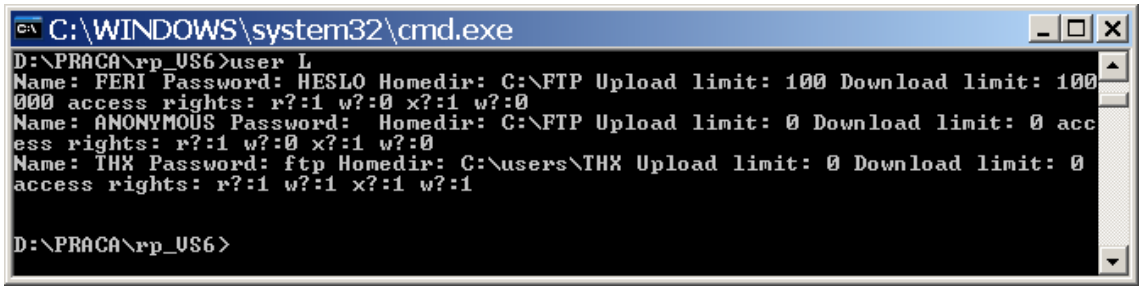
`PW`	- pre zmenu hesla, program nasledovne vyzve užívateľa k zadaniu hesla
`IP`	- pre zmenu IP adresy servra, program vyzve užívateľa k zadaniu IP
`PORT`	- pre zmenu administrátorského portu, program vyzve užívateľa k zadaniu portu
`Q`	- pre ukončenie programu (pre koniec práce s ADM príkazmi možno použiť príkaz BYE)
`?`	- pre vypísanie helpu (obsahuje popis lokálnych ale aj ADM príkazov)
`CONNECT`	- pre pripojenie na server



Obrázok 11 – ADM (konzolový) – práca s programom

6.4 USER (príkazový riadok)

Program USER, ktorý je tiež súčasťou tejto práce, umožnuje zmenu nastavení servra v dobe, keď FTPk nebeží. Funguje iba lokálne, ale nevyžaduje žiadné spojenie so servrom. Naopak ho ale nemožno použiť počas behu servra, pretože by mohlo dôjsť k poškodeniu súborov s nastaveniami. Program v sebe obsahuje implementovaný ADM príkaz USER, spúšťa sa z príkazovej riadky a ako parameter dostane priamo parameter ADM príkazu user. Keďže sa názov programu zhoduje s názvom príkazu, možno povedať, že takto sa dá s ADM protokolom pracovať priamo z príkazovej riadky. Po spustení program vykoná príkaz a vypíše príslušné odpovede, ako keby odpovedal server.



Obrázok 12 – USER – práca z príkazového riadku

7 Otvorenosť a rozšíriteľnosť servra

Jednou z lepších vlastností nášho FTPk je jeho modifikovateľnosť a rozšíriteľnosť. Presnému špecifikovanie ADM protokolu umožňuje vznik ďalších administrátorských programov. Podpora pluginov zabezpečuje možnosť pridávania a modifikácie ďalších funkcií k programu. Samotný program sme navyše navrhli tak, že nové funkcie možno pomerne jednoducho doprogramovať priamo do tela programu. V tejto kapitole sa rozoberajú jednotlivé možnosti ďalšieho vývoja, ako aj popisujú už implementované možnosti rozširovania funkcií.

7.1 Využitie cudzích programov pre administráciu

Aby mohol ľubovoľný program komunikovať so servrom cez ADM protokol, stačí sa pripojiť na administrátorský port. Druhou možnosťou je pripojenie sa na FTP port a používanie ADM protokolu cez príkaz SITE. V oboch prípadoch postačia programy telnetovského typu, ako napr. PuTTY [P4] a pod. Jediné, čo sa od programu vyžaduje, je schopnosť posielať používateľom napísaného textu a vypisovania odpovedí servra na užívateľove príkazy. Používateľ síce prichádza o akékoľvek výhody asistencie práce s ADM protokolom, ktoré poskytujú administrátorské programy, v prípade núdze ale takéto riešenie poskytuje možnosť zmeny nastavení servra. Navyše takéto programy na komunikáciu možno nájsť takmer všade, rozhodne sú rozšírenejšie ako samotné administrátorské programy.

