FTP CLIENT SERVER REPORT

Eric Rado

4018056

1. Introduction

The task at hand was to implement a file transfer protocol client server without using an ftp library. The programming language of choice for my project was python. At the beginning I was not too familiar with ftp until extensive research was done. I started out first researching the command first and got a knowledge that the system behave like FileZilla or FIU ocelot servers. A peer on the Moodle class forums recommended an excellent website on how the commands work and their given inputs which helped a lot. Once this was done a study of sockets and servers was conducted. During this research I learned that a socket is a way to communicate between two programs such as sending messages or transferring files amongst each other. There are two type of sockets TCP and UDP, so for my project I choose TCP sockets because they are more reliable in data transfer unlike UDP sockets. Reading the textbook assigned for the course enlighten how a server program's socket works such as the methods the sockets should use such as bind, listen, and accept while the client program's socket connects.

2. Problem Statement

The professor assigned us to create an ftp client server system which follow RFC 759 guidelines. The interesting part of the project was that the server was going to hold multiple clients and each client has to be treated in its own separate thread. At first it sounded challenging but this was only the beginning because there was more rules the ftp client server system had to follow. There are four type of clients the server accepts and they are administrator, user, a user that is locked and a user that is not allowed. The administrator has special actions that a regular client does not have such as being able to access any user directory while the user is only

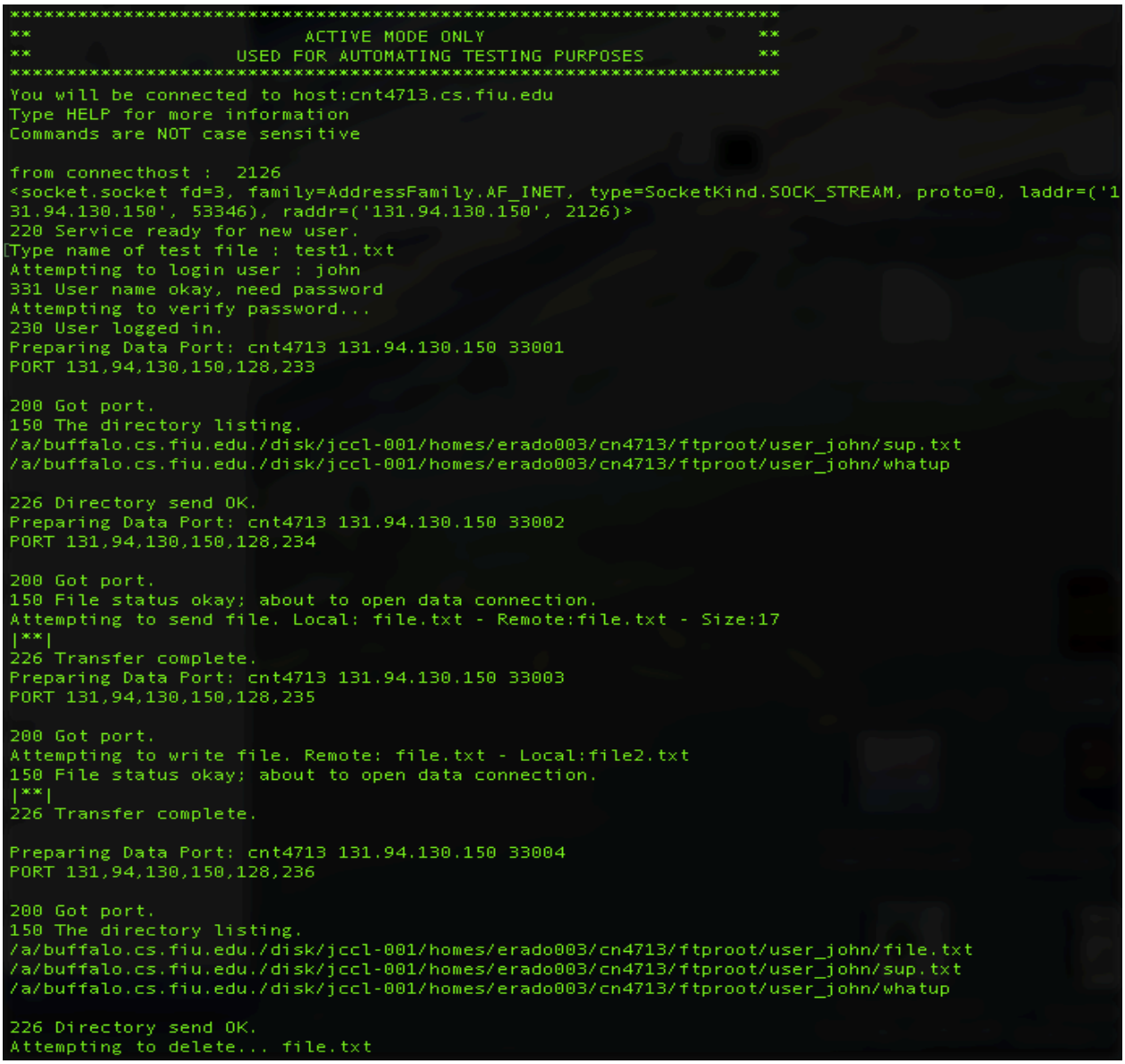
allowed to work on their own personal directory and have no access to another user's directory. In addition, if a user attempts to log in with failure a certain amount of times they are locked from the server. My server has to initiate or close connection from another program through an ftp service port provided by the professor. The server also has to accept ftp commands sent from the client, run the command in the message, and then send a message to client telling the client if the action was taken or not and why. The messages transferred also have to include a number code based on ftp. Since the project was tested on the school server, the professor gave us a port range for each student to test our project so there would not be any collision when testing.

