Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

SOCKET PROGRAMMING

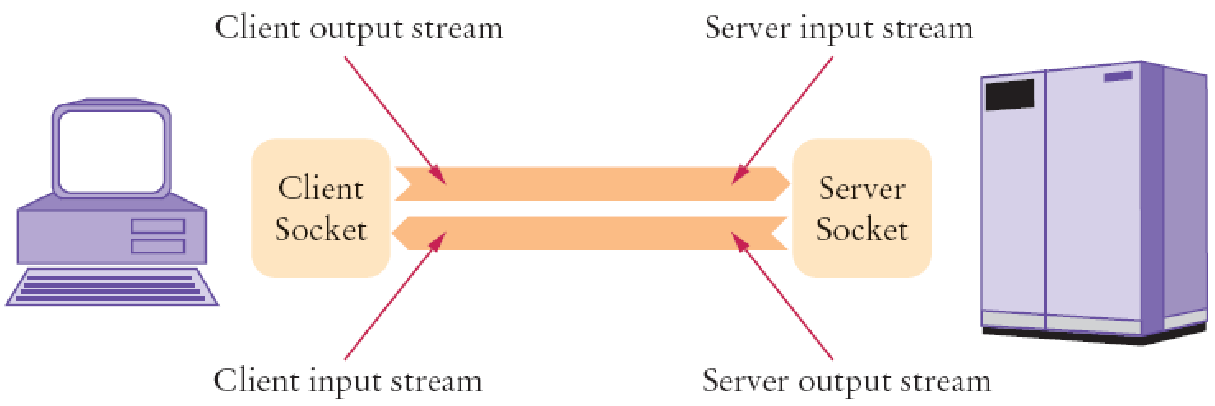


Figure 5 Client and Server Sockets

Name: Samuel.

Surname: Martín Morales.

Course: Second.

Title: Informatics engineering.

Date: 23/05/2021

Contact email address: alu0101359526@ull.edu.es

Mobile phone: 640584774

Index

Brief description of the developed application and description of the developed protocol.

This last practice consists in an implementation of a simple FTP server. This implementation needs to have the basic function of the server, functions like upload and download files from a single folder.

A **FTP server** is a File Transfer Protocol server that transfers files between a client and a server in order to store or retrieve them from the server. This protocol is based on the **TCP** transport protocol and uses ports 20 and 21. These two ports are the control connection (port 21) and the data connection (port 20). **The control connection** is used to transfer commands and the replies to these commands. On the other hand, exists two different file transfer modes (active mode and passive mode).

In the **active mode**, the client starts listening to the port and sends the FTP commands **PORT** to the **FTP server**. The server will then connect back from port 20 to the port of the client. The main problem of this procedure is that this can generate a problem with firewalls that drop non known incoming connections.

In the **passive mode**, the client starts the two types of connections of the **FTP server** (control and data) like active mode. This procedure solves the issue of the active mode.

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

Guide for the compilation of the source code:

First, are necessary the different source codes: ClientConnection.h, ClientConnection.cpp, common.h, FTPServer.h, FTPServer.cpp, ftp_server.cpp, Makefile. The last source code is the most important, because with this source code we can compilate the FTP server program.

Second, in a Ubuntu terminal we need to be in the directory that contains all of the source codes, so, in a terminal and in the directory route, we execute the command **make clean**. Now, we eliminate all of the executable programs that have the name **ftp_server.** Then we execute the command **make all** to compile the program. At this point, we have the **ftp_server.**

