# Computer networks TCP assignment

SE20UARI147

B.Sreevaatsav

Programming language used :- Python

For transferring files using TCP, we first create a server socket which receives the file and send the data from client.

**Proper comments have been added to the both client.py and server.py files.**

Here, I have transferred a CSV file from a windows PC to another MAC pc using TCP protocol.

Github link :- https://github.com/SreevaatsavB/CNlite

Follow the instructions in the README file to start file transferring.

Code explanation and screenshots in next slides:-

EXPLORER

CNLITE-MAIN

client.py

server.py

server.py

server.py > server_prog

```python
import socket
import os

# Device's IP address

SERVER_HOST = [redacted]
# SERVER_HOST = '''Device's IP address'''
# SERVER_HOST = "0.0.0.0"
SERVER_PORT = 6666
BUFFER_SIZE = 9216


def server_prog():
    # Creating a server socket to recieve the data
    server_socket = socket.socket()

    # making a server socket
    server_socket.bind((SERVER_HOST, SERVER_PORT))

    # 6 unaccepted connections that the system will allow before refusing ne
    server_socket.listen(6)
    print("LISTENING AS ",SERVER_HOST,":",SERVER_PORT)

    # Accepting the connection made
    client_socket, ADDRESS = server_socket.accept()

    print(ADDRESS, "IS CONNECTED")

    filename = client_socket.recv(BUFFER_SIZE).decode("utf-8")
    print("FILENAME IS ", filename)
    file = open(filename, "w")
    # After the filename was succesfully recieved, server opens a new file w
    client_socket.send("FILE NAME RECIEVED".encode("utf-8"))

    # Recieving the data from sender(client socket)
    data = client_socket.recv(BUFFER_SIZE).decode("utf-8")
    print("Length of the data recieved :- ",len(data), " Bytes")
    # Writing that daat into the file opened
    file.write(data)
    client_socket.send("DATA RECIEVED".encode("utf-8"))

    # Closing the sockets after sucessfull data transfer
    print("File recieved and read")
    client_socket.close()
    print("FILE TRANSFER DONE")
    server_socket.close()
    print("DISCONNECTED")
```

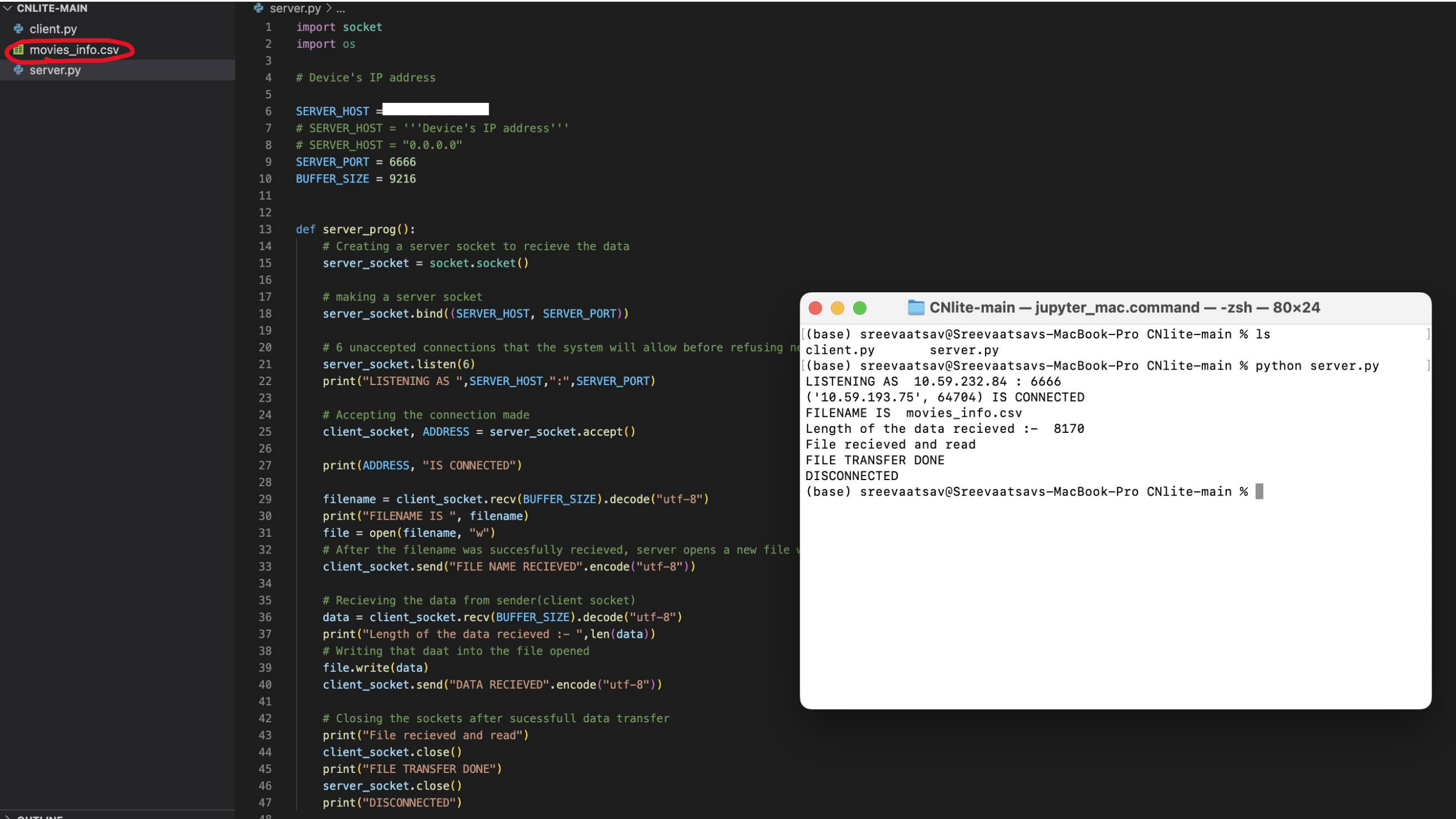
CNlite-main — jupyter_mac.command — python server.py — 80×24

```
(base) sreevaatsav@Sreevaatsavs-MacBook-Pro CNlite-main % ls
client.py       server.py
(base) sreevaatsav@Sreevaatsavs-MacBook-Pro CNlite-main % python server.py
LISTENING AS  10.59.232.84 : 6666
('10.59.193.75', 64704) IS CONNECTED
FILENAME IS  movies_info.csv
Length of the data recieved :-  8170
File recieved and read
FILE TRANSFER DONE
DISCONNECTED
(base) sreevaatsav@Sreevaatsavs-MacBook-Pro CNlite-main % python server.py
LISTENING AS  10.59.232.84 : 6666
```