3. Methodology

In order to solve this genuine problem, I started off writing a basic server and client program which accepts a client socket and then the server transfers a message to the client. From here I added multi-threading to the server to accept multiple clients and transfer messages to each of the clients. Each thread on the server was created by a class defined in my program that will handle each client thread. Inside the class I began to write common functions that the client will also have such as login, quit, bye. My methodology when writing code was to work on each side of the ftp simultaneously but always focusing on one function at a time. An example of this was when I wrote the login function I wrote the receiving side which was the server then a send message to the client. I switched to my client program and wrote the code to receive the message sent from the server. Also as I wrote code, each function was tested before moving onto to code the next function unless it depended on another function. A configuration and user file was created to solve this project. The configuration file is in ini format so it can take advantage of a very helpful library called configParser in python. The configuration file contains vital

information that the server needs to know such as the root directory that holds all the user's private directory, port ranges that are allowed for this project, and how many connections can be handled before the server begins to reject connections. The user file is a regular text file which contains the user name, password and permission in that same order.

4. Results



```
Attempting to delete... file.txt
250 Requested file action okay, completed.
Quitting...
Attempting to log out...
220 Command okay, user is logged out
221 Service closing control connection.
Thank you for using FTP
erado003@cnt4713:~/cn4713 222%
```

The following images are a result from test1.txt provided by the professor on the Moodle website.

```
**********************************************************************
**                       ACTIVE MODE ONLY                           **
**                  USED FOR AUTOMATING TESTING PURPOSES            **
**********************************************************************
You will be connected to host:cnt4713.cs.fiu.edu
Type HELP for more information
Commands are NOT case sensitive

from connecthost :  2126
<socket.socket fd=3, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr=('1
31.94.130.150', 53369), raddr=('131.94.130.150', 2126)>
220 Service ready for new user.
Type name of test file : mytest1.txt
Attempting to login user : eric
331 User name okay, need password
Attempting to verify password...
230 User logged in.
Preparing Data Port: cnt4713 131.94.130.150 33001
PORT 131,94,130,150,128,233

200 Got port.
150 The directory listing.
/a/buffalo.cs.fiu.edu./disk/jccl-001/homes/erado003/cn4713/ftproot/admin_eric/hello.txt
/a/buffalo.cs.fiu.edu./disk/jccl-001/homes/erado003/cn4713/ftproot/admin_eric/adminStuffs

226 Directory send OK.
Attempting to delete... hello.txt
250 Requested file action okay, completed.
200 cdup executed
Preparing Data Port: cnt4713 131.94.130.150 33002
PORT 131,94,130,150,128,234

200 Got port.
150 The directory listing.
/a/buffalo.cs.fiu.edu./disk/jccl-001/homes/erado003/cn4713/ftproot/user_fred
/a/buffalo.cs.fiu.edu./disk/jccl-001/homes/erado003/cn4713/ftproot/user_john
/a/buffalo.cs.fiu.edu./disk/jccl-001/homes/erado003/cn4713/ftproot/user_Jasmine
/a/buffalo.cs.fiu.edu./disk/jccl-001/homes/erado003/cn4713/ftproot/admin_eric
/a/buffalo.cs.fiu.edu./disk/jccl-001/homes/erado003/cn4713/ftproot/user_Ortega
/a/buffalo.cs.fiu.edu./disk/jccl-001/homes/erado003/cn4713/ftproot/user_samuel

226 Directory send OK.
Attempting to cd to user_Ortega
200 cd executed
212 Directory created.
Preparing Data Port: cnt4713 131.94.130.150 33003
PORT 131,94,130,150,128,235

200 Got port.
150 The directory listing.
/a/buffalo.cs.fiu.edu./disk/jccl-001/homes/erado003/cn4713/ftproot/user_Ortega/mystuffs
/a/buffalo.cs.fiu.edu./disk/jccl-001/homes/erado003/cn4713/ftproot/user_Ortega/adminMadeIt

226 Directory send OK.
```

The following images are a result of a test created by me in which a the admin logs on to the server then is able to navigate to another user private directory and was able to create a directory in the user directory and is then removed.

5. Analysis

This project has helped me gain knowledge in network programming and python programming language which was new to me before starting the project. Picking up python was not a very difficult task because it's very straightforward and its's library is one of the most helpful of all the programming languages I have encounter. In order to know which libraries and modules were necessary for the project research was done on the web to discover them. Once I found out which libraries and modules were necessary, I read the python help manual to know what modules are contained in the library. For example, when I discovered the os library, I looked at the manual and found out it had modules for testing if a variable is a valid file or directory and removing them if necessary. Sometimes the help manual was hard to comprehend so I searched their functionality on the web on stack overflow or blogs from other programmers. In order to run system tests on my project argument parsing was necessary which I had no clue on how to do. A YouTube video helped enlighten me on this topic. Some of the problems encountered writing code for this project was socket errors, incorrect paths being input, locking a user after incorrectly inputting their password a certain amount of times, change directory not actually moving to the specified directory. The learning experience brought by this project was a

handful and I believe is a project that can be put in a resume because of the overall difficulty it presented. This project has been hands down the most difficult project I have ever been assigned throughout all the courses I have taken but the most enjoyable after the project was finished. This project has helped me gain a ton of knowledge and cannot wait to see what knowledge I will gain in the following project.

6. Reference

https://www.youtube.com/watch?v=q94B9n_2nf0 Python 3 Advanced Tutorial 3 – Argparse

http://www.ftp-commands.com/file-transfer-protocol/append+local-file+%5Bremote-file%5D/

https://www.ietf.org/rfc/rfc959.txt

https://pypi.python.org/pypi/configparser

computer networking a top down approach