Concurrent and Distributed Programming Assignment 2

Eoin Clayton 16326173
Robin O’Shea 163196296

Declaration on Plagiarism

Assignment Submission Form

This form must be filled in and completed by the student(s) submitting an assignment

Name(s): Eoin Clayton, Robin O’Shea
Programme: CASE 4
Module Code: CA4006
Assignment Title: Client-Server problem
Submission Date: 13/04/2020
Module Coordinator: Rob Brennan

Name(s): Eoin Clayton, Robin O’Shea Date: 13/04/2020

About

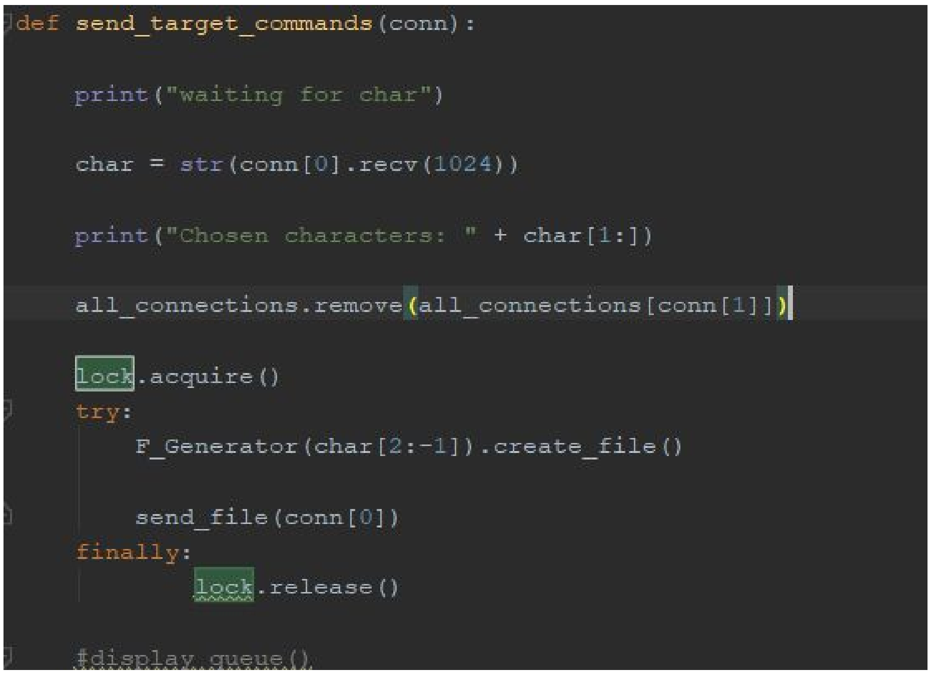
The following assignment was to construct a common client/server interactive service, whereby the client communicates to the server via pattern/filtering requests and from these received requests, the server provides a retrieved list of the chunks in a text file of these corresponding patterns accordingly. The server must accommodate for more than one client outside of its local area network.

Constructing server:

In dealing with the server, there were two main functionalities that needed to be incorporated, both of which needed to be running concurrently i.e Listening for connections, sending confirmation of clients connection while also accepting these connections and handling requests. For this reason, multithreading was integrated to deal with both scenarios. In handling these requests they are passed to file generator class and verified as being a statistical chunk request of our word.txt file and if valid generating a new text file to be supplied to the client otherwise sending the client a message of its invalid request.

Managing connections:

The management of connections is done by a job queue of length two i.e. two threads. Once the server accepts a connection it then passes the client's request into the FileGenerator class at this stage the thread is locked to lock anymore connections from being accepted until the earlier connection has received a response from processing in the file generator class which verifies if the request is valid as a parameter to generate a new text file, if so A new text file will be generated and sent back to the client otherwise the output 'Invalid input' is issued to the client after that the thread is unlocked and the thread continues accepting connections

