Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Кафедра Компьютерных систем и программных технологий

Отчет по лабораторной работе

по дисциплине «Сети ЭВМ и телекоммуникации»

на тему: «FTP – сервер на языке Java»

Работу выполнил:

студент гр. 43501/3 Родина В.В.

Руководитель Вылегжанина К.Д.

«___» _____ 2016 г

Санкт-Петербург

2016

1. Постановка задачи

Реализовать FTP – сервер на языке программирования Java для ОС Windows.

Для тестирования программы использовать готовый FTP – клиент.

2. Ход работы

Итак, приведем немного теории, на основе которой написан сервер. На параллельно изучаемом курсе «Защита информации» была проделана лабораторная работы с использованием утилиты Wireshark, одним из заданий было изучение протокола FTP.

FTP (англ. File Transfer Protocol — протокол передачи файлов) — стандартный протокол, предназначенный для передачи файлов по TCP-сетям (например, Интернет). Использует 21-й порт. FTP часто используется для загрузки сетевых страниц и других документов с частного устройства разработки на открытые сервера хостинга.

Протокол построен на архитектуре «клиент-сервер» и использует разные сетевые соединения для передачи команд и данных между клиентом и сервером. Пользователи FTP могут пройти аутентификацию, передавая логин и пароль открытым текстом, или же, если это разрешено на сервере, они могут подключиться анонимно. Можно использовать протокол SSH для безопасной передачи, скрывающей (шифрующей) логин и пароль, а также шифрующей содержимое.

FTP может работать в активном или пассивном режиме, от выбора которого зависит способ установки соединения. В активном режиме клиент создаёт управляющее TCP-соединение с сервером и отправляет серверу свой IP-адрес и произвольный номер клиентского порта, после чего ждёт, пока сервер запустит TCP-соединение с этим адресом и номером порта. В случае, если клиент находится за брандмауэром и не может принять входящее TCP-соединение, может быть использован пассивный режим. В этом режиме клиент использует поток управления, чтобы послать серверу команду PASV, и затем получает от сервера его IP-адрес и номер порта, которые затем используются клиентом для открытия потока данных с произвольного клиентского порта к полученному адресу и порту.

Активный режим

Подключаемся к FTP-узлу CDC. Попробуем найти и загрузить файл справки Readme (рис.1).

```
C:\Users\v>ftp ftp.cdc.gov
Связь с ftp.cdc.gov.
220 Microsoft FTP Service
Пользователь (ftp.cdc.gov:(none)): anonymous
331 Anonymous access allowed, send identity (e-mail name) as password.
Пароль:
230 User logged in.
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection.
.change.dir
.message
pub
Readme
Siteinfo
up.htm
w3c
welcome.msg
226 Transfer complete.
ftp: 72 байт получено за 0,00 (сек) со скоростью 72,00 (КБ/сек).
ftp> get Readme
200 PORT command successful.
125 Data connection already open; Transfer starting.
226 Transfer complete.
ftp: 1428 байт получено за 0,14 (сек) со скоростью 9,92 (КБ/сек).
ftp> quit
221 Goodbye.
```

Рис.1. Подключение к серверу и отправка команд

Проанализируем полученные данные в Wireshark. Данные отфильтрованы по ip-адресу сервера и протоколам tcp и ftp. Первые три пакета отображают протокол транспортного уровня TCP, создающий надёжный сеанс связи – трехстороннее рукопожатие (рис.2). Протокол TCP регулярно используется во время сеанса связи для контроля доставки датаграмм, проверки их поступления и управления размером окна. Для каждого обмена данными между FTP-клиентом и FTP-сервером запускается новый сеанс TCP. По завершении передачи данных сеанс TCP закрывается. По завершении сеанса FTP протокол TCP выполняет плановое отключение и прекращение работы.

	Destination	Protocol	Length	Info
.1.35	198.246.117.106	TCP	66	59031 → 21 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=1 SACK_PERM=1
.117.106	192.168.1.35	TCP	66	21 → 59031 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1380 WS=256 SACK_PERM=1
.1.35	198.246.117.106	TCP	54	59031 → 21 [ACK] Seq=1 Ack=1 Win=8192 Len=0

Рис. 2. Создание сеанса связи.

Как только сеанс TCP установлен, появляется возможность для передачи FTP-трафика между компьютером и FTP-сервером. Когда FTP-сервер отправляет FTP-клиенту сообщение Response: 220, сеанс TCP на FTP-клиенте отправляет подтверждение сеансу TCP на сервере. Эту последовательность можно увидеть в приведенном ниже окне захвата данных программой Wireshark (рис. 3)

56 3.607300	198.246.117.106	192.168.1.35	FTP	81 Response: 220 Microsoft FTP Service
57 3.607341	192.168.1.35	198.246.117.106	TCP	54 59031 → 21 [ACK] Seq=1 Ack=28 Win=8165 Len=0

Рис.3. Подтверждение сеанса клиентом

Проходим процесс аутентификации (рис.4)

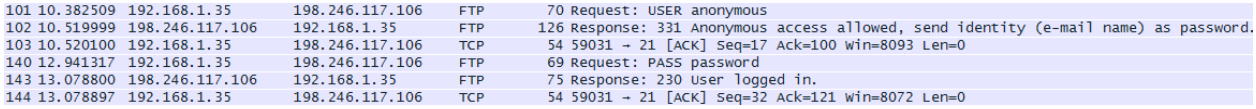

Рис.4. Аутентификация

Далее пытаемся получить какие-либо данные, например список файлов и папок.

Сначала передается команда порт с параметрами, где первые 4 числа – ip-адрес моего компьютера, последние два для вычисления номера порта: 230*256+155=59035.

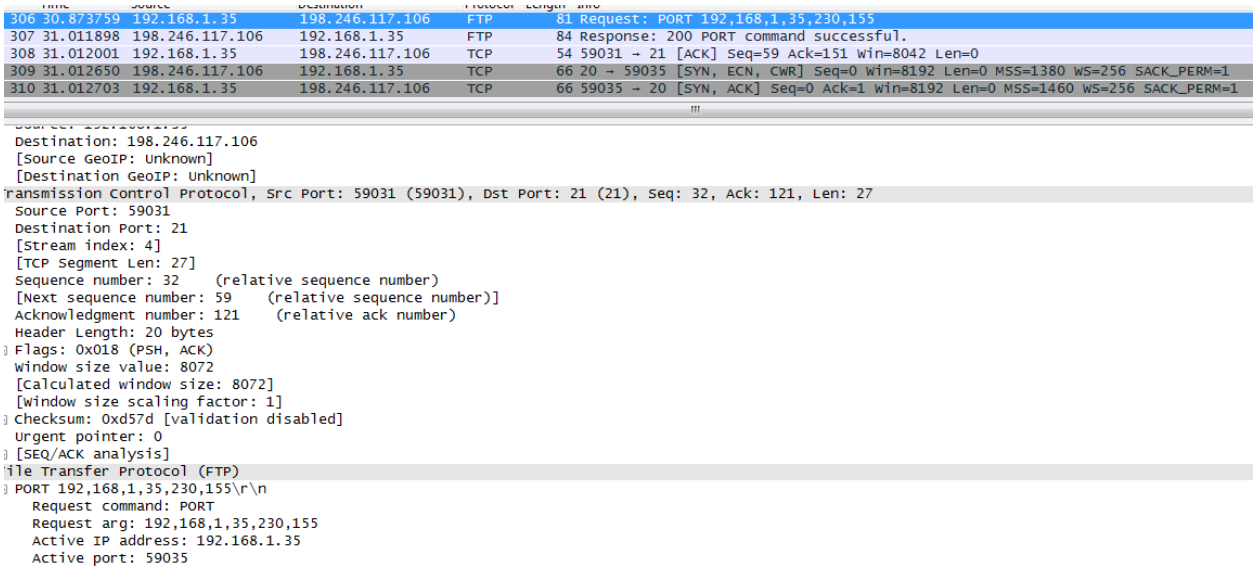

Рис.5. Команда порт и получение номера порта

Далее отправляется команда netlist (рис.6)