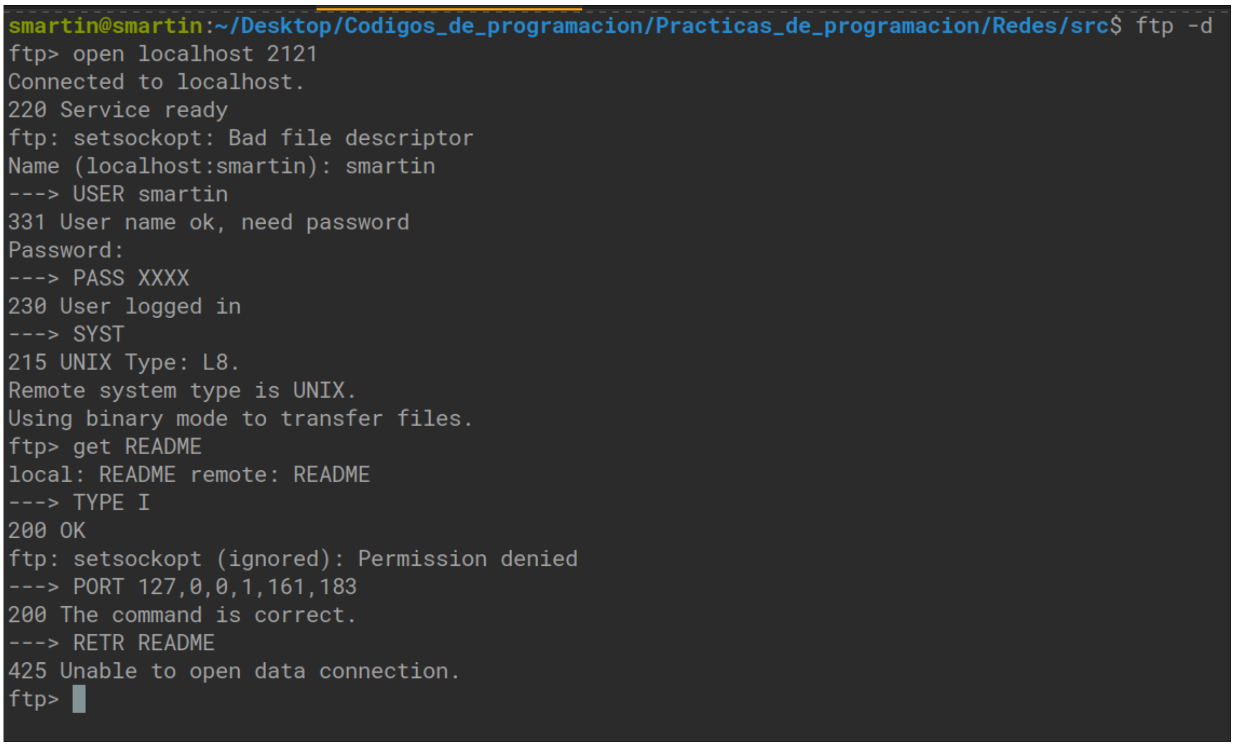
Third, we execute the **FTP server**, so, we use the command **./ftp_server.** Now, the terminal is thinking, so, the ftp server is now connected.

Fourth, we open a new window terminal, so we execute the command **ftp -d** to execute the **FTP client** and connect to the **FTP server.** Finally, we execute the command **open localhost 2121** to start the connection with the FTP server.