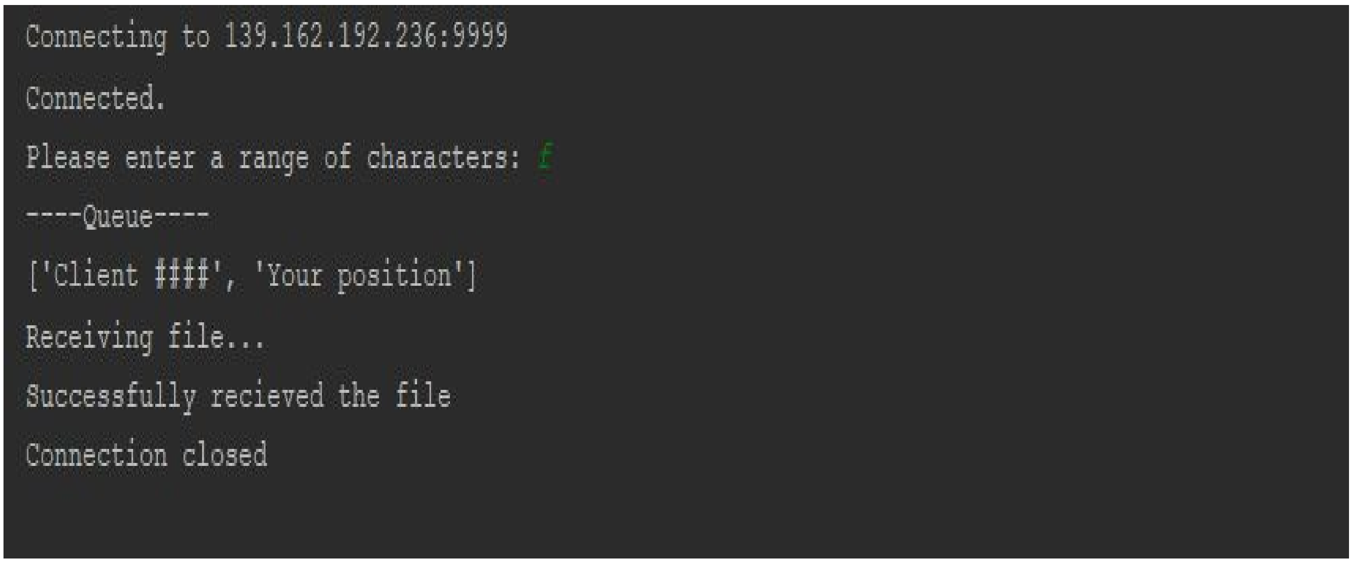


```
def send_target_commands(conn):

    print("waiting for char")

    char = str(conn[0].recv(1024))

    print("Chosen characters: " + char[1:])

    all_connections.remove(all_connections[conn[1]])

    lock.acquire()
    try:
        F_Generator(char[2:-1]).create_file()

        send_file(conn[0])
    finally:
            lock.release()

    #display_queue()
```

Client.py Example

The image below represents the client-side command prompt interface from a machine when connecting and sending out a filtering request and receiving a text file from a corresponding request from the original text file. The following lines are output to the terminal numerically

1. Here we can see that we are connecting to the server
2. We are now prompted that we are connected
3. We are then asked from the server of our requested data, this can be in the format A-C implying a range of words that begin with A, B or C, or singular such as A or concatenated like AB to strings beginning with a or b
4. A queue of clients requesting metadata on the server is sent to display the client's current position in the queues read from left to right
5. The file is now created and being sent from the server
6. data is retrieved and file created in clients directory

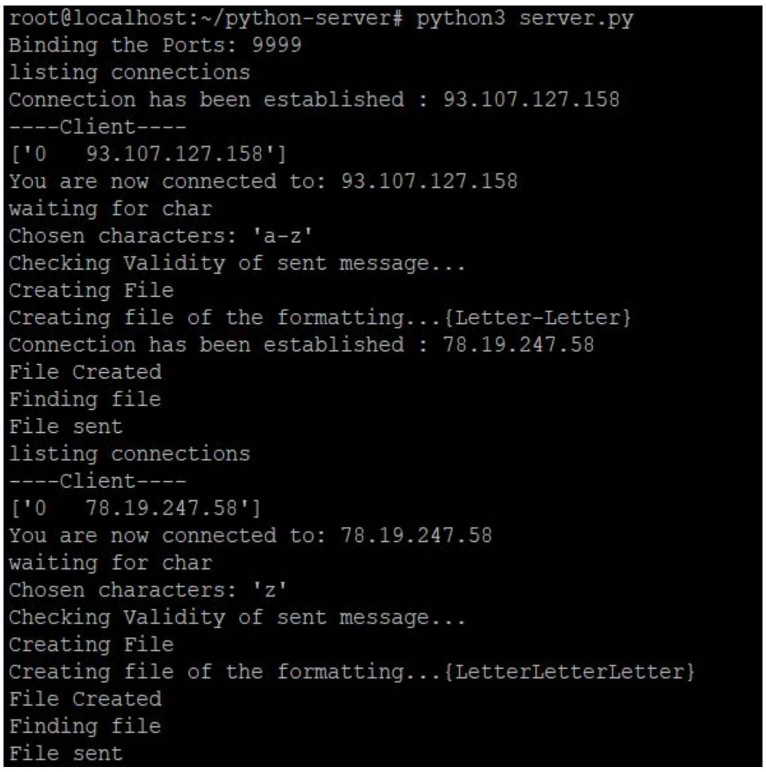


```
Connecting to 139.162.192.236:9999

Connected.

Please enter a range of characters: f

----Queue----

['Client ####', 'Your position']

Receiving file...

Successfully recieved the file

Connection closed
```

