**Инструкция по сборке**

В консоли из папки, где хранятся файлы совершить манипуляции:   
python tftp\_server.py  
python tftp\_client.py

**Инструкция по использованию**

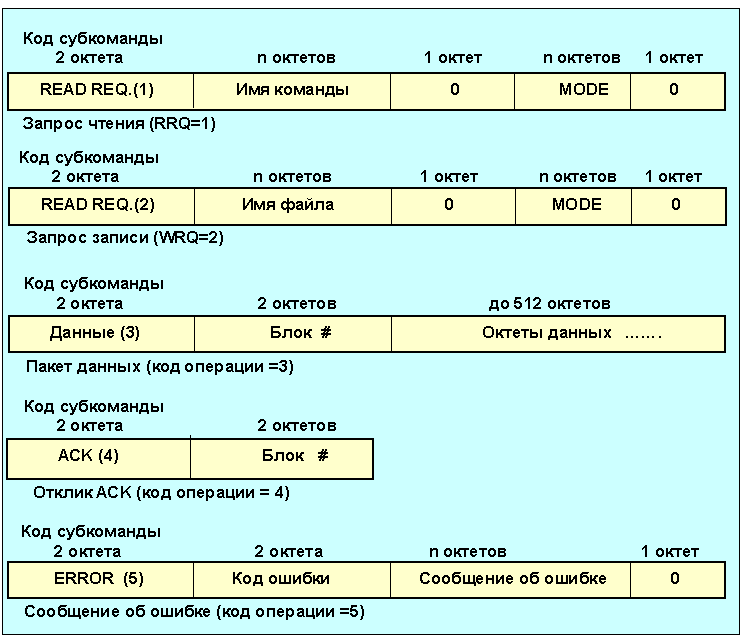
PUT (file.name) (file.name) позволяет отправить файл на сервер

GET (file.name) (file.name) позволяет получить файл с сервера

**Описание**

TFTP (англ. Trivial File Transfer Protocol — простой протокол передачи файлов) используется главным образом для первоначальной загрузки бездисковых рабочих станций. TFTP является упрощенной версией FTP. В отличие от FTP, не содержит возможностей аутентификации и основан на транспортном протоколе UDP (порт 69), а не TCP.

Сначала в TFTP-пакете идет поле размером в 2 байта, определяющее тип пакета:



После получения RRQ-пакета сервером, он сразу посылает в качестве подтверждения пакет с данными и с ID пакета, равным единице. В случае с WRQ-запросом — сервер должен прислать ACK-пакет c номером пакета 0.  
Каждый пакет данных содержит номер блока (block number), который затем используется в пакете подтверждения. Каждый пакет данных содержит 512 байт данных, за исключением последнего пакета, который содержит от 0 до 511 байт данных. Когда клиент получает пакет данных, который содержит меньше, чем 512 байт, он считает, что получил последний пакет.  
Так как TFTP использует ненадежный UDP, то именно от конкретной реализации TFTP зависит, как будут обработаны потерянные и дублированные пакеты. В случае потери пакета отправитель отрабатывает тайм-аут и осуществляет повторную передачу.

Класс сервера – TFTP\_Server, хранит информация о сокете, абсолютный путь директории сервера, а также словарь с клиентами. TFTP\_Server содержит функцию launch(), в которой в бесконечном цикле происходит чтение с сокета – данные и информация о клиенте. Далее, если это новый клиент, то добавляется запись в словарь клиентов с ключом – сокет клиента и значение – класс обработчик (ClientHandler). Затем вызывается основная функция обработчика клиента - request\_handle(), который извлекает из полученных данных код пакета (первые 2 байта). В зависимости от кода полученного пакета вызывается соответствующая функция обработчик - rrq\_handle(), wrq\_handle(), data\_handle(), ack\_handle(), error\_handle() или unexpected\_handle().

В любом из обработчиков сначала происходит извлечение соответствующих полей полученного пакета. В RRQ и WRQ пакетах содержится имя файла, которое с помощью функции os.path.join() из библиотеки os совмещается с директорией сервера, формируя итоговый абсолютный адрес файла. RRQ и WRQ обработчики также запускают в отдельном потоке функцию переотправки пакета, а обработчики DATA и ACK сбрасывают переотправку пакетов. На RRQ пакет сервер сразу формирует первый пакет данных для клиента и сразу проверяет критерий остановки передачи (данные меньше 512 байт). Обработчик WRQ отсылает клиенту посылку подтверждения записи файла. Обработчик DATA проверяет корректность номера принятой посылки, отправляет посылку подтверждения и проверяет критерий остановки передачи. Обработчик ACK формирует следующий отправляемый блок данных, проверяет критерий остановки передачи. Обработчик ERROR отображает информацию об ошибки и вызывает функцию завершения сервера.