FTP - протокол передачи файлов

Как только мы научились передавать данные между двумя узлами, можно приступить как реализации "полезных" программ необходимых для наших целей. Однако, писать свой протокол под каждое новое приложение неразумно. Поэтому консорциумом для довольно "частых" задач были придуманы стандартные протоколы. Протоколы, которые нужны для решения пользовательских задач, а не задач поддержания работы самой всемирной сети, называются протоколами уровня Приложений.

Изучение работы таких протоколов:

- даёт понимание работы самого протокола;
- расширяет кругозор понимания работы самой всемирной сети;
- развивает в голове архитектурное мышление.

Введение

Одним из самый старых протоколов прикладного уровня является протокол передачи файлов. Поскольку файл является одним из базовых понятий всей информатики в целом, его передача от одной машины к другой очень важна. Даже сейчас скачивания и загрузка файлов одна из осиновых задач, в любой области при использовании всемирной сети. Невозможно придумать более нужную задачу, чем передача файла, ну кроме передачи сообщения (почтовый протокол).

Основным протоколом для передачи файлов является FTP - File Transfer Protocol. Он описан в документах консорциума: https://tools.ietf.org/html/rfc959

Также есть очень краткое и ёмкое описание тут: https://eax.me/ftp-descr/

Данный протокол основывается на идеологии клиент-сервер, причём роли сервера и клиента жестко заданы. Здесь не допускается смена ролей приложений при изменении connect на accept и обратно. Т.е. приложение, которое принимает соединения (TCP-сервер), является и FTP-сервером. А клиент в смысле TCP соединения, также является и FTP клиентом. Поэтому в данной главе роль Сервера и Клиента чётко определена и мы о них будем говорить явно. Задачами этого протокола в рамках FTP клиента и сервера являются:

- скачивание клиентом файла из папки с сервера;
- закачивание клиентом файла в папку на сервер;
- просмотр и изменение дерева каталогов на сервере;
- проверка сервером правомерности действие и авторизация пользователя клиента.
- и некоторые другие.

Стандартным портом протокола является порт: **21**

Описание

При всей своей древности, протокол FTP использует достаточно интересный подход нескольких подключений. Одно из таких подключений называется управляющим каналом, в нём передаются команды от Клиента к Серверу, на которые сервер отвечает текстовой информацией. Остальные каналы служат для передачи данных, в частности файлов. Как правило, используется только два канала - один для управления, второй для скачивания и закачивания файлов. Но никто не запрещает открывать больше каналов и передавать множество файлов одновременно, в рамках одного управляющего соединения. Помимо этого на один Сервер может одновременно подключится несколько клиентов, и каждый их них может управлять и скачивать/закачивать данные. Протокол является сессионным - т.е. после TCP подключения, необходимо пройти авторизацию, только тогда FTP сервер будет обрабатывать команды Клиента, и откроет для него сессию. Далее в рамках открытой сессии Клиент можете отсылать команды и получать на них ответы от Сервера.

Чтобы скачать хотя бы один файл необходимо:

1. Открыть сессию (авторизоваться)
2. Перейти в нужный каталог
3. Отправить команду на скачивания файла
4. Подождать приход данных на канале данных
5. Завершить сессию.

Отметим, что в других протоколов скачивание файла может проходить по другому, например без открытия сессии, как в HTTP.

Общие понятия при работе с командами

Все команды, которые отсылает Клиент имеют признак конца - возврат каретки и переход на новую строку - CRLF : "\r\n". Синтаксис таков:

```
КОМАНДА ОПЦИИ \r\n
```

Отвечает сервер интересным образом - сначала идёт трёхзначный код состояния ответа, затем текст ответа и потом \r\n. Если ответ состоит из нескольких строк, то после трехбуквенного кода стоит дефис и последняя строка будет иметь пробел после кода. Таким образом, синтаксис ответа выглядит:

```
КОД Текст ответа \r\n
```

или

```
КОД-Текст ответа 1\r\n
КОД-Текст ответа 2\r\n
КОД-Текст ответа 3\r\n
КОД Текст ответа последний \r\n
```

Коды - это трехзначные числа делящиеся на группы:

- 2xx - успех
- 4xx и 5xx - команда не выполняема (нет прав и прочее)
- 1xx и 3xx - ошибка или ответ частичный

При подключении Сервер сразу же отсылает приветственное сообщение с кодом 220. После чего Сервер будет ожидать команд от клиента.

Команды и обработка

При анонимном подключении, чтобы открыть сессию достаточно отослать:

Команды 1.1:

```
> USER anonymous
< 220 Anonymous logged in
```

После чего сессия будет открыта, для её завершения, достаточно послать команду QUIT или просто закрыть управляющий сокет.

При не анонимном подключении сначала необходимо отравить логин, а потом отдельной командой пароль в незашифрованном виде:

Команды 1.2:

```
> USER mylogin
< 331 User mylogin OK. Password required
> PASS mymegapass
> 230 OK. Current directory is /
```

В этом случае мы заходим авторизированным и именным пользователем. Как видим в этом протоколе пароль передаётся в незащищённом режиме, что является недостатком данного протокола. Для обеспечения полной безопасности обычно включается режим SSL/TLS, но об этом позже.