7.2 Pluginy

FTPk podporuje pluginy pre implementovanie menej používaných funkcií. Tieto pluginy ale možno využiť nie len na tie funkcie, na ktoré boli pôvodne plánované (t.j. EBCDIC typ a compressed mode). Po zadaní mode EBCDIC nemusí nutne plugin prekladať súbory do znakovej sady EBCDIC, ale namiesto toho ich môže napríklad kryptovať. Na druhej strane bude ale musieť byť prispôsobený FTP klient, ktorý bude vedieť, že mode EBCDIC v skutočnosti znamená enkrypciu a bude mať v sebe implementovaný aj algoritmus na dekryptovanie. Toto je jeden zo spôsobov ako zlepšiť bezpečnosť prenosu. Keďže enkrypciu si musí užívateľ do pluginu naprogramovať sám, predpokladá sa, že to zvládne tiež aj pre svojho FTP klienta. Takáto kombinácia upraveného servra a klienta poskytuje navyše funkcionalitu, ktorá nebola do servra naprogramovaná. Užívateľ si takto k hotovému servru, bez toho aby menil jeho zdrojový kód a znovu ho kompiloval, môže prirábať vlastné ľubovoľné funkcie.

7.3 Možnosti ďalšieho vývoja servra

Medzi funkcie, ktoré by bolo vhodné implementovať do ďalších verzií, patrí kompresia log súborov, podpora MD5 alebo SHA1 hashovania hesiel a SSH pre komunikáciu cez FTP control connection a tak splnenie secure FTP štandardu. Tieto funkcie by sa dali ľahko pridať pomocou už hotových externých knižníc. Server ich momentálne neobsahuje preto, lebo cieľom tejto práce nebolo implementovať

komprimačné a bezpečnostné algoritmy, ale skôr napísať server tak, aby bol na takúto zmenu pripravený, čo sa aj podarilo.

7.4 Možnosti ďalšieho vývoja ADM protokolu

Pre rozvoj ADM protokolu je potrebné, aby server FTPk mal v sebe naprogramovanú podporu týchto nových rozšírení protokolu. Spočiatku by malo postačovať pridávanie jednotlivých ADM príkazov, predovšetkým pre monitoring aktuálneho stavu, odpájanie pripojených užívateľov a pod. S pridávaním ďalších funkcií a možností nastavovania servra by mali pribúdať aj ADM príkazy využívajúce tieto funkcie a umožňujúce meniť nové nastavenia. Ak by nebolo pridávanie ADM príkazov postačujúce pre využitie nových možností servra, je nutná zmena samotného protokolu a princípu jeho fungovania. Táto zmena ale bude znamenať, že staršie administrátorské programy, ktoré mali ako poslednú záchranu pre komunikáciu a využívanie nových ADM príkazov implementovanú konzolu, sa môžu stať nepoužiteľnými. Zatiaľ sa ale zdá, že momentálne princípy fungovania ADM protokolu poskytujú skutočne dostatočne široké možnosti pre pridávanie nových príkazov. Zmena princípov fungovania protokolu by bola robená skôr z iných dôvodov, ako napr. snahy zlepšiť strojové spracovanie, alebo prívetivosť pre užívateľov. Pre začiatok sa črtá ako dobrá nová funkcia, možnosť vypisovania helpu, ktorý bude obsahovať vždy aktuálny popis všetkých implementovaných príkazov. Ďalším užitočnou informáciou by mohlo byť zisťovanie verzie protokolu.

7.5 Možnosti ďalšieho vývoja ADM programov

Podpora skriptovania sa zdá ako silný nástroj pre prácu s ADM protokolom. Administrátorské programy, ktoré by umožňovali a uľahčovali tvorenie skriptov, by mohli užívateľom poskytnúť možnosť si k týmto správcovským programom v podstate dorábať vlastné funkcie. Graficky orientované programy by mali držať krok s vývojom ADM protokolu a pridávať grafické rozhrania pre všetky nové príkazy. Konzolovo orientované by mohli pridať istú asistenciu napríklad pri vypĺňaní zložitejších parametrov jednotlivých príkazov. Ako dobrá sa javí tiež možnosť zaznamenávania priebehu spojenia a teda následovnej spätnej kontroly.

8 Záver

Na záver možno konštatovať, že výber vhodného FTP servra by mal závisieť predovšetkým od očakávaného spôsobu využívania. Pre niekoho nie je dôležitý výkon a možnosť vzdialenej správy, ale požaduje čo najprehľadnejšie a najjednoduchšie nastavovanie. Niekto iný naopak požaduje rozsiahle možnosti správy a vzhľad grafického administrátorského programu ho až tak nezaujíma. Produkt tejto bakalárskej práce sa istým spôsobom snaží ísť oboma smermi. Súčasné splnenie tak rozdielnych požiadaviek však nie je jednoduché a treba robiť isté kompromisy. Preto sú výsledkom práce tri rôzne verzie administrátorských programov, pretože ich spojenie do jedného nebolo možné. Jeden je orientovaný skôr na prehľadnosť nastavení a užívateľskú prívetivosť, druhý beží v textovom režime a práca s ním sa podobá práci s príkazovým riadkom a tretí sa skutočne ovláda iba z príkazového riadku.