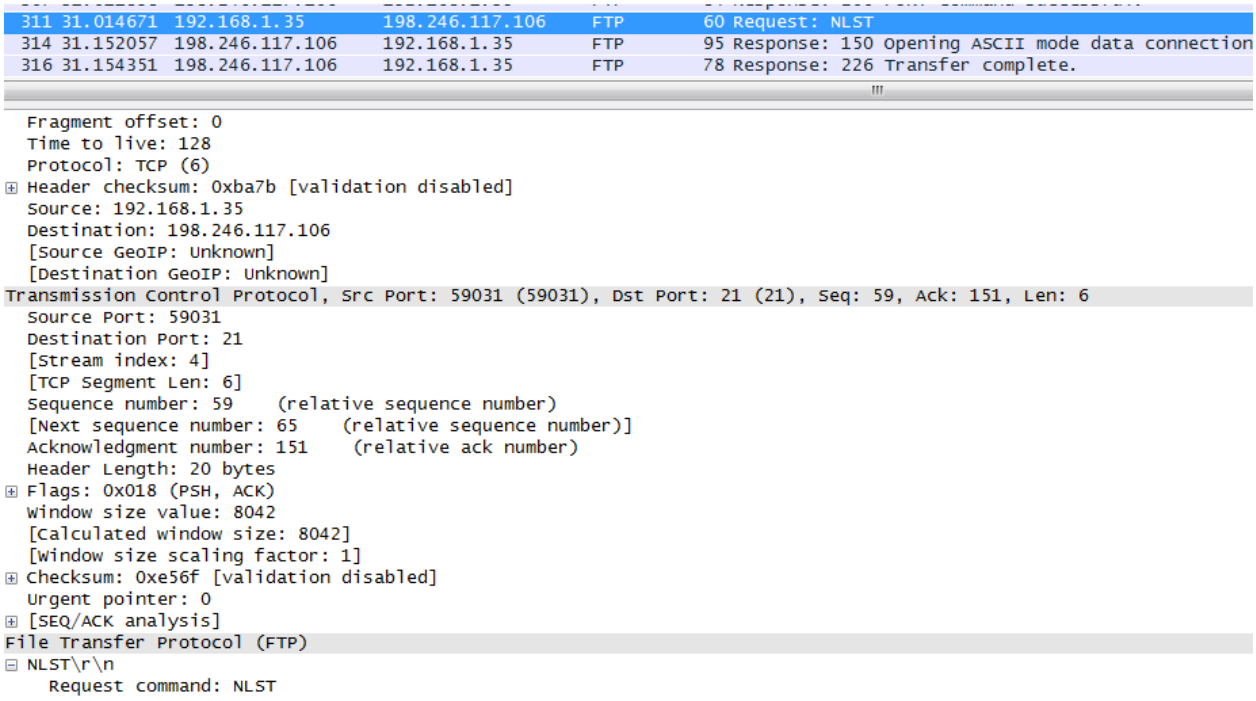

Рис.6. Отправка команды ls

Пакет номер 316 уведомляет о завершении передачи.

Далее отправляем команду скачивания файла Readme (рис.7). Теперь порт 59036, данные передаются по каналу с 20 порта ftp-сервера на 59036 порт клиента.

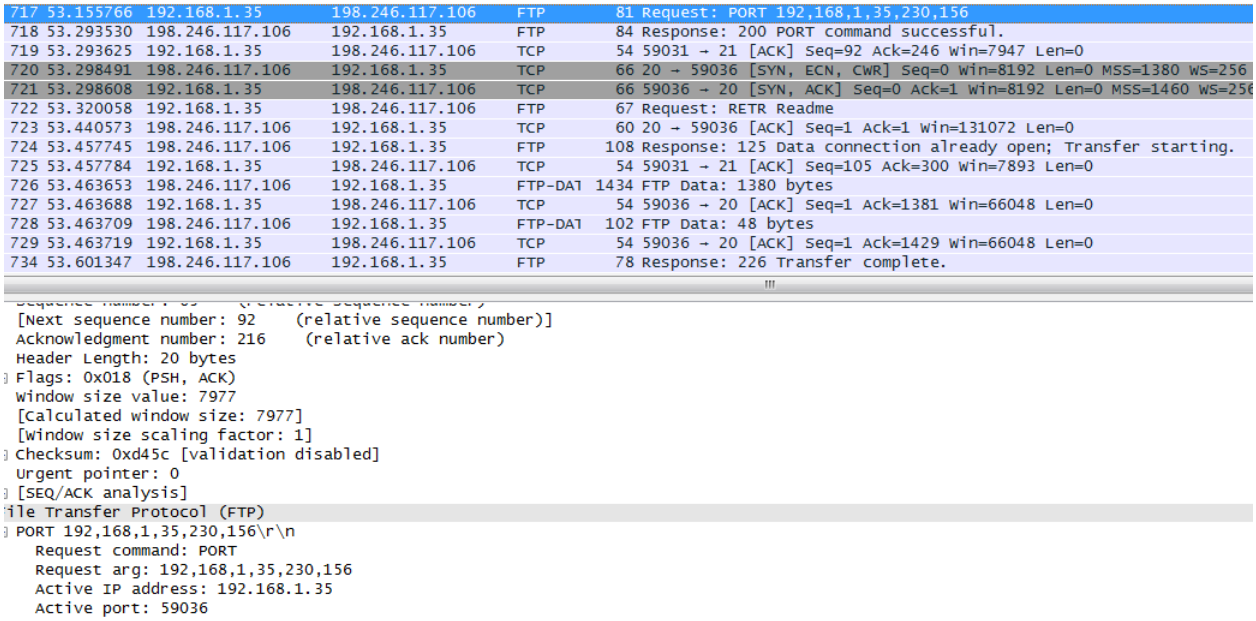

Рис.7. Скачивание файла

Завершаем сеанс – инициируем выход (рис.8, 9)