Test cases

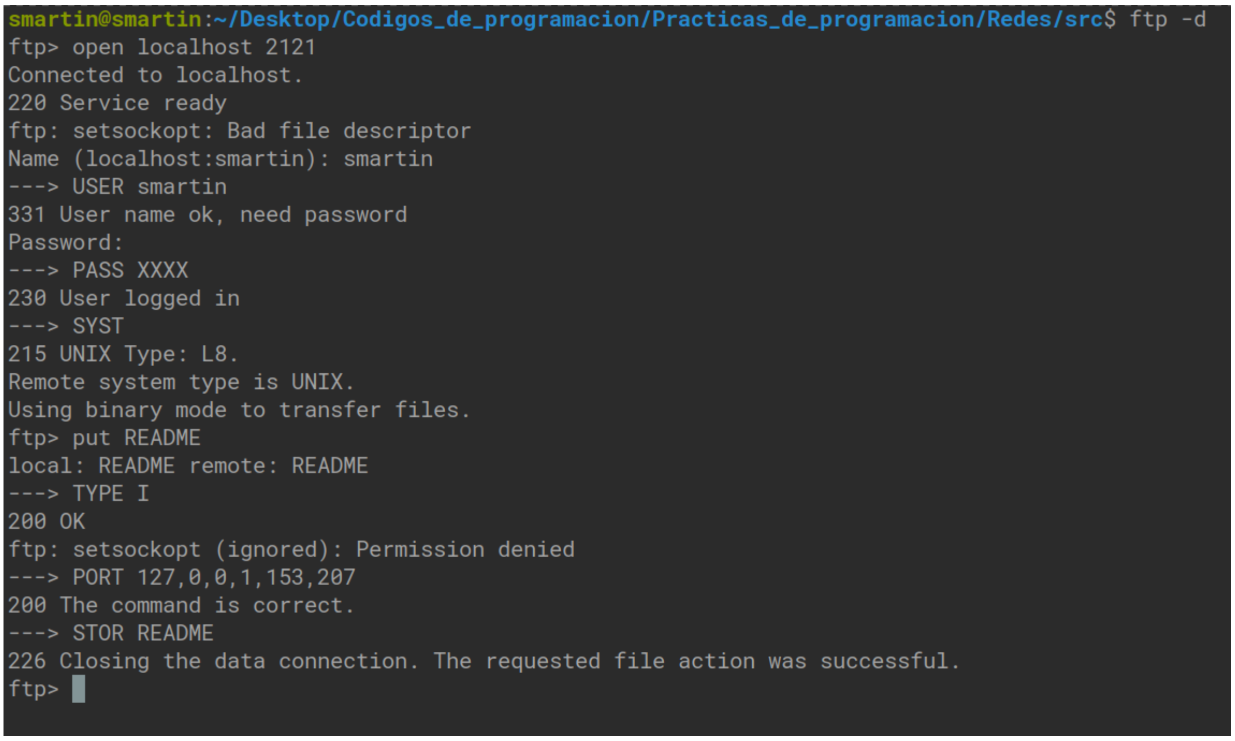
To understand everything described above, now, we can see a series of examples of the **FTP commands**:

Retrieve a remote file:



```
smartin@smartin:~/Desktop/Codigos_de_programacion/Practicas_de_programacion/Redes/src$ ftp -d
ftp> open localhost 2121
Connected to localhost.
220 Service ready
ftp: setsockopt: Bad file descriptor
Name (localhost:smartin): smartin
---> USER smartin
331 User name ok, need password
Password:
---> PASS XXXX
230 User logged in
---> 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 127,0,0,1,161,183
200 The command is correct.
---> RETR README
425 Unable to open data connection.
ftp> 
```

Store a file on the remote host:



```
smartin@smartin:~/Desktop/Codigos_de_programacion/Practicas_de_programacion/Redes/src$ ftp -d
ftp> open localhost 2121
Connected to localhost.
220 Service ready
ftp: setsockopt: Bad file descriptor
Name (localhost:smartin): smartin
---> USER smartin
331 User name ok, need password
Password:
---> PASS XXXX
230 User logged in
---> SYST
215 UNIX Type: L8.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> put README
local: README remote: README
---> TYPE I
200 OK
ftp: setsockopt (ignored): Permission denied
---> PORT 127,0,0,1,153,207
200 The command is correct.
---> STOR README
226 Closing the data connection. The requested file action was successful.
ftp> 
```

Appendix: source code

Link to the directory that content the source files of the FTP server:

https://drive.google.com/drive/folders/1htwTz9yRYjCcpHB1yKnQ1PjUWo41kMgG?usp=sharin
g

Bibliography

Notes to understand the FTP functioning:

https://www.cartagena99.com/recursos/alumnos/apuntes/5_ftp-tftpv3.pdf

Explanation of sockets functions:

http://informatica.uv.es/it3guia/ARS/practicas/Funciones.pdf