Существуют несколько режимов кодирования передачи данных, два основных из низ это:

- **текстовый** - используется при передаче ASCII файлов
- **бинарный** - используется при передаче любых файлов. Самый предпочтительный способ. Надёжен как швейцарские часы.

Для включения бинарного режима передачи данных, необходимо отправить команду:

Команды 1.3:

```
> TYPE I
< 200 TYPE is now binary
```

Сама передача данных будет осуществлена по другому каналу - каналу данных, для его активации есть два режима: активный и пассивный. Поскольку активный, как правило, требует наличие белого АйПи адреса у Клиента, то использовать будет Пассивный режим.

Для завершения сессии достаточно отправить команду:

Команды 1.4:

```
> QUIT
< 221 Logout.
```

или же просто в одностороннем порядке зарыть управляющий сокет.

Пассивный режим передачи данных

Данный режим предусматривает, то что сервер откроет ещё один порт для подключения, к которому Клиент должен будет подключиться. Для открытия порта, Клиент посылает специальную команду, в ответе на которую сервер присылает информацию о том, куда надо подключиться для канала данных.

Команды 1.5:

```
> PASV
< 227 Entering Passive Mode (127,0,0,1,12,565)
```

В ответе Сервера в скобочках передаётся 6 байт, которые обозначают АйПи адрес и порт для канала данных. Первый 4 байта - это АйПи адрес, вторые два байта это порт. Как только информация получена, необходимо подключиться к этому каналу связи и ожидать по нему данных.

Скачивание файла с сервера

Для загрузке файла находящегося в текущем каталоге необходимо при работающем канале данных, отправить команду RETR.

Команды 1.6:

```
> RETR a.dat
< 150 Accepted data connection
```

После того, как файл будет передан Клиенту, Сервер уведомит управляющий канал о том, что передача закончена, кодом 226. Каким образом, можно определить, что в передающем канале нам был передан весь файл? Тут есть три варианта:

1. Ожидать прихода на управляющий канал сообщения с кодом 226. Однако, в этом случае данные по второму каналу ещё могли не успеть дойти. Поэтому данный способ не является надежным.
2. Можно заранее узнать размер файла, выполнив команду LIST. В этом случае придётся делать разбор ответа и вытаскивание размера файла. И к тому же между командой LIST и нашим фактическим скачиванием файл мог успеть измениться, и его размер может уже не соответствовать нашим ожиданиям.
3. После передачи файла пассивное соединение закрывается. Поэтому достаточно читать файл, пока он сокет на был закрыт.

Закачивание файла на сервер

Для загрузки файла в текущий каталог необходимо при работающем канале данных, отправить команду STOR.

Команды 1.7:

```
> STOR b.dat
< 150 Accepted data connection
```

Соответственно по каналу данных, надо передать содержимое файла и после окончания закрыть соединение. Когда сервер получит весь файл, он уведомить об этом в управляющем канале. Сервер положит файл с именем b.dat в текущий каталог.

Работа с директориями

При заходе на FTP сервер пользователь всегда находится в некотором каталоге, так же как и при работе с UNIX терминалом. Узнать текущий каталог можно с помощью команды:

Команды 1.8:

```
> PWD
< 257 "/" is your current location
```

Перейти в другую директорию можно с помощью команды аналогичной команде cd в UNIX терминале:

Команды 1.9:

```
> CWD folder
< 250 OK. Current directory is /folder
```

Указание путей при использовании этой команды можно как в абсолютном формате (полный путь от корня), так и в относительном (от текущей директории).

Отдельная команда подняться в родительский каталог:

Команды 1.10:

```
> CDUP
< 250 OK. Current directory is /
```

Если у Вас есть права на создание каталога, то его можно создать с помощью команды:

Команды 1.11:

```
> MDK newfolder
< 257 "/newfolder" - Directory successfully created
```

А вот посмотреть список файлов в текущем каталоге более хитрый. Поскольку список может быть достаточно большим, то он будет передаваться по каналу данных, а не в командном режиме. Поэтому тут надо действовать аналогично скачиванию файла: открыть канал передачи данных, отправить команду, получить данные по второму каналу, и получить подтверждение передачи на управляющем канале. Команда на список файлов:

Команды 1.12:

```
> LIST
< 150 Accepted data connection
```

У этой команды также существуют разнообразный набор ключей, как и в аналогичной UNIX команде ls. Ответ сервера будет выдан в UNIX формате, к примеру:

Транскрипт 1.1: список файлов в папке

```
drwxrwxr-x    2 1000       hydra            4096 Apr 10 04:05 06sajiP71TsHXf
drwxrwxr-x    2 1000       hydra            4096 Apr 10 04:05 0A5suk0HdMFGAa
drwxrwxr-x    2 1000       hydra            4096 Apr 10 04:05 0PlXqFU3ItlHFx
```

Разбор данного списка уже ложится на плечи клиента, который из него может вытащить всю необходимую информацию о файлах на удалённой машине.