Server je však len jeden a preto je množina ním poskytovaných služieb a funkcií kompromisom. Na jednej strane podporuje takmer všetky príkazy protokolu. Na strane druhej by bolo žiadúce aby bol ADM protokol rozšírený a umožňoval napr. zisťovať viac informácií o aktuálnom dianí na servri. Kompletná implementácia všetkých možných želaných funkcií však presahuje rozsah tejto práce. Server samotný je koncipovaný tak, aby umožňoval ľahkú rozšíriteľnosť a pridávanie ďalších funkcií. Dôležitým prvkom je samotný ADM protokol, ktorý dáva užívateľovi voľnosť si nad týmto protokolom vytvárať vlastné programy. Tie môžu implementovať aj veľmi netriviálne a neočakávane funkcie, ktoré majú využitie iba pre jeho potreby a niekto iný by ich pravdepodobne nikdy nevyužil. Dá sa teda povedať, že v dostatočnej miere sa podarilo splniť požiadavku univerzálnosti programu.

Jedným z hlavných cieľov práce bola efektivita naprogramovaného servra. Čo sa týka náročnosti na výkon procesora, serveru postačí aj veľmi slabý počítač. V podstate na beh servra s rezervou stačí počítač, ktorý spĺňa minimálne požiadavky OS (server vyžaduje Windows 2000 alebo lepší). Pamäťová náročnosť by sa ale dala zlepšiť istou optimalizáciou využívania winapi. Ako príklad stačí to, že náš FTPk bez logovacieho okna zaberá približne o 1MB pamäte menej ako program s oknom. Dôležité je, že s narastajúcim počtom súčasne pripojených užívateľov sa zvyšuje množstvo spotrebovanej pamäte lineárne, pomerne pomalým tempom. Približne o 20 KB na každého pripojeného užívateľa. Limitom výkonnosti celého servra teda bude vo väčšine prípadov nie množstvo inštalovanej pamäte, alebo výkon procesora, ale rýchlosť diskov, prípadne rýchlosť spojenia medzi klientom a servrom. Tieto posledné dve výkon limitujúce oblasti určite server vyťaží úplne a nenechá žiadne rezervy (narozdiel od Serv-U [P1], ktorý občas nevyužije všetku dostupnú rýchlosť spojenia). Teda v prípade, že sú limitom rýchlosti prenosu výkon disku, alebo rýchlosť spojenia, použitie iného FTP servra veľmi nepomôže. Rozdiel vo výkone bude pravdepodobne v rozmedzí chyby merania rýchlosti.

Oblasťami, ktoré by si zaslúžili zlepšenie sú bezpečnosť a použitie kompresie na log súbory. Tieto funkcie možno implementovať pomerne rýchlo a ľahko použitím externých knižníc. Program ich neobsahuje, pretože by neboli dielom autora, a ak by aj, tak sú mimo oblasť zamerania práce.

Literatúra a použité zdroje informácii

[1] RFC 959 – FTP (http://www.faqs.org/rfcs/rfc959.html)
[2] RFC 854 – telnet (http://www.faqs.org/rfcs/rfc854.html)
[3] Run Length Encoding (RLE)
(http://www.data-compression.info/Algorithms/RLE/index.htm)
[4] Douglas E. Comer: Internetworking with TPC/IP,
Volume 1: Principles, Protocols, and Architecture, Prentice-Hall, 1991
[5] Introduction to Multithreading, Superthreading and Hyperthreading
(http://arstechnica.com/articles/paedia/cpu/hyperthreading.ars)
[6] Multiprocessing
(http://www.pcguide.com/ref/cpu/arch/extSMP-c.html)

Použité programy

[P1] Serv-u FTP server – RhinoSoft (http://www.serv-u.com/)
[P2] Gene 6 FTP server – Gene6 SARL (http://www.g6ftpserver.com/)
[P3] freeFTPd server – freeFTPd Team (http://freeftpd.com/)
[P4] PuTTY – Simon Tatham and PuTTY team
(www.chiark.greenend.org.uk/~sgtatham/putty)