The above screenshot is when the server waits for the TCP connection with client, here we can see that there's no movies_info.csv in the current working directory.

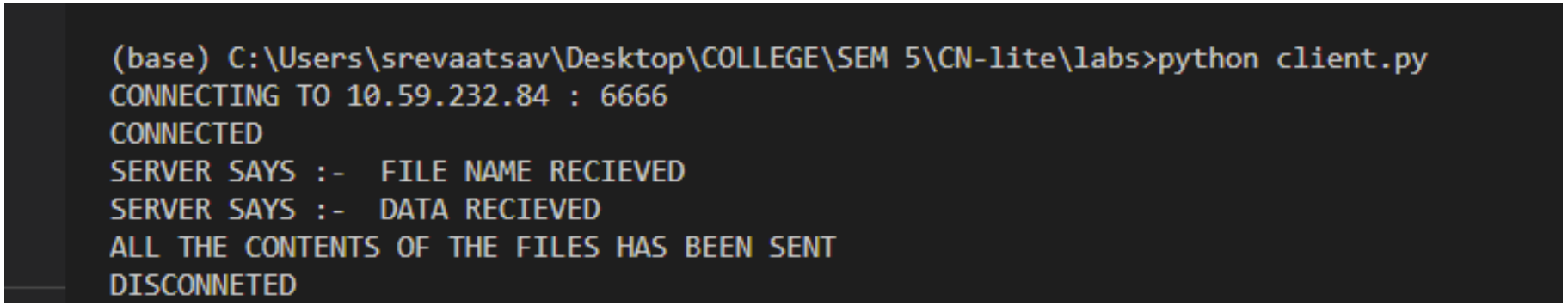
The next screenshot is after the file transfer from the server's(receiver) side. We can see that the file movies_info.csv will be created in the current working directory. (Highlighted in red)

Here the IP addresses were not shown, enter the desired IP addresses while working with the file.


CNLITE-MAIN
client.py
movies_info.csv
server.py
server.py > ...
1 import socket
2 import os
3
4 # Device's IP address
5
6 SERVER_HOST =
7 # SERVER_HOST = '''Device's IP address'''
8 # SERVER_HOST = "0.0.0.0"
9 SERVER_PORT = 6666
10 BUFFER_SIZE = 9216
11
12
13 def server_prog():
14 # Creating a server socket to recieve the data
15 server_socket = socket.socket()
16
17 # making a server socket
18 server_socket.bind((SERVER_HOST, SERVER_PORT))
19
20 # 6 unaccepted connections that the system will allow before refusing ne
21 server_socket.listen(6)
22 print("LISTENING AS ",SERVER_HOST,":",SERVER_PORT)
23
24 # Accepting the connection made
25 client_socket, ADDRESS = server_socket.accept()
26
27 print(ADDRESS, "IS CONNECTED")
28
29 filename = client_socket.recv(BUFFER_SIZE).decode("utf-8")
30 print("FILENAME IS ", filename)
31 file = open(filename, "w")
32 # After the filename was succesfully recieved, server opens a new file w
33 client_socket.send("FILE NAME RECIEVED".encode("utf-8"))
34
35 # Recieving the data from sender(client socket)
36 data = client_socket.recv(BUFFER_SIZE).decode("utf-8")
37 print("Length of the data recieved :- ",len(data))
38 # Writing that daat into the file opened
39 file.write(data)
40 client_socket.send("DATA RECIEVED".encode("utf-8"))
41
42 # Closing the sockets after sucessfull data transfer
43 print("File recieved and read")
44 client_socket.close()
45 print("FILE TRANSFER DONE")
46 server_socket.close()
47 print("DISCONNECTED")
