МИНОБРНАУКИ РФ

ФГБОУ ВПО Тверской государственный технический университет

Кафедра “Программное обеспечение”

Курсовая работа

дисциплина «Операционные системы и сети»

Тема: «Протокол передачи файлов FTP»

Выполнил: студент группы

ПИН 17.06

Роман Иванов

Проверил:

Прохныч А.Н

Тверь 2019

Оглавление

[Введение 3](#_Toc28347799)

[Аналитическая часть 3](#_Toc28347800)

[Описание 3](#_Toc28347801)

[Краткая история 4](#_Toc28347802)

[Описание протокола 5](#_Toc28347803)

[Основные команды 5](#_Toc28347804)

[Коды ответов FTP 6](#_Toc28347805)

[Пример 7](#_Toc28347806)

[Техническое задание 8](#_Toc28347807)

[Проектная часть 8](#_Toc28347808)

[Архитектура приложения 8](#_Toc28347809)

[Реализация 9](#_Toc28347810)

[Заключение 11](#_Toc28347811)

# Введение

**Цель курсовой работы**: ознакомится со стеком протоколов TCP/IP. Понять принципы протокола передачи файлов FTP. Научится писать сетевые приложения.

**Задача**: Изучить спецификацию протокола FTP. Написать сетевое приложение для работы с файлами на сервере посредством данного протокола, используя программный интерфейс Socket. Предоставить отчет по данной программе.

# Аналитическая часть

## Описание

Протокол передачи данных — набор соглашений интерфейса логического уровня, которые определяют обмен данными между различными программами. Эти соглашения задают единообразный способ передачи сообщений и обработки ошибок при взаимодействии программного обеспечения разнесённой в пространстве аппаратуры, соединённой тем или иным интерфейсом.

Виды протоколов:

* **Стандартизированный** протокол передачи данных - позволяет разрабатывать интерфейсы (уже на физическом уровне), не привязанные к конкретной аппаратной платформе и производителю. Примеры: USB, Bluetooth.
* **Сигнальный** протокол - используется для управления соединением (установки, переадресации, разрыва связи). Примеры: RTSP, SIP
* **Сетевой протокол** — набор правил и действий (очерёдности действий), позволяющий осуществлять соединение и обмен данными между двумя и более включёнными в сеть устройствами.

**TCP/IP** — набор протоколов передачи данных, получивший название от двух принадлежащих ему протоколов: TCP (англ. Transmission Control Protocol) и IP (англ. Internet Protocol)

Наиболее известные протоколы, используемые в сети Интернет:

* **HTTP** (Hyper Text Transfer Protocol) — это протокол передачи гипертекста. Протокол HTTP используется при пересылке Web-страниц между компьютерами, подключенными к одной сети.
* **FTP** (File Transfer Protocol) — это протокол передачи файлов со специального файлового сервера на компьютер пользователя. FTP дает возможность абоненту обмениваться двоичными и текстовыми файлами с любым компьютером сети. Установив связь с удаленным компьютером, пользователь может скопировать файл с удаленного компьютера на свой или скопировать файл со своего компьютера на удаленный.
* **POP3** (Post Office Protocol) — это стандартный протокол почтового соединения. Серверы POP обрабатывают входящую почту, а протокол POP предназначен для обработки запросов на получение почты от клиентских почтовых программ.
* **SMTP** (Simple Mail Transfer Protocol) — протокол, который задает набор правил для передачи почты. Сервер SMTP возвращает либо подтверждение о приеме, либо сообщение об ошибке, либо запрашивает дополнительную информацию.
* **TELNET** — это протокол удаленного доступа. TELNET дает возможность абоненту работать на любой ЭВМ находящейся с ним в одной сети, как на своей собственной, то есть запускать программы, менять режим работы и так далее. На практике возможности ограничиваются тем уровнем доступа, который задан администратором удаленной машины.

В данной курсовой работе используется протокол **FTP.**

## Краткая история

Первая реализация протокола (1971 г.) предусматривала обмен между клиентом и сервером сообщениями, состоящими из заголовка (72 бит) и данных переменной длины. Заголовок сообщения включал в себя запрос к FTP-серверу или ответ от него, тип и длину передаваемых данных. В качестве данных передавались параметры запроса (например, путь и имя файла), информация от сервера (например, список файлов в каталоге) и сами файлы. Таким образом, команды и данные передавались по одному и тому же каналу.