Prílohy

A Obsah CD

V koreňovom adresári sa nachádza text tejto práce v elektronickej forme. Súbor BCprace.doc je vo formáte MS Word 2000, BCprace.rtf je v rich text formáte. Ďalej sa tu nachádzajú 4 adresáre s programami. V každom z nich sú dva podadresáre – source obsahuje zdrojové kódy programu a podadresár doc obsahuje programátorskú dokumentáciu. Užívateľská dokumentácia k jednotlivým programom je súčasťou textu tejto práce. V samotnom adresári programu sa nachádza spustiteľný program so všetkými súbormi, ktoré potrebuje pre svoj beh. Pre podrobnosti a informácie o inštalácii a používaní viz príslušné podkapitoly tejto práce. Všetky programy a dokumenty nachádzajúce sa na CD sme vytvorili v rámci tejto bakalárskej práce.

Následuje popis jednotlivých adresárov:

\ADM – obsahuje ADM (gui), pre bližšie informácie viz kapitola 6.2
\ADMk – obsahuje ADM (konzolový), pre bližšie informácie viz kapitola 6.3
\USER – obsahuje USER (príkazový riadok), pre bližšie informácie viz kapitola 6.4
\FTPk – obsahuje FTPk, pre bližšie informácie viz kapitola 4

Súbor zaloha.zip v koreňovom adresári obsahuje skomprimovaný celý obsah CD.

B Vývojová dokumentácia

Program FTPk vznikal vo viacerých fázach. Úplne na začiatku neumožňoval žiadne nastavenia, inicializácia takmer nebola potrebná. Ako prvé vzniklo hlavné vlákno programu, ktoré robí winapi registráciu, spúšťa mainTHREAD, ktorý vznikol ako druhý. Samotné hlavné vlákno ani mainTHREAD neobsahujú takmer žiadnu funkcionalitu. Možnosť vypisovania aktuálneho diania do okna bola jednou z prvých funkcií a pomáhala pri ladení programu. V tejto fáze sa robili nastavenia servra ručne, zmenou hodnôt premenných v zdrojovom kóde.

V ďalšej fáze došlo k vzniku protocol interpreter threadu. Predtým než bolo možné interpretovať FTP príkazy, bolo nutné do piTHREADU naprogramovať parsing prichádzajúcich dát na samotný FTP príkaz a jeho parameter. Ako prvé boli implementované jednoduché FTP príkazy, ktoré neumožňovali prenos súborov. Neskôr bol napísaný trTHREAD a bolo teda možné pridať FTP príkazy pracujúce s prenosom súborov do piTHREADU. V tejto dobe bolo dopracované čítanie parametrov servra z príkazového riadku. Lokálne bolo možné meniť nastavenie užívateľských kónt pomocou programu USER, ktorý vznikol hneď po dokončení trTHREADu.

Následovne mohol vznikať adminTHREAD. ADM gui vznikal spolu s vývinom tohto threadu, slúžil pre testovanie postupne implementovávaných funkcií. K týmto si hneď dorábal záložky pre uľahčené používanie. Úplne na záver bol zdrojový kód programu upravovaný do prehľadnejšej formy. Tam kde to malo zmysel, vznikli funkcie pre presných výpis chýb. Posledným vytvoreným programom bol konzolový ADM, ktorý slúži predovšetkým ako ukážka možností ADM protokolu. Pre podrobný popis fungovania FTPk servra viz príloha C.

C Detailný popis fungovania FTPk servra

Server je koncipovaný ako mnohovláknová aplikácia. Po spustení hlavné vlákno prevedie inicializáciu, načíta pluginy, vytvorí sockety pre administrátorské, FTP control connection a FTP dátové rozhrania. Inicializáciou sa myslí tiež samostatné spustenie a registrácia procesu, podľa nastavenia vytvorenie log okna. Ďalej sa inicializuje winsock2, obsadí sa ip adresa na danom porte pre FTP control connection. V prípade akýchkoľvek problémov v tejto fáze program končí, pretože ak nastane nejaká chyba pri inicializácií, zbytok programu by nemohol korektne pracovať. Napríklad ak je obsadený port či už pre FTP control connection, alebo ADM rozhrania, alebo sa nepodarilo iniciovať winsock2, program nemôže fungovať. V takomto prípade je ešte pred ukončením vypísaná chybová hláška oznamujúca konkrétnu chybu. Na konci inicializácie sa spustí vlákno mainTHREAD, ktoré počúva na porte FTP control connection. Program je písaný priamo vo winapi – dôvodom boli lepšie možnosti kontroly programu a vyšší výkon. Program v podstate skoro vôbec nepracuje s oknami a dôležitá je iba práca s winsockom a vláknami. Pre podrobnosti o práci socketov a komunikácií cez TCP/IP viz [4].