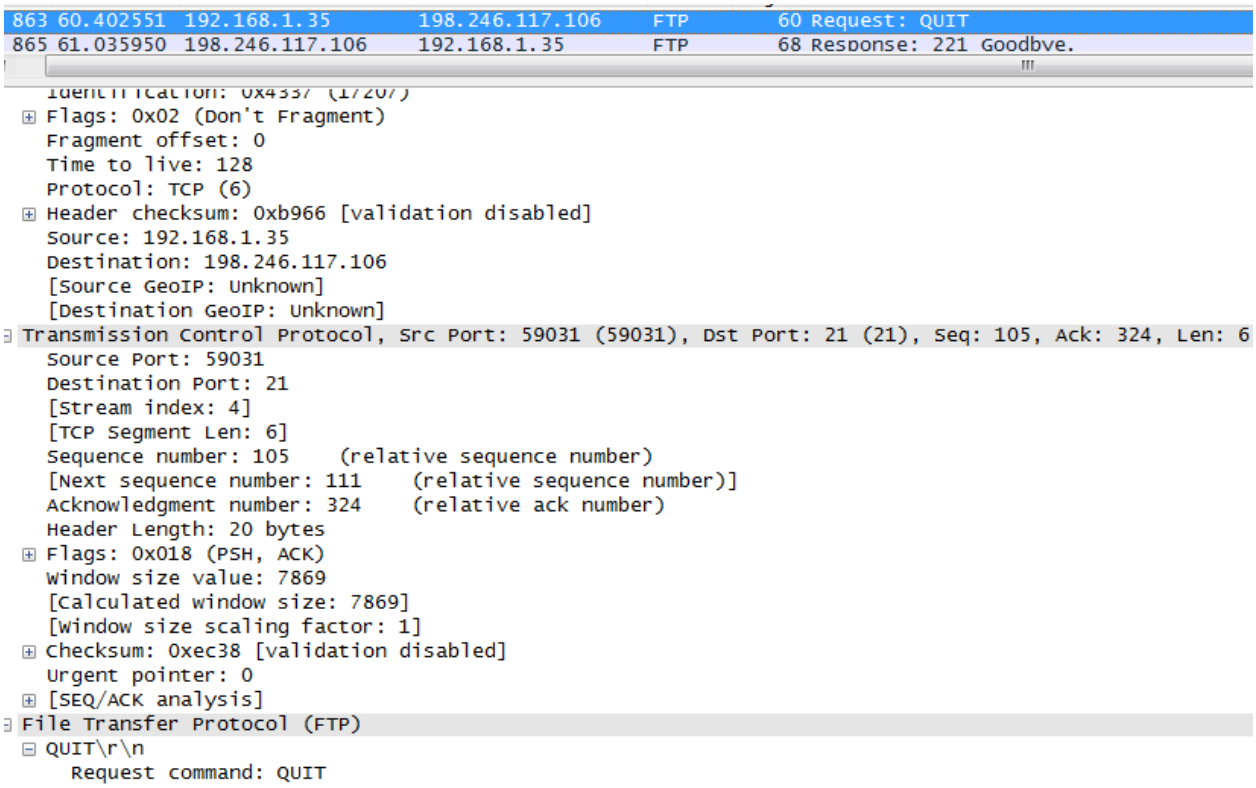

Рис.8. Завершение сеанса

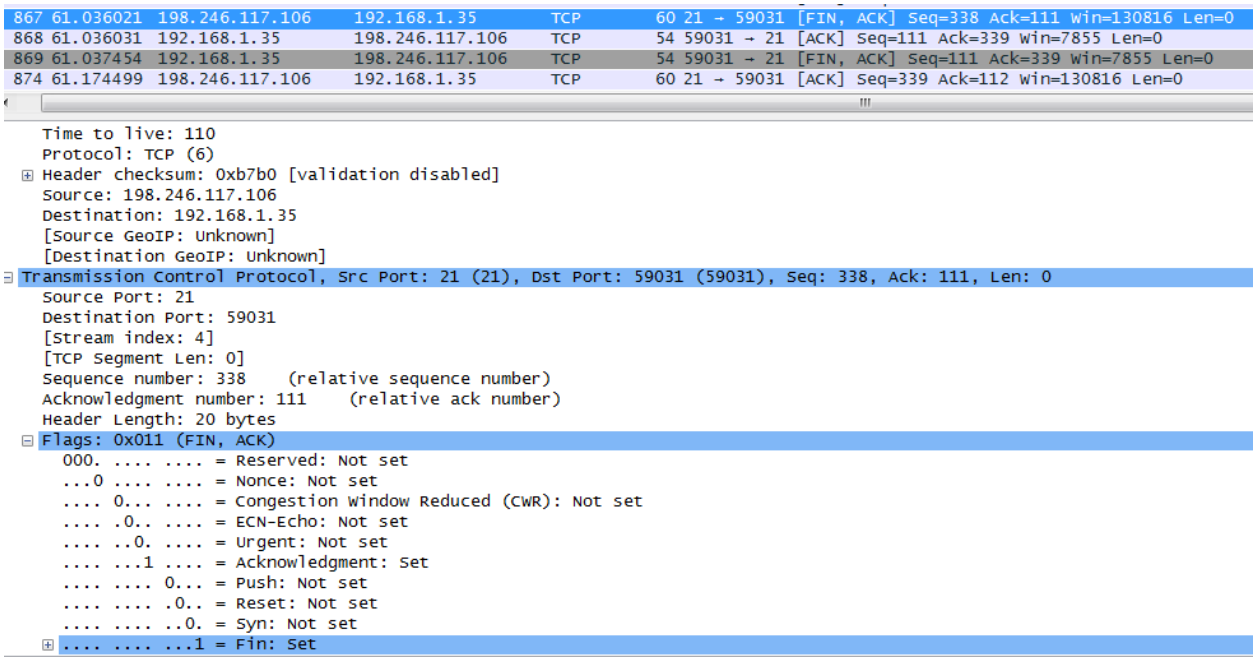

Рис. 9. Завершение сеанса

На рисунке 9: FTP-сервер отправляет пакет с флагом завершения FIN. Компьютер (клиент) отправляет ACK, чтобы подтвердить получение FIN для завершения сеанса связи между сервером и клиентом в пакете номер 868. В пакете 869 компьютер посылает FIN FTP-серверу, чтобы завершить сеанс TCP. FTP-сервер отправляет ответ, содержащий ACK в пакете 874, чтобы

подтвердить получение FIN от компьютера. После этого сеанс TCP между FTP-сервером и компьютером завершается.

На рис. 10 изображено завершение сеанса, где подписаны номера соответствующих пакетов.

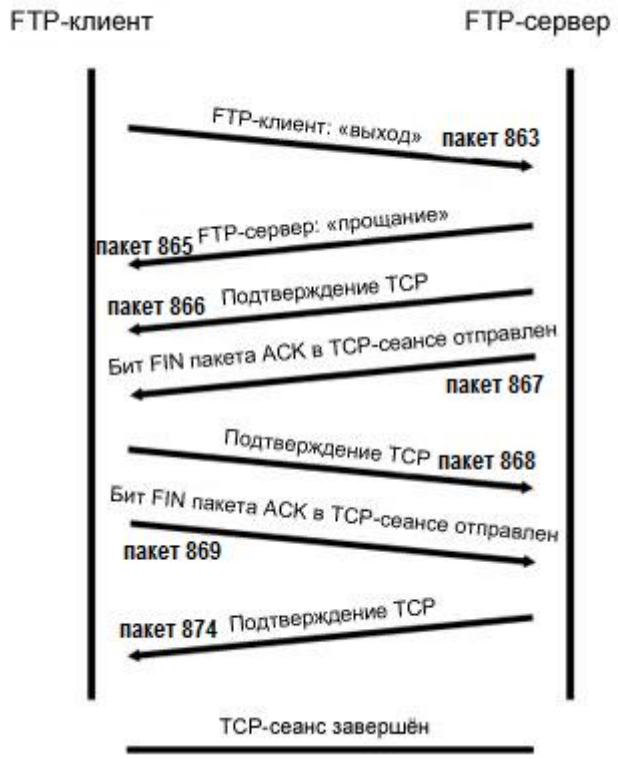

Рис. 10. Завершение сеанса

Пассивный режим

В пассивном режиме, после приветствия, сервер сообщает клиенту номер TCP-порта (из динамического диапазона 1024-65535), к которому можно подключится для установки соединения передачи данных. Порты в таком соединении как со стороны клиента, так и со стороны сервера оказываются произвольными.