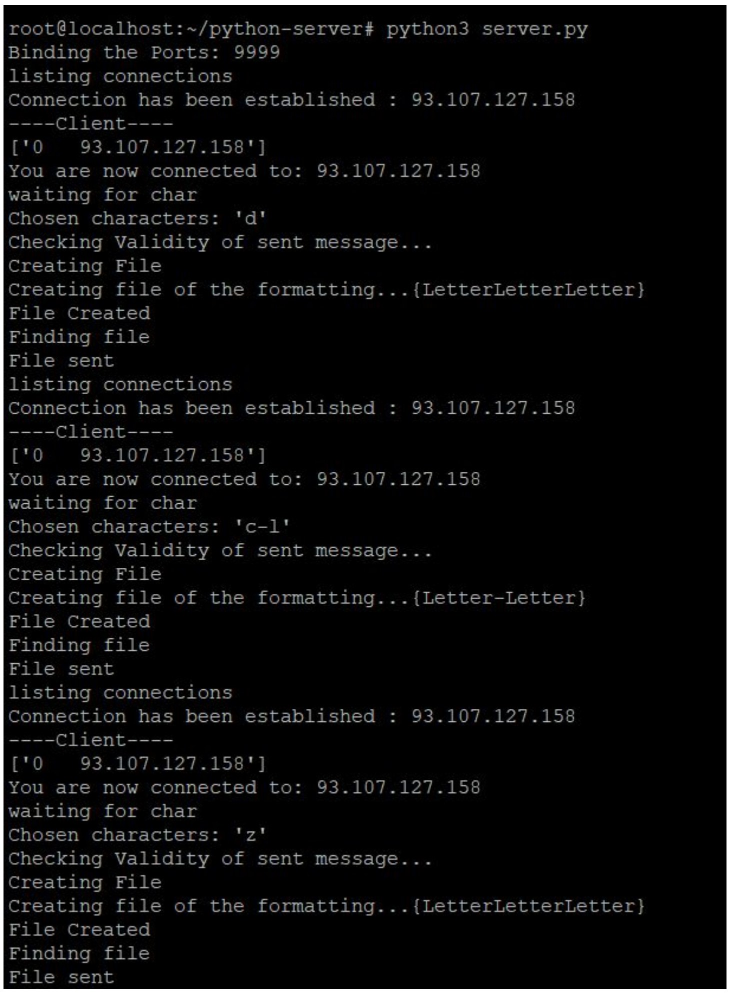
Server.py example

Once the server starts running, it prints the port number it is binding to and waits for a client to connect to it.
Once a client connects it displays the IP address of the client which is only visible on the server-side.
The server then waits for the connected client to send a chunk containing characters eg. a-c.
These characters are then checked for validity and if valid is sent to file_generator.py to create the necessary output file.
File_generator.py takes the character range and searches through a text file of words and writes all words beginning with the relevant letter however many times is said in the file. Eg. a is sent to the server and a file containing all words beginning with a is created.
This new file is then saved to the same directory as the server and sent to the client and saved in the same directory as the client.py file.
Once the file is sent the server ends its connection with the client.
The server can also have multiple connections at the one time but can only create one file at a given time.