Hlavné vlákno potom prejde do winapi message loop a reaguje na prichádzajúce systémové podnety (ako napr. požiadavka na prekreslenie okna, alebo po zatvorení okna požiadavka na ukončenie programu).

mainTHREAD počúva na danom porte, v prípade pokusu o pripojenie skontroluje, či je všetko v poriadku – teda zistí ip adresu a port a ostatné dostupné informácie o druhej strane – tieto sa neskôr predajú piTHREADu. Na každého napojeného klienta teda beží jeden piTHREAD. V prípade, že počas pokusu o pripojenie nedošlo k žiadnej chybe (napr. nedostatok voľných socket descriptorov a pod.), dôjde nakoniec k spusteniu piTHREADu. Tento ako parameter dostane informácie o druhej strane a už pripojený socket, prostredníctvom ktorého môže okamžite komunikovať. V prípade, že by nastala nejaká chyba pri pripájaní sa, program vypíše chybovú hlášku a podľa druhu chyby buď pokračuje – napr. timeout, resp. nedostatok systémových zdrojov – po čase sa môžu uvoľniť a teda nie je dôvod končiť program. Alebo končí – v prípade vážnej chyby ako je napr. zlyhanie sieťového podsystému a pod.

piTHREAD – protocol interpreter thread – je vlákno, ktoré komunikuje s druhou stranou a interpretuje FTP protokol – spracováva príkazy, vykonáva potrebnú činnosť s nimi spojenú a posiela odpovede. PI thread predstavuje najväčšiu časť programu, po prijatí príkazu od klienta sa tento rozparsuje na príkaz a parametre. Ešte pred samotným rozparsovaním sa samozrejme kontroluje a reaguje na chyby pri komunikácii. Väčšinou sa iba vypíše chybová hláška. Ak je chyba závažnejšia – napr. odpojenie druhej strany, PI thread končí, pretože nemá odkiaľ dostávať príkazy. Kontrola správnosti parametrov FTP príkazu sa vykonáva pri každom príkaze – priamo v jeho procedúre. Po parsingu teda program prejde do „hlavného cyklu“ kde po nájdení odpovedajúceho príkazu vykoná procedúru jemu prislúchajúcu. Ak nenájde odpovedajúcu procedúru, pošle prijatý príkaz na spracovanie pluginom. Pri každom príkaze sa skontroluje správnosť parametrov, vykoná sa samotná príkazom požadovaná činnosť a pošle sa odpoveď o úspechu alebo neúspechu. V prípade príkazov spôsobujúcich prenos údajov sa spúšťa trTHREAD.

trTHREAD vlákno, ktoré sa pripojí/čaká na spojenie od druhej strany (podľa nastaveného aktívneho alebo pasívneho režimu) vykonáva prenos údajov. Samostatné vlákno sa spúšťa preto, pretože aj počas prenosu sa spracúvajú FTP príkazy – napr. aj

na prerušenie bežiaceho prenosu dát a pod. Teda piTHREAD musí spracovávať FTP príkazy a súčasne musí prebiehať prenos súborov. trTHREAD sa stará iba o prenos jedného daného súboru. Toto vlákno beží iba počas prenosu, po jeho skončení sa ukončí. Na každého pripojeného užívateľa teda pripadajú naraz najviac dva thready. V priebehu pripojenia môže jeden užívateľ vyvolať spustenie (a ukončenie) viacerých trTHREADov. Nanajvýš beží ale iba jeden naraz, pretože FTP nepodporuje súčasný prenos viacerých súborov. trTHREAD je sám o sebe pomerne jednoduchý – v úvode sa buď sám pripojí, alebo počká na pripojenie druhej strany a potom pokračuje v prenose súboru. Možné chyby vzniknuté pri pripájaní sú ošetrené príslušnými chybovými hláškami a podľa závažnosti chyby thread končí, alebo pokračuje. Po vykonaní prenosu sa pošle cez protokolové spojenie informácia o úspechu. V prípade napr. nemožnosti otvoriť súbor, dojdenia voľného miesta na disku a pod. sa odošle hláška o neúspechu. V trTHREADE sa počas samotného prenosu robí kontrola presiahnutia upload a download limitov.