Подключимся к ftp-серверу ftp.yandex.ru и рассмотрим переход в пассивный режим.

На рис.11 видим, что в пакете 326 отправляется команда PASV. Сервер отправляет ответ с параметрами, где первые 4 числа – его ip-адрес, а два последних – для вычисления порта, к которому будет подключаться клиент для установки соединения передачи данных: 203*256+79=52047

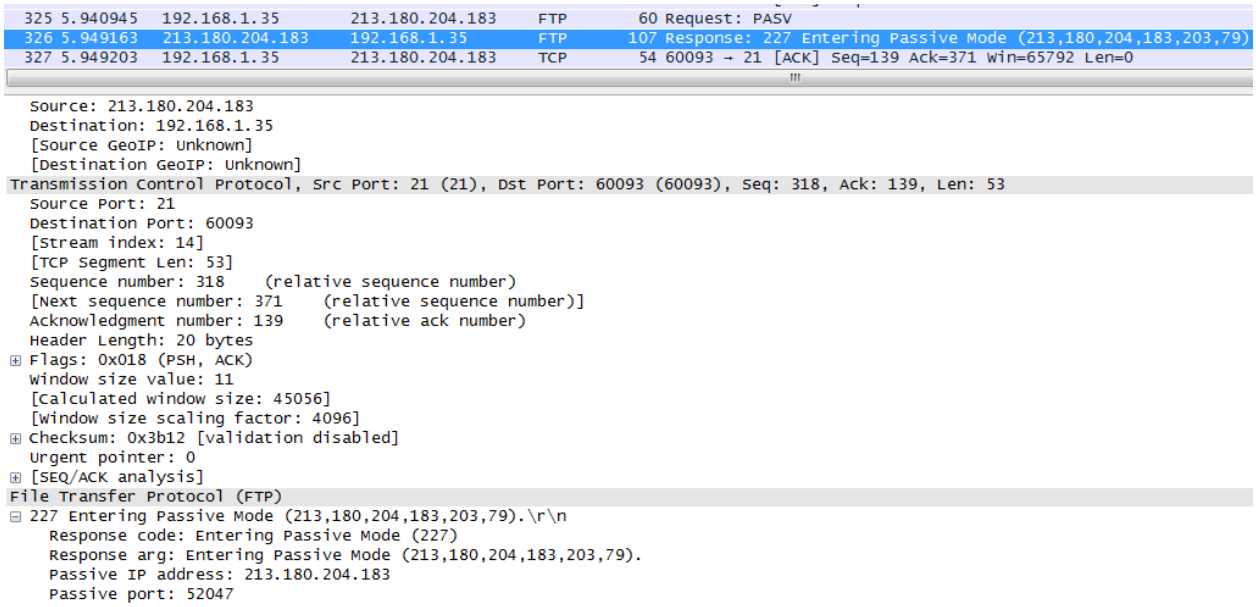

Рис.11. Переход в пассивный режим

На рисунке 12 видим, что в пакете 334 отправляются данные, порт-источник – это порт ftp-сервера 52047 (который был высчитан ранее) и порт-приемник – порт клиента 60094.

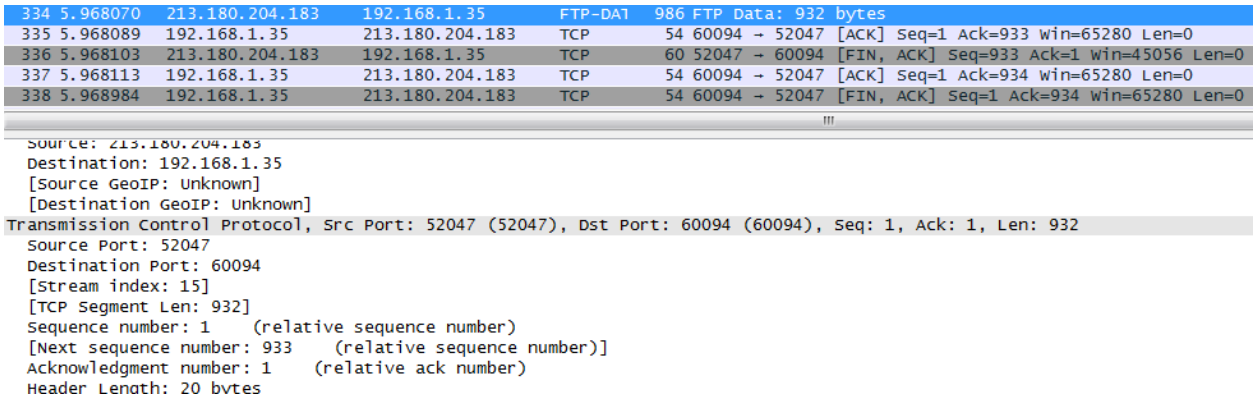

Рис.12. Передача данных в пассивном режиме

Далее происходит завершение сеанса: в пакете 336 установлен флаг FIN, он отправляется сервером, клиент отвечает пакетом (337) с флагом ACK. Далее клиент отправляет пакет (338) с флагом FIN, в ответ получает пакет (341) от сервера с установленным флагом ACK (рис. 13).

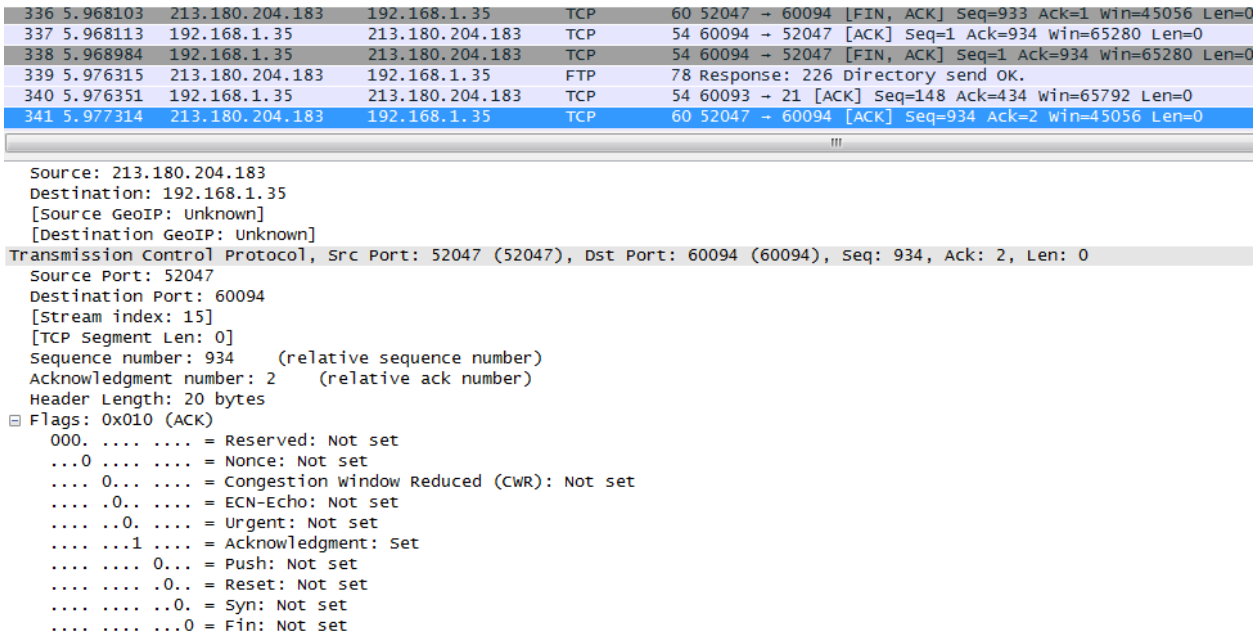

Рис. 13. Завершение сеанса

После выполнения данной работы понятен механизм передачи файлов между клиентом и сервером. Далее приведены результаты работы написанной программы. Программа написана в среде Netbeans, для тестирования в качестве клиента использована готовая программа – клиент FileZilla.

