Program 5 (FTP Client)

Reza Naeemi

FTP, or File Transfer Protocol, is a method of transmitting files from one computer to another over a TCP/IP connection. This FTP Client is written from scratch and is designed to mimic the functionality and interaction of the native Linux console FTP client. It operates by establishing a TCP connection between the client and a remote server over port 21 (or another user specified port). Commands and responses are handled on the full-duplex control socket connection as character streams and decoded to determine requested interactions. For ASCII or binary data streams, a second TCP socket connection must be established.

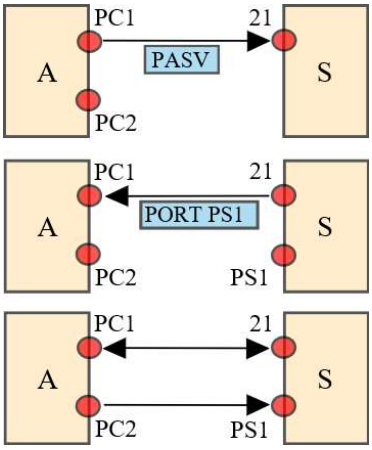

Figure 1: Illustration of starting a passive connection using port 21

Using commands such as LS, GET, or PUT require a second data channel to be established with the FTP server using "passive mode." For each issuance of commands requiring a data channel, the client requests passive mode which invokes a response from the server, returning a secondary hostname and port number for the client to connect to. The client creates a second full-duplex connection to this address and receives data from the server. When the transfer is complete, the data channel is terminated. In active mode, the client begins listening for incoming data connections from the server on a local port to transfer data. This FTP client supports passive mode only and does not support active mode.

The source code is broken down into two class files and one main command loop:

- ftp.cpp: usage: host-name [port]
  The program can be started using the following syntax:
  - ./ftp : Opens the client only.
  - ./ftp domain.tld : Opens the client and connects to domain.tld automatically.
  - ./ftp domain.tld XX : Opens the client and connects to domain.tld automatically on custom port XX, where XX is a positive number.

  The main loop is responsible for displaying the ftp> prompt and accepting user command inputs.

- FTPClient.cpp:
  The interpreter class is responsible for all client application logic. It accepts user input and maps a given command to instruction code for that command. If additional input is necessary, the tokenizer is used to analyze that input for the command interpreter to act on. Client interaction

is determined by inputs from the user and responses from the FTP server. Supported commands: Open, CD, PWD, SYST, GET, PUT, DEL, LS, Close, Quit

- Socket.cpp:
  The ftp class handles interactions with system I/O relating to disk and TCP socket usage, formatted as required by the interpreter class. Low-level logic can be found here.