The following are two examples of the server.py file running:



```
root@localhost:~/python-server# python3 server.py
Binding the Ports: 9999
listing connections
Connection has been established : 93.107.127.158
----Client----
['0   93.107.127.158']
You are now connected to: 93.107.127.158
waiting for char
Chosen characters: 'a-z'
Checking Validity of sent message...
Creating File
Creating file of the formatting...{Letter-Letter}
Connection has been established : 78.19.247.58
File Created
Finding file
File sent
listing connections
----Client----
['0   78.19.247.58']
You are now connected to: 78.19.247.58
waiting for char
Chosen characters: 'z'
Checking Validity of sent message...
Creating File
Creating file of the formatting...{LetterLetterLetter}
File Created
Finding file
File sent
```

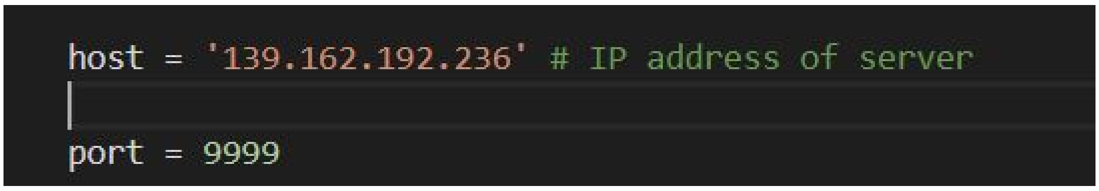


```
root@localhost:~/python-server# python3 server.py
Binding the Ports: 9999
listing connections
Connection has been established : 93.107.127.158
----Client----
['0   93.107.127.158']
You are now connected to: 93.107.127.158
waiting for char
Chosen characters: 'd'
Checking Validity of sent message...
Creating File
Creating file of the formatting...{LetterLetterLetter}
File Created
Finding file
File sent
listing connections
Connection has been established : 93.107.127.158
----Client----
['0   93.107.127.158']
You are now connected to: 93.107.127.158
waiting for char
Chosen characters: 'c-l'
Checking Validity of sent message...
Creating File
Creating file of the formatting...{Letter-Letter}
File Created
Finding file
File sent
listing connections
Connection has been established : 93.107.127.158
----Client----
['0   93.107.127.158']
You are now connected to: 93.107.127.158
waiting for char
Chosen characters: 'z'
Checking Validity of sent message...
Creating File
Creating file of the formatting...{LetterLetterLetter}
File Created
Finding file
File sent
```

How to run

Running locally

Once all files are downloaded. The only change needed is in the client.py file.



```
host = '139.162.192.236' # IP address of server

port = 9999
```

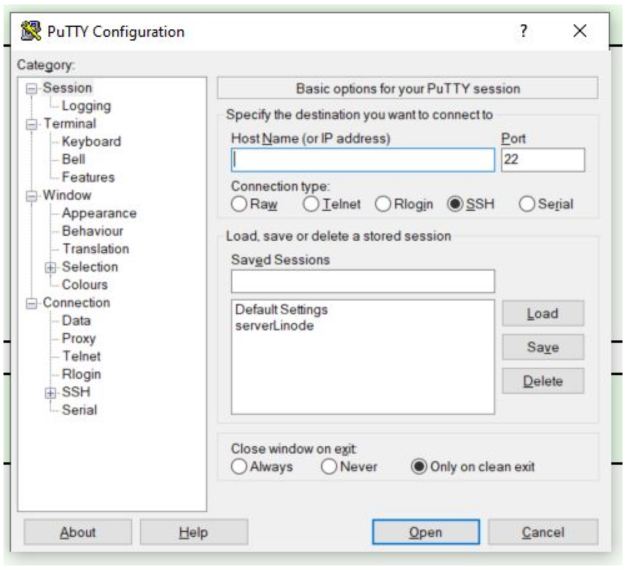
The host variable needs to be changed to the IP address in which the server is located
This IP address can be found by running ipconfig in the command prompt and look for IPv4 address.
Copy and paste that ip address into the host variable.

Once this is done, using the command prompt navigate to the directory that the server.py is located in and run ‘python server.py’. This will start the server.
Then using another command prompt navigate to the directory where the client.py is located and run ‘python client.py’. The client will then be connected to the server.
The client.py file can be run multiple times in the terminal.

Running through putty

Download the client.py file to your local machine
Ensure putty is downloaded on your machine.
https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

Once installed the following screen will appear.



Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Saved Sessions: Default Settings, serverLinode
Close window on exit: Always Never [✓ Only on clean exit]

In the Host name box enter ‘139.162.192.236’ and press open.
You will be asked to login and enter a password. The login name is ‘root’ and password is ‘Assignment2’’
A terminal will then be displayed. Enter ‘cd python-server’ to navigate to the directory and enter ‘python3 server.py’ to run the server.
The client.py file is then run locally from your command prompt. Navigate to the directory it is stored in and run ‘client.py’
The client.py file can also be run on multiple machines.