Итак, с помощью утилиты ipconfig в командной строке смотрим адрес нашего компьютера:

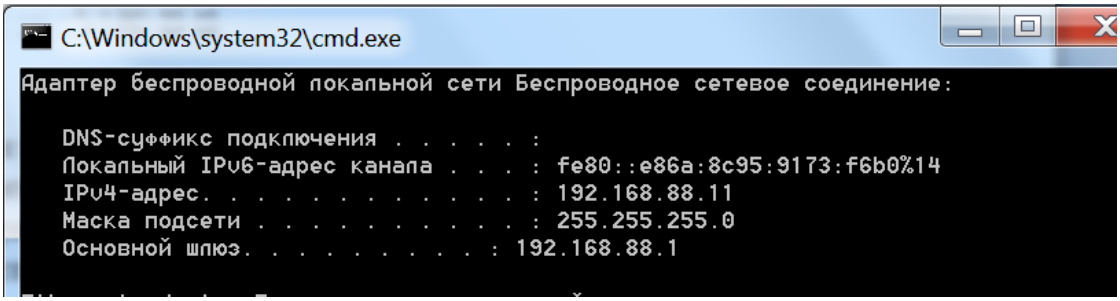

Рис. 14. IP-адрес компьютера

Вводим его в поле «Хост» в программе – клиенте.

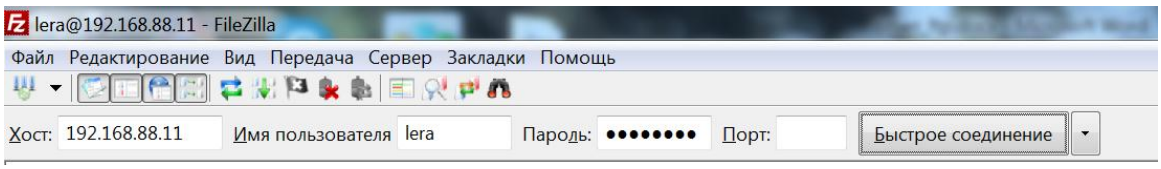

Рис. 15. Настройка параметров

Имя пользователя и пароль находятся в txt – файле, который расположен в папке проекта. Его можно менять.

Соединяемся с нашим сервером, нажав «Быстрое соединение»:

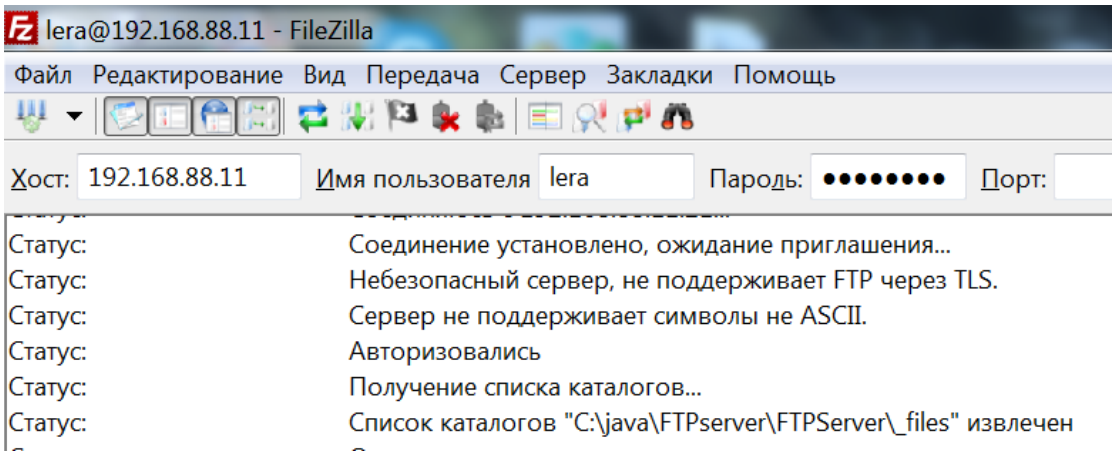

Рис. 16. Установка соединения и авторизация

Видим, что авторизация успешно завершена, далее получили список каталогов, хранящихся в папке _files, мы создали папку «downloads», в которую будут помещены скачанные файлы.

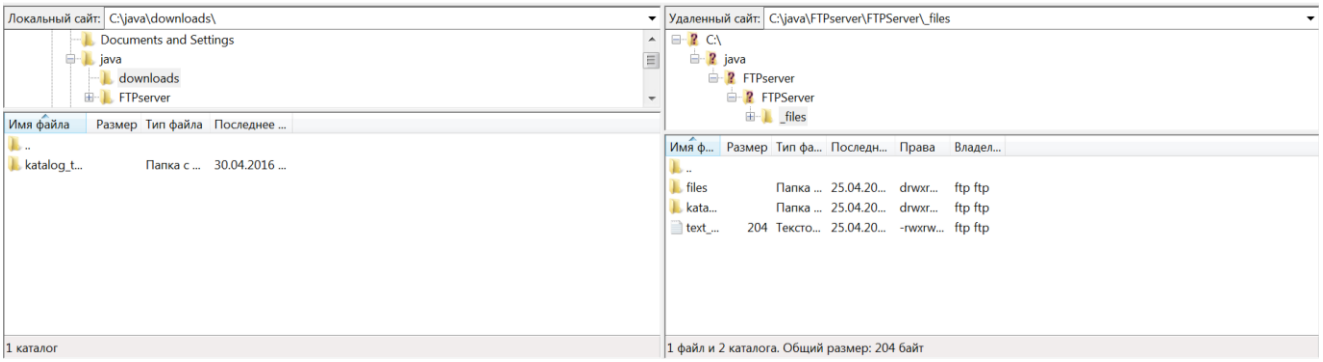

Рис. 17.

Слева скачанные файлы, справа то, что хранится на сервере.

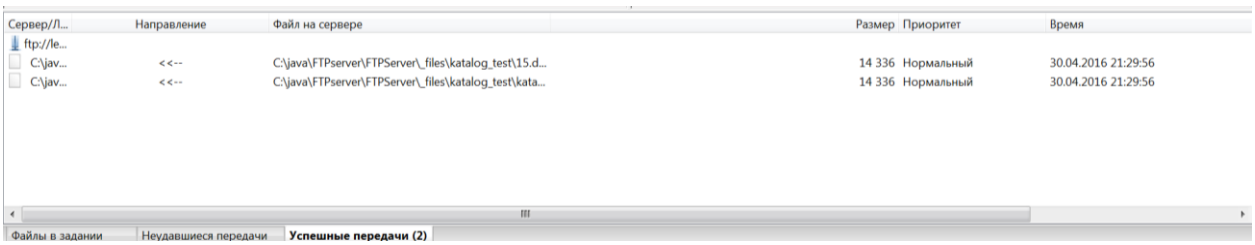

Рис. 18. Успешные передачи данных

3. Выводы

Таким образом, удалось реализовать простейший FTP – сервер на языке Java. На нем реализация была достаточно не сложной и не трудоемкой, в отличие от других языков программирования.

