Report Sections and Information.

FTP Client and Server for CNT 4713 U01/U02 Net-centric Computing

Alex Batista 5000200

FTP Client and Server implementation:

Problem Statement: Implementation of a client and server that can talk to each other and provide functionality seen in basic ftp servers.

Methodology: The client was provided to us in class and we were only required to fill in certain functionality that we would need our server to handle. This includes storing and store unique, as well as retreive interfaces.

The client proved to be the easier part of the assignment as the client served as a basic interface for the server. You tell the server what you want to do via datasocket and ftp socket commands such as "LIST" or "RETRV" and whatever other data those functions need. The data port was already implemented for us and did the calculations with the min max stuff.

The server implementation was much more complicated and required more in-depth research. Using https://gist.github.com/scturtle/1035886 as my base and the quiz 3 result provided to us in class I was able to construct a working server that at first just dealt with simple operating system commands then eventually port opened commands such as list and store/retrieves. This was the tougher part of the server so breaking it down to just the basics such as make directory, remove directory, delete, cwd and pwd. These commands did not require a data port to manipulate the server and allowed me to get a good base of understanding of how a server works. With that in mind I was able to gain the confidence to begin working within the port command and creation of data sockets that the client will produce to connect to server for the sending and retrieving of data.

The configuration Module was put together from a forum answer question from https://moodle.cis.fiu.edu/v3.1/mod/forum/discuss.php?d=19115#p40223 that gave me a great tutorial to instantly create a configuration file which I could parse and return that data to implement User and Password. It can be easily extended for use with the client configuration as well as providing administration. The module is naively implemented without optimization to be considered. Zipping two small lists into a dictionary is great but if I had 1 million users and 1 million passwords, I would dare not use that implementation

also modifying a user's account would be a nightmare. This is certainly an area for improvement.

Multi-threaded Client was implemented for a later turn in requirement which basically took any datasocket operation and put that operation on a separate thread. This was implemented in the code by those functions acting as the interface instead and throwing their data into a threaded function which then handles those specific implementations such as store or get and running that operation in th thread instead. Because I am primarily using the terminal to test, you have small visual hiccups such as the main thread ftp prompt getting buried in thread print results. Just type as normal.

Results: Implementation of the server and client took roughly about 35 hours. The process was first implementing the client and testing with classftp micarock520 on the cnt4713.cs.fiu.edu and getting back
verbose replies such as running LS command :

```
-rw-rw-rw- 1 user group 30407   ClientMod.py-rw-rw-rw- 1 user group 11393
ClientServerFinal.zip-rw-rw-rw- 1 user group 3340       ConfigFile.py
-rw-rw-rw- 1 user group 4096    Docmentation.txt
FTP>-rw-rw-rw- 1 user group 170 example.ini-rw-rw-rw- 1 user group 14194
Quiz3.py-rw-rw-rw- 1 user group 1167 README.txt-rw-rw-rw- 1 user group 7312
ServerFTP.pydrwxrwxrwx 1 user group 0   testDir-rw-rw-rw- 1 user group 186
testDocSuper.txt-rw-rw-rw- 1 user group 29075   TesterClient.pydrwxrwxrwx 1 user
group 4096     __pycache__
226 Directory send OK.
```

Note: Above you'll notice the multithread hiccup with FTP prompt (to just type as normal.)

Then the implementation of the server came next which was the more difficult endeavor as I had to make sure data port stuff was calculating correctly and being called correctly. The operating system stuff was first hard to grasp but eventually simplified out once I started using the built in OS stuff from python.

Analysis: So one of the things that hit me the hardest when starting this project (despite starting it late) was that there really isn't much going on in the RFC we were directed to. All it really provides is directives but without examples I had to fully dissect the client code to learn how anything really worked. I also found that there was not much online resources for implementing the server as far as tutorials and any simple case studies. This made the project more difficult than it should've been. There certainly is an opportunity here to fill in the gaps for those who will come after us and I think a good opportunity for extra credit. Creating a tutorial for a simple client and server that gives the main idea without all the complicated features. One of the major functionalities to finish the tutorial with is calculating the data_port stuff with great detail and not just a blind/naive implementation (trusting that it works.)

Despite that challenge, I think the project is certainly entertaining as a learning experience when you can work in groups. I worked directly with another classmate Steven Caceres and we implemented all features and functions together while solving problems we had no clue about using reasoning and trial and error approaches. As an extra credit assignment this project would be ideal but as a main project it certainly put the fire under us to quickly analyze the problems we would face with quizzitive reasoning then jump directly into implementation. I think a way to improve or better prepare the students to overcome this obstacle for future classes would be to do code along segments in class that help put together a bare-bones skeleton server allowing students to stop the code along with questions or writing certain portions of the server then quizzing the students on that portion of the server.

References:

- Sample FTP Server(Python 2) - https://gist.github.com/scturtle/1035886
- Pachev l joseph - https://moodle.cis.fiu.edu/v3.1/mod/forum/discuss.php?d=19115#p40223
- Threading Examples(Python 3) - https://docs.python.org/3/library/threading.html
- Operating System commands - https://docs.python.org/3.1/library/os.html