В 1972 г. протокол был полностью изменён, и принял вид, близкий к современному. Команды с параметрами от клиента и ответы сервера передаются по TELNET-соединению (канал управления), для передачи данных создаётся отдельное соединение (канал данных).

В последующих редакциях была добавлена возможность работы в пассивном режиме, передачи файлов между FTP-серверами, введены команды получения информации, смены текущего каталога, создания и удаления каталогов, сохранения файлов под уникальным именем. Некоторое время существовали команды для передачи электронной почты через FTP, однако впоследствии они были исключены из протокола.

В 1980 г. FTP-протокол стал использовать TCP. Последняя редакция протокола была выпущена в 1985 г. В 1997 г. появилось дополнение к протоколу, позволяющее шифровать и подписывать информацию в канале управления и канале данных. В 1999 г. выпущено дополнение, посвящённое интернационализации протокола, которое рекомендует использовать кодировку UTF-8 для команд и ответов сервера и определяет новую команду LANG, устанавливающую язык ответов.

## Описание протокола

### Основные команды

* ABOR — Прервать передачу файла
* CDUP — Сменить каталог на вышестоящий.
* CWD — Сменить каталог.
* DELE — Удалить файл (DELE filename).
* EPSV — Войти в расширенный пассивный режим. Применяется вместо PASV.
* HELP — Выводит список команд, принимаемых сервером.
* LIST — Возвращает список файлов каталога. Список передаётся через соединение данных.
* MDTM — Возвращает время модификации файла.
* MKD — Создать каталог.
* NLST — Возвращает список файлов каталога в более кратком формате, чем LIST. Список передаётся через соединение данных.
* NOOP — Пустая операция.
* PASS — Пароль.
* PASV — Войти в пассивный режим. Сервер вернёт адрес и порт, к которому нужно подключиться, чтобы забрать данные. Передача начнётся при введении следующих команд: RETR, LIST и т. д.
* PORT — Войти в активный режим. Например PORT 12,34,45,56,78,89. В отличие от пассивного режима для передачи данных сервер сам подключается к клиенту.
* PWD — Возвращает текущий каталог.
* QUIT — Отключиться.
* REIN — Реинициализировать подключение.
* RETR — Скачать файл. Перед RETR должна быть команда PASV или PORT.
* RMD — Удалить каталог.
* RNFR и RNTO — Переименовать файл. RNFR — что переименовывать, RNTO — во что.
* SIZE — Возвращает размер файла.
* STOR — Закачать файл. Перед STOR должна быть команда PASV или PORT.
* SYST — Возвращает тип системы (UNIX, WIN, …).
* TYPE — Установить тип передачи файла (бинарный, текстовый).
* USER — Имя пользователя для входа на сервер.

### Коды ответов FTP

Ниже представлено краткое описание кодов ответа, которые могут быть возвращены FTP-сервером. Эти коды были стандартизированы IETF в RFC 959. Как было сказано ранее, код ответа — трёхзначное число. Первая цифра отвечает за один из трёх исходов: успех, отказ или указание на ошибку либо неполный ответ.

* 2xx — Успешный ответ
* 4xx/5xx — Команда не может быть выполнена
* 1xx/3xx — Ошибка или неполный ответ

Вторая цифра определяет тип ошибки:

* x0z — Синтаксическая.
* x1z — Информация. Соответствует информационному сообщению.
* x2z — Соединения. Сообщение относится к управляющему соединению либо к соединению данных.
* x3z — Соответствует сообщениям об аутентификации пользователя и его правах.
* x4z — Не определено.
* x5z — Файловая система. Соответствует сообщению о состоянии файловой системы.

Третья цифра окончательно специфицирует ошибку.

### Пример

В примере жирным шрифтом выделены команды, которые подаёт клиент; обычным шрифтом — сообщения сервера. Примечания отделены двумя слэшами:

220 FTP server ready.

220 Hello World!

**USER anonymous**

331 Anonymous login ok, send your complete email address as your password

**PASS \*\*\*\*\*\*\*\*\*\*\*\***

230 Logged in anonymously.

**PASV**

227 Entering Passive Mode (192,168,254,253,233,92) //Клиент должен открыть соединение на переданный IP

**LIST**

150 Here comes the directory listing. // Сервер передаёт список файлов в каталоге

226 Directory send OK.

**CWD incoming**

250 Directory successfully changed.

**PASV**

227 Entering Passive Mode (192,168,254,253,207,56)

**STOR example.avi**

150 Ok to send data. // Клиент передаёт содержимое файла

226 File receive OK.

**QUIT**

221 Goodbye.

### Техническое задание

Разработать десктопное кросплатформенное приложение, которое является FTP клиентом, позволяющим:

* Устанавливать соединение с FTP сервером через адрес хоста и ввод логина и пароля
* Выдавать список файлов, хранящихся на сервере
* Загружать файлы на сервер
* Скачивать файлы с сервера
* Удалять файлы с сервера
* Переименовывать файлы на сервере

# Проектная часть

## Архитектура приложения

Пакет **Application** – код графического интерфейса:

* ClientForm - запуск приложения
* StartController – управления действиями пользователя в окне соединения с сервером
* Controller – управления действиями пользователя в окне работы с файлами
* LogsController - правления действиями пользователя в окне вывода логов

Пакет **Logging** – хранение взаимодействия пользователя с сервером

* Logger – добавление и вывод логов

Пакет **Own FTP** – собственная библиотека для работы с FTP сервером

* CommandBuilder – создание команд для запросов к серверу
* MyFTPClient – реализация интерфейса FtpService, позволяющая взаимодействовать с сервером (соединение, скачивание и загрузка файлов)
* TransferTask – вспомогательный класс для скачивания и загрузки

## Реализация

Данная программа написана на языке Java (версии 8).

Для автоматизации сборки проектов используется фреймворк Maven.

Графический интерфейс написан с помощью библиотеки JavaFx.

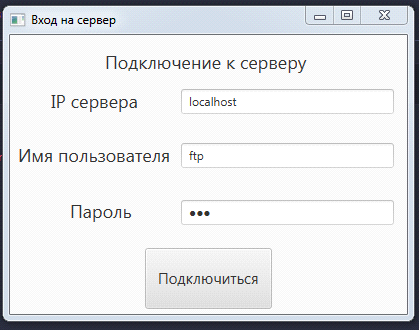
Метод для подключения к FTP серверу:

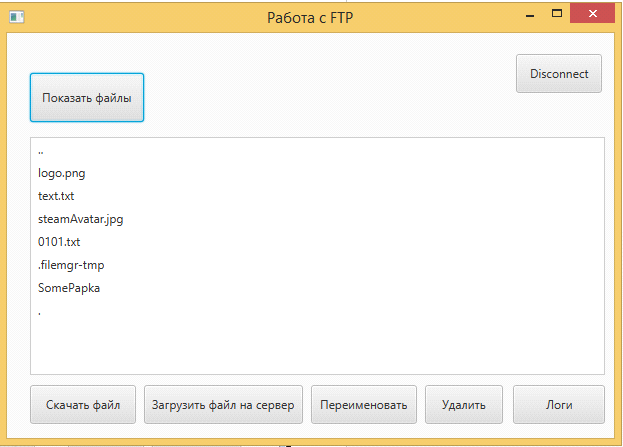
@Override  
public boolean connect(String hostAddress, String login, String password) throws IOException {  
  
 BufferedReader read = null;  
  
 try {  
  
 Socket socket = new Socket(hostAddress, PORT);  
  
 read = new BufferedReader(new InputStreamReader(socket.getInputStream()));  
 read.readLine();  
  
 commandBuilder.setInputStream(socket.getInputStream());  
 commandBuilder.setOutputStream(socket.getOutputStream());  
 if(commandBuilder.connect(login, password))  
 {  
 connect = true;  
 return true;  
 }  
 return false;  
  
 } catch (IOException e) {  
 e.printStackTrace();  
 return false;  
 } finally {  
 if (read != null)  
 {  
 try {  
 // read.close();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
 }  
}

Метод для загрузки файлов:

@Override  
public void downloadSingleFile(String remoteFilePath, String savePath) {  
 if(!commandBuilder.isConnected()) return;  
  
 String fileR = remoteFilePath;  
 if(fileR==null) return;  
 File fileL=new File(savePath);  
  
 try {  
 TransferTask trf = new TransferTask(  
 commandBuilder.download(fileR),  
 new FileOutputStream(fileL));  
  
 trf.startTransfer();  
  
 } catch (FileNotFoundException e) {  
 e.printStackTrace();  
 }  
}

Скриншоты программы:





# Заключение

В ходе работы над данной курсовой были освоены принципы протокола FTP и реализовано приложение для взаимодействия с файлами на сервере по данному протоколу.