4. Листинги

SERVTHREAD.JAVA

```
package ftpserver;


import java.io.*;
import java.net.*;
import java.util.Stack;
import java.util.Enumeration;
import java.util.Iterator;
import java.util.LinkedList;


import java.util.StringTokenizer;


public class servThread implements Runnable
{
   protected String addr;
   protected static int nextPort = 55000;


   private LinkedList<String> USERS = null;
   private LinkedList<String> PASSWORDS = null;


   protected Socket clientSock = null;


   private BufferedReader reader;
```

```
    private PrintWriter writer;

    private dataThread DATA_THREAD;

    private char type = 'I';

    private String username;
    private boolean isAuth = false;

    private final String baseDirectory = System.getProperty("user.dir") + "\\_files";
    private String currentDir = baseDirectory;

    public servThread(Socket clientSocket, String addr, int cPort)
    {
        try
        {
            this.clientSock = clientSocket;
            this.reader = new BufferedReader(new
InputStreamReader(clientSock.getInputStream()));
            this.writer = new PrintWriter(new
OutputStreamWriter(clientSock.getOutputStream()), true);
            this.DATA_THREAD = new dataThread(this);

            this.addr = addr;

            BufferedReader BFreader = new BufferedReader(new
FileReader("passwords.txt"));
            USERS = new LinkedList<String> ();
            PASSWORDS = new LinkedList<String> ();
```

```
            String line;
            StringTokenizer st;
            while((line = BFreader.readLine()) != null)
            {
                st = new StringTokenizer(line);
                if(st.countTokens() != 2)
                    continue;
                this.USERS.add(st.nextToken());
                this.PASSWORDS.add(st.nextToken());
                st = null;
            }
        } catch (IOException e)
        {
            System.out.println("FTP server creation failed");
        }
    }

    @Override
    public void run()
    {
        System.out.println("Base directory is " + baseDirectory);

        reply(220, "Server ready");
        try
        {
            this.loop();
        } catch (Exception e)
        {
```

```
        System.out.println("Server control channel: commands loop failed");
        e.printStackTrace();
    } finally
    {
        try
        {
            this.clientSock.close();
            System.out.println("Client connection closed");
        } catch(IOException e)
        {
            e.printStackTrace();
        }
    }
}


private void loop() throws IOException
{
    String line, cmd;
    String fileFR = null;
    while((line = this.ReadAndPrintMsg("--> Client says : ")) != null)
    {
        StringTokenizer st = new StringTokenizer(line);
        cmd = st.nextToken();
        if (cmd.equalsIgnoreCase("user") && !this.isAuth)
        {
            int ans = this.setUser(st.nextToken());
            System.out.println("UserName: stringTokenizer = "+ans);
```

```
    if(ans==0)
    {
        reply(331, "Username successfully changed-need password");
    } else if (ans == 1)
    {
        reply(230, "Username successfully changed");
    }
} else if (cmd.equalsIgnoreCase("pass") && !this.isAuth)
{
    this.checkLogin();
    int result = this.checkPassword(st.nextToken().toString());

    if(result == 0)
        reply(220, "Password successfully taken");
    else if (result == 1)
        reply(531, "Incorrect password");
    else
        reply(532, "No such user");
} else if (cmd.equalsIgnoreCase("quit"))
{
    reply(221, "Goodbye!!!");
    break;
} else if (!this.isAuth)
{
    reply(530, "Please login with USER and PASS.");
    continue;
} else if (cmd.equalsIgnoreCase("rnfr"))
{
```

```
    checkLogin();
    fileFR = st.nextToken();
    reply(200, "RNFR accepted");
} else if (cmd.equalsIgnoreCase("rnto"))
{
    checkLogin();
    String fileTO = st.nextToken();
    File f = new File(this.currentDir + fileFR);
    if (!f.isDirectory()){
        f.renameTo(new File(this.currentDir + fileTO));
        reply(200, "RNTO accepted");
    } else {
        reply(550, "RNTO it is directory!!!");
    }

} else if (cmd.equalsIgnoreCase("type"))
{
    checkLogin();
    String typeStr = st.nextToken();
    if(typeStr.length() != 1)
    {
        reply(500, "Cant use this type");
    } else
    {
        reply(200, "Type accepted");
        this.type = typeStr.charAt(0);
        System.out.println("Type accepted: " + this.type);
    }
```

```
} else if (cmd.equalsIgnoreCase("syst"))
{
   reply(210, "Windows 7");
} else if (cmd.equalsIgnoreCase("noop"))
{
   reply(200, "noop accepted");
}else if (cmd.equalsIgnoreCase("pasv"))
{
   checkLogin();
   String addrStr = this.addr;
   st = new StringTokenizer(addrStr, ".");
   String h1 = st.nextToken();
   String h2 = st.nextToken();
   String h3 = st.nextToken();
   String h4 = st.nextToken();
   int port = getNextPort();
   int p1 = port/256;
   int p2 = port%256;
   this.DATA_THREAD.setDataPort(p1*256+p2);
   reply(227, "Entering Passive Mode (" +
         h1 + "," + h2 + "," + h3 + "," + h4 + "," + p1 + "," + p2 + ")");
   System.out.println("Entering Passive Mode");
   this.DATA_THREAD.openPort();
} else if (cmd.equalsIgnoreCase("cwd") || cmd.equalsIgnoreCase("xcwd"))
{
   checkLogin();
   String newDir = this.getFullName(st);
```

```
    System.out.println("CWD cmd dir: " + newDir);
    if(newDir.length()==0)
        newDir = this.baseDirectory;

    newDir = this.resolvePath(newDir);
    File f = new File(newDir);
    if(!f.exists()) {
        reply(550, "No such directory: " + newDir);
    } else if(!f.isDirectory()) {
        reply(550, "Not a directory: " + newDir);
    } else {
        currentDir = newDir;
        System.out.println("current dir: " + currentDir);
        reply(250, "cwd command successfull");
    }
} else if (cmd.equalsIgnoreCase("rmd") || cmd.equalsIgnoreCase("xrmd"))
{
    checkLogin();
    String dirName = this.getFullName(st);
    String path = this.resolvePath(dirName);

    File dir = new File(path);
    if(!dir.exists())
        reply(550, "Directory doesn't exist: " + dirName);
    else if (!dir.isDirectory())
        reply(550, dirName + " is not directory");
    else if (!dir.delete())
        reply(550, "Error deleting directory " + dirName);
```

```
    else{
        reply(250, "Directory successfuly removed: " + path);