Ďalším vláknom bežiacim počas behu servra je **adminTHREAD**, ktorý je interpretrom pre ADM protokol. Svojou štruktúrou a princípom fungovania sa veľmi podobá na piTHREAD. Toto je tiež dané podobným princípom fungovania ADM a FTP protokolov. FTPk podporuje naraz iba jedného pripojeného administrátora, aby nedošlo k poškodeniu súborov s nastaveniami.

Pri spúšťaní ľubovoľného threadu sa mu môže odovzdať iba jeden parameter – pointer. Preto je všetka komunikácia medzi threadmi realizovaná tak, že pri spustení dostane thread pointer na štruktúru s ďalšími údajmi. Väčšina potrebných údajov je teda pospájaná podľa významu do štruktúr, napr. štruktúra o pripojenom užívateľovi obsahuje jeho domovský adresár, limity na prenos dát a pod. Pointer na túto štruktúru je potom obsahom štruktúry navrhnutej pre spustenie trTHREADu. Tá zase obsahuje informácie potrebné pre samotný trTHREAD, aby sa mohol napojiť a vykonať prenos (napr. meno súboru, či ho ma prijať alebo poslať, ip adresa kam sa napojiť a pod.). Nakoniec dostane trTHREAD ako parameter pri spustení pointer na túto štruktúru.

Informácie o užívateľovi sa načítajú zo súboru user.dat pri logine – ktorý zvykne nasledovať hneď po pripojení sa na FTP control connection. Niektoré FTP príkazy (ako napr. HELP) sú však dostupné aj bez prihlásenia. Pred vykonaním každého príkazu sa teda kontroluje či je užívateľ oprávnený žiadať jeho vykonanie – napr. či je už prihlásený a potom či má dané prístupové práva. Toto sa väčšinou týka príkazov na prácu so súbormi a adresármi.

Používanie vlákien, a predovšetkým pridelenie vlákna každému užívateľovi, umožňuje plnohodnotné využitie viacprocesorových systémov, kde možno vlákna veľmi jednoducho rozdeliť medzi jednotlivé procesory. FTPk dokáže v prípade potreby využiť všetky procesory. Nie je obmedzený nejakým počtom, od ktorého by ďalšie procesory využiť nevedel. Toto sa javí v dnešnej dobe multijadrových procesorov a viacprocesorových systémov ako veľmi dobrá vlastnosť. Pre ďalšie informácie o multiprocessingu viz [6].

Multithreadovosť celého programu súvisí priamo s FTP protokolom. Aby server mohol obsluhovať viacero užívateľov, potrebuje na každého užívateľa jedno PI vlákno, ktoré komunikuje s druhou stranou. Samozrejme, je možné aj riešenie bez vlákien. Ale toto riešenie by bolo pomerne zložité, navyše by malo isté problémy s charakterom komunikácie cez TCP. Problémom je totižto nespoľahlivý prenos, kde môže napr. spojenie spadnúť a pod. V takomto prípade vlákno čaká istý počet sekúnd (timeout) za odpoveďou z druhej strany, pričom sa samo blokuje. Čo v prípade piTHREADU nevadí, pretože bez príkazov druhej strany nemá čo vykonávať a neblokuje pritom ostatné vlákna. Navyše v tomto zablokovaní ani nezaťažuje procesor. V prípade iba

jedného vlákna a postupného spracovania príkazov jednotlivých klientov, by zablokovanie znamenalo nutnosť čakania pre všetkých klientov.

Ak by nebol program multivláknový [5] a používal iba jedno vlákno, kde by simuloval súčasný beh interpretra pre viacero naraz pripojených užívateľov. Týchto by mal zaradených vo fronte a postupne vykonával FTP príkazy jednotlivých klientov. Potom by sa mohlo stať, že príkaz jedného klienta, ktorý vyžaduje dlhé spracovanie by znamenal pre ostatných klientov nutnosť vyčkať za skončením priebehu tohto príkazu. Hoci by títo ostatní klienti chceli vykonávať iba jednoduchšie príkazy, ktoré prebehnú rýchlo, museli by čakať za skončením spracovania dlhého príkazu cudzieho klienta. Program by tiež nevedel dostatočne dobre využívať viacprocesorové systémy a mal by problém v podstate využiť už len druhý procesor, nie ešte nejaké ďalšie. Takýto koncept teda nebol vhodný pre výkonný a efektívny server. K efektivite samozrejme patrí schopnosť využívať všetky dostupné systémové prostriedky, čo by bolo v prípade konceptu simulovanej súbežnosti interpretácie obtiažne.