Universidad de La Laguna

Sockets Programming

Redes y sistemas distribuidos

Javier Farrona Cabrera
alu0101541983@ull.edu.es

Index

Brief description of the developed application

This project's goal is to put in place a basic file transfer protocol server that enables file uploads and downloads from a single folder. As far as we are aware, a common TCP network protocol called File Transfer Protocol (FTP) is used to move files from a server to a client so that the client can save or retrieve the files from the server.

This protocol uses the data connection (port 20) and the control connection (port 21) to function. Commands and the responses to these commands are transferred across the control connection port. However, in this specific context, the term "command" refers to the textual instructions that the client will transmit to the server, as specified by the FTP protocol.

Two distinct file transfer modes exist:

- **Active mode:** The client initially establishes a connection to the server command port (port 21) from an arbitrary unprivileged port N, which must be 1024 or larger. Next, the client issues the FTP command **`PORT M`** to the server and begins listening on port M (M = N +1).
  The client's firewall is the primary issue with this option. This is because, when the server reconnects to the client, the firewall has the ability to prevent connections from unknown ports.
- **Passive mode:** The client supports passive mode in order to address the problem that occurs when using active mode. The client establishes both connections (control and data) in this mode.
  The client establishes two random ports in the client-server connection, the first of which connects to port 21. Next, a port number is returned by the server in response to a PASV instruction issued by the client. Next, in order to send the data, the client establishes a connection to that port.

A guide for the compilation of the source code and the necessary steps to execute the server program

In order to see how the program works, it is necessary to compile it as follows:

1. We start by executing the makefile make.
2. We compile the ftp_server ./ftp_server.

Test cases

Put in Passive Mode

```
---> SYST
215 UNIX Type: L8.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> passive
Passive mode on.
ftp> put prueba
local: prueba remote: prueba
---> TYPE I
200 OK
ftp: setsockopt (ignored): Permission denied
---> PASV
227 Entering Passive Mode (127,0,0,1,163,99).
---> STOR prueba
150 File status okay; about to open data connection.
125 Data connection already open; transfer starting.
8 bytes sent in 0.00 secs (15.7828 kB/s)
ftp>
```

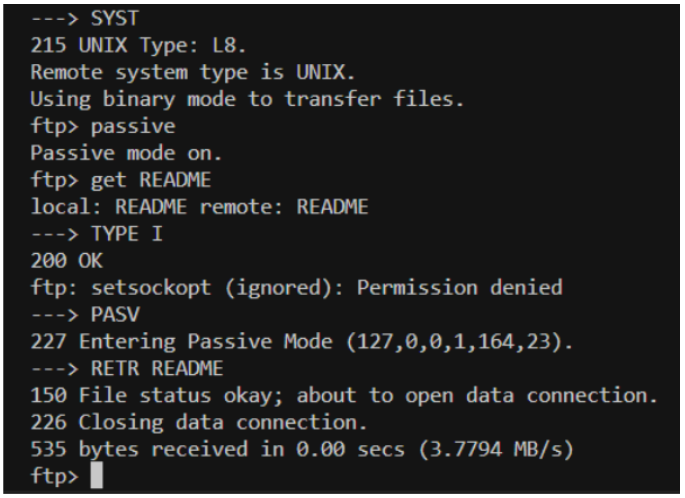
Put in Active Mode

```
---> SYST
215 UNIX Type: L8.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> put prueba
local: prueba remote: prueba
---> TYPE I
200 OK
ftp: setsockopt (ignored): Permission denied
---> PORT 127,0,0,1,229,167
200 OK
---> STOR prueba
150 File status okay; about to open data connection.
125 Data connection already open; transfer starting.
8 bytes sent in 0.00 secs (5.7445 kB/s)
ftp>
```

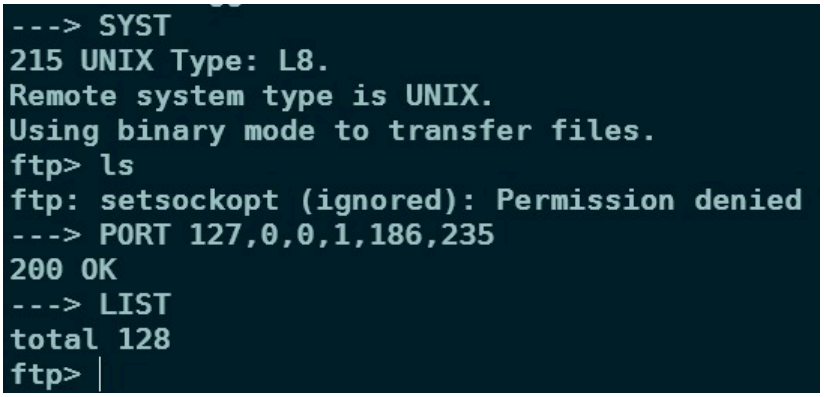
File in Active Mode

```
---> SYST
215 UNIX Type: L8.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> get README
local: README remote: README
---> TYPE I
200 OK
ftp: setsockopt (ignored): Permission denied
---> PORT 127,0,0,1,148,59
200 OK.
---> RETR README
150 File status okay; about to open data connection.
226 Closing data connection.
535 bytes received in 0.00 secs (6.7134 MB/s)
ftp>
```

File in Passive Mode



List in Active Mode



List in Passive Mode