        System.out.println("Directory deleted successfuly: " + path);
    }
} else if (cmd.equalsIgnoreCase("dele"))
{
    checkLogin();
    String fName = this.getFullName(st);
    String path = this.resolvePath(fName);

    File f = new File(path);

    if(!f.exists())
        reply(550, "File doesn't exist: " + path);
    else if (!f.delete())
        reply(550, "Error deleting file: " + path);
    else
    {
        reply(250, "File deleted successfuly: " + path);
        System.out.println("File deleted successfuly: " + path);
    }
} else if (cmd.equalsIgnoreCase("mkd") || cmd.equalsIgnoreCase("xmkd"))
{
    checkLogin();
    String dirName = this.getFullName(st);
    String path = this.resolvePath(dirName);

    File dir = new File(path);
```

```
    if(dir.exists())
        reply(550, "Directory already exist: " + dirName);
    else if(!dir.mkdir())
        reply(550, "Error creating directory "+ dirName);
    else{
        this.reply(257, "Directory created: " + path );
        System.out.println("Directory created: " + path);
    }

} else if (cmd.equalsIgnoreCase("cdup"))
{
    checkLogin();
    String newDir = this.resolvePath("..");
    System.out.println("resolvePath result: " + newDir);
    currentDir = newDir;
    reply(250, "cdup command successfull");
} else if (cmd.equalsIgnoreCase("retr"))
{
    checkLogin();
    String path = this.resolvePath(this.getFullName(st));
    this.DATA_THREAD.sendFile(path);
} else if (cmd.equalsIgnoreCase("stor"))
{
    checkLogin();

    String path = this.resolvePath(this.getFullName(st));
    this.DATA_THREAD.receiveFile(path);
} else if (cmd.equalsIgnoreCase("port"))
```

```
{
    checkLogin();
    String portStr = st.nextToken();
    st = new StringTokenizer(portStr, ",");
    String h1 = st.nextToken();
    String h2 = st.nextToken();
    String h3 = st.nextToken();
    String h4 = st.nextToken();
    int p1 = Integer.parseInt(st.nextToken());
    int p2 = Integer.parseInt(st.nextToken());

    String dataHost = h1 + "." + h2 + "." + h3 + "." + h4;
    int dataPort = (p1 << 8) | p2;

    this.DATA_THREAD.setDataPort(dataHost, dataPort);

    reply(200, "Port cmd succedeed");

} else if (cmd.equalsIgnoreCase("list") || cmd.equalsIgnoreCase("nlst") )
{
    checkLogin();
    String path = null;
    if(st.hasMoreTokens()){
        path = st.nextToken();
        if (path.charAt(0) == '-'){
            path = currentDir;
        }
    } else {
```

```
                path = currentDir;
            }

            System.out.println("Sending list in : " + path);
            //reply(200, "Port cmd succedeed");
            this.DATA_THREAD.sendList(path);
        } else if (cmd.equalsIgnoreCase("pwd") || cmd.equalsIgnoreCase("xpwd"))
        {
            this.checkLogin();
            reply(257, "\"" + this.currentDir + "\"" + " is current directory");
            System.out.println("pwd cmd anser : " + "\"" + this.currentDir + "\"" + "
is current directory");
        } else
        {
            System.out.println("cmd unknown: " + cmd);
            reply(500, "Command not supported: " + cmd);
        }
    }
  }
  String getFullName(StringTokenizer tok)
  {
    String elem=null, fullName=null;
    while(tok.hasMoreTokens())
    {
      elem = tok.nextToken().toString();
      if(fullName != null)
        fullName = fullName + " " + elem;
      else
```

```
            fullName = elem;
        }
        System.out.println("FullName function result: " + fullName);
        return fullName;
    }

    private int setUser(String user)
    {

        this.username = user;
        if(user.equalsIgnoreCase("anonymous")){
            this.isAuth = true;
            return 1;
        }
        System.out.println("username successfully changed to " + user);
        return 0;
    }
    private int checkPassword(String pass)
    {
        Iterator<String> it_u = USERS.iterator();
        Iterator<String> it_p = PASSWORDS.iterator();
        while(it_u.hasNext())
        {
            String p = it_p.next();
            if(this.username.equalsIgnoreCase(it_u.next()))
            {
                //checkPassword
                if(pass.equals(p))
```

```
            {
                System.out.println("password successfully changed");
                this.isAuth = true;
                return 0;
            }
            else
            {
                System.out.println("password wrong");
                return 1;
            }
        }
    }
    System.out.println("No such user: " + this.username);
    return 2;
}
private void checkLogin()
{
    if(this.username == null)
    {
        reply(400, "Please login first");
    }
}
void reply(int code, String msg)
{
    System.out.println("MSG to client <-- " + code + " " + msg);
    this.writer.println(code + " " + msg);
}
```

```
String ReadMsg()
{
   String ans = null;
   try
   {
      ans = this.reader.readLine();
   } catch (IOException e)
   {
      System.out.println("Error while reading msg from client");
   }

   return ans;
}
String ReadAndPrintMsg(String prefix)
{
   String ans = ReadMsg();
   System.out.println(prefix + " " + ans);
   return ans;
}
String resolvePath(String path)
{
   if (path.length() == 1){
      if (path.charAt(0) == '\\' || path.charAt(0) == '/'){
         path = this.currentDir.charAt(0) + ":\\";
      } else
         path = this.currentDir + "\\" + path;
   }else if (path.charAt(0) == '/'){
      path = this.baseDirectory;
```

```
    System.out.println("123: " + path );
}else if (path.charAt(1) != ':')
    path = this.currentDir + "\\" + path;

StringTokenizer pathSt = new StringTokenizer(path, "\\");
Stack segments = new Stack();
while(pathSt.hasMoreTokens())
{
    String segment = pathSt.nextToken();
    if(segment.equalsIgnoreCase(".."))
    {
        if(segments.size()!=1)
            segments.pop();
    } else if (segment.equalsIgnoreCase("."))
    {//Пропускаем
    } else
    {
        segments.push(segment);
    }
}
StringBuffer pathBuf = new StringBuffer();
Enumeration segmentsEn = segments.elements();
while (segmentsEn.hasMoreElements())
{
    pathBuf.append(segmentsEn.nextElement());
    if (segmentsEn.hasMoreElements())
        pathBuf.append("\\");
}
```

```
        return pathBuf.toString() + "\\";
    }

    public static int getNextPort(){
        if (nextPort != 65500){
            nextPort +=1;
        } else {
            nextPort = 55000;
        }
        return nextPort;
    }

}
```

MAIN.JAVA

```
package ftpserver;

import java.util.logging.Level;
import java.util.logging.Logger;
import java.io.IOException;

public class Main
{
    public static void main(String[] args)
    {
        FTPserver server;
```

```
if (args.length == 2){
    try{
        server = new FTPserver( args[0], Integer.parseInt(args[1]) );
    } catch ( NumberFormatException e ){
        System.out.println("Bad arguments!!!");
        server = new FTPserver();
    }
}
else{
    server = new FTPserver();
}

Thread ftpThread = new Thread(server);
ftpThread.start();

boolean cont = true;
while(cont)
{
    try
    {
        Thread.sleep(1000);
        try
        {
            if(System.in.read() == 'q')
            {
                System.out.println("Quit command catched");

                ftpThread.interrupt();
```

```
                    cont = false;
                }
            } catch (IOException e)
            {
                e.printStackTrace();
            }
        } catch (InterruptedException e)
        {
            Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, e);
        }
    }

    System.out.println("Server finished");
    System.exit(0);
  }

}
```

DATATHREAD.JAVA

```
package ftpserver;

import java.io.*;
import java.net.*;

import java.util.Date;
import java.util.Locale;
import java.text.SimpleDateFormat;
```

```
public class dataThread implements Runnable
{
   private servThread SERVER_THREAD;
   private int DATA_PORT = 20;
   private String DATA_HOST;
   private boolean isPasv = true;
   Socket dataSocket = null;
   ServerSocket servSocket = null;

   dataThread(servThread SRVTHREAD)
   {
      this.SERVER_THREAD = SRVTHREAD;
   }

   public void setDataPort( int port ){
      this.isPasv = true;
      this.DATA_PORT = port;
   }
   public void openPort() throws IOException{
       if (isPasv){
            servSocket = new ServerSocket(this.DATA_PORT);
            dataSocket = servSocket.accept();

            //if (dataSocket = null)
             //System.out.println("error");

          }else{
            dataSocket = new Socket(this.DATA_HOST, this.DATA_PORT);
```

```
        }
}

@Override
public void run()
{

}

public void setDataPort(String host, int port)
{
   this.DATA_PORT = port;
   this.DATA_HOST = host;
   this.isPasv = false;
   System.out.println("Connection settings for data transmittion:");
   System.out.println("\tHost = " + host + "\n\tPort = " + port);
}

public int sendList(String path)
{
   //dataSocket = null;
   //servSocket = null;
   try
   {
      File dir = new File(path);
      String filenames[] = dir.list();
      int filesCount = (filenames == null ? 0 : filenames.length);
```

```
        PrintWriter writer = new PrintWriter(new
OutputStreamWriter(dataSocket.getOutputStream()), true);


        this.SERVER_THREAD.reply(150, "Creating data connection successfull -
starting list transmittion");


        for(int i = 0; i < filesCount; i++)
        {
            String fName = filenames[i];
            File file = new File(dir, fName);
            this.listFile(file, writer);
            System.out.println("LIST processing: " + dir + "\\" + fName);
        }
        writer.flush();
        System.out.println("LIST [" + filesCount + " file(s)] transfered");
        this.SERVER_THREAD.reply(226, "Transfer complete");
    } catch (ConnectException e)
    {
        this.SERVER_THREAD.reply(425, "Can't open data connection. Pleasy
try again");
        return 1;
    }catch (Exception e)
    {
        e.printStackTrace();
        this.SERVER_THREAD.reply(550, "No such directory");
        return 1;
    } finally
    {
```

```
        try
        {
            if(dataSocket != null)
                dataSocket.close();
            if(servSocket != null)
                servSocket.close();
        } catch (IOException e)
        {}
    }
    return 0;
}
public void receiveFile(String path)
{
    FileOutputStream fos = null;
    //Socket dataSocket = null;
    //ServerSocket servSocket = null;
    try
    {

        /*if (isPasv){
            servSocket = new ServerSocket(this.DATA_PORT);
            dataSocket = servSocket.accept();
        }else{
            dataSocket = new Socket(this.DATA_HOST, this.DATA_PORT);
        }*/
        InputStream in = dataSocket.getInputStream();

        File f = new File(path);
```

```
    if(f.exists())
    {
        this.SERVER_THREAD.reply(550, "File already exist: " + path);
        return;
    }
    fos = new FileOutputStream(f);
    if(this.DATA_PORT == -1)
    {
        this.SERVER_THREAD.reply(500, "Send a PORT cmd first");
        return;
    }
    this.SERVER_THREAD.reply(150, "Starting to receive file " + path);
    //Здесь непосредственно прием файла
    byte buf[] = new byte[1024];
    int nread;
    while((nread = in.read(buf, 0, 1024)) > 0)
    {
        fos.write(buf, 0, nread);
    }
    in.close();

    this.SERVER_THREAD.reply(226, "Transfer completed successfuly");
} catch(ConnectException ce)
{
    this.SERVER_THREAD.reply(420, "Connection error");
    return;
} catch (FileNotFoundException e)
{
```

```
            this.SERVER_THREAD.reply(500, "File not exist");
            return;
        } catch (UnknownHostException e)
        {
            System.out.println("Host unknown");
            this.SERVER_THREAD.reply(500, "Host unknown");
            return;
        } catch (Exception e)
        {
            System.out.println("Unknown exception");
            this.SERVER_THREAD.reply(500, "exception unknown");
            return;
        } finally
        {
            try
            {
                if(fos != null)
                    fos.close();
                if(dataSocket != null)
                    dataSocket.close();
                if(servSocket != null)
                    servSocket.close();

            } catch(IOException e)
            {}
        }
    }
    public void sendFile(String path)
```

```
{
    FileInputStream fis = null;
    //Socket dataSocket = null;
    //ServerSocket servSocket = null;
    try
    {
        /*if (isPasv){
            servSocket = new ServerSocket(this.DATA_PORT);
            dataSocket = servSocket.accept();
        }else{
            dataSocket = new Socket(this.DATA_HOST, this.DATA_PORT);
        }*/
        OutputStream out = dataSocket.getOutputStream();

        File f = new File(path);
        if(!f.isFile())
        {
            this.SERVER_THREAD.reply(550, "Not a file");
            return;
        }
        fis = new FileInputStream(f);
        if(this.DATA_PORT == -1)
        {
            this.SERVER_THREAD.reply(500, "Send a PORT cmd first");
            return;
        }
        this.SERVER_THREAD.reply(150, "Starting to transfer file " + path);
        //Здесь непосредственно передача файла
```

```
    byte buf[] = new byte[1024];
    int nread;
    while((nread = fis.read(buf)) > 0)
    {
        out.write(buf, 0, nread);
    }
    fis.close();

    this.SERVER_THREAD.reply(226, "Transfer completed");
} catch (FileNotFoundException e)
{
    this.SERVER_THREAD.reply(500, "File not exist");
    return;
} catch (UnknownHostException e)
{
    System.out.println("Host unknown");
    this.SERVER_THREAD.reply(500, "Host unknown");
    return;
} catch (Exception e)
{
    System.out.println("Unknown exception");
    this.SERVER_THREAD.reply(500, "exception unknown");
    return;
} finally
{
    try
    {
        if(fis != null)
```

```
                fis.close();
            if(dataSocket != null)
                dataSocket.close();
            if(servSocket != null)
                servSocket.close();
        } catch(IOException e)
        {
            //ignore
        }
    }
}
private void listFile(File f, PrintWriter writer)
{
    Date date = new Date(f.lastModified());
    SimpleDateFormat dateFormat = new SimpleDateFormat("MMM dd hh:mm",
Locale.ENGLISH);
    String dateStr = dateFormat.format(date);

    long size = f.length();
    String sizeStr = Long.toString(size);
    int sizePadLength = Math.max(13 - sizeStr.length(), 0);
    String sizeField = pad(sizePadLength) + sizeStr;

    if (f.isDirectory())
        writer.print("d");
    else
        writer.print("-");
    writer.print("rwxrwxrwx   1 ftp      ftp ");
```

```
        writer.print(sizeField);
        writer.print(" ");
        writer.print(dateStr);
        writer.print(" ");
        writer.println(f.getName());
    }
    private static String pad(int length)
    {
        StringBuffer buf = new StringBuffer();
        for (int i =0; i < length; i++)
        {
            buf.append((char)' ');
        }
        return buf.toString();
    }
}
```

FTPSERVER.JAVA

```
package ftpserver;

import java.net.ServerSocket;
import java.net.Socket;
import java.io.IOException;

public class FTPserver implements Runnable
{
    protected String addr = "192.168.88.11";
    protected int SERVER_PORT = 21;
```

```
    protected ServerSocket servSock = null;

    public FTPserver(){

    }

    public FTPserver(String addr, int server_port){
        this.addr = addr;
        System.out.println(this.addr);
        this.SERVER_PORT = server_port;
    }

    @Override
    public void run()
    {
        openServSocket();
        while(true)
        {
            Socket clientSock = null;
            try
            {
                clientSock = this.servSock.accept();
            } catch (IOException e)
            {
                System.out.println("Catched exception while waiting for client
connections on server");
            }
            new Thread(new servThread(clientSock, this.addr,
this.SERVER_PORT)).start();
```

```
        }
    }


    private void openServSocket()
    {
        System.out.println("Opening server socket :\nSERVER PORT = " +
this.SERVER_PORT);
        try
        {
            this.servSock = new ServerSocket(this.SERVER_PORT);
        } catch (IOException e)
        {
            throw new RuntimeException("Error while opening port " +
this.SERVER_PORT, e);
        }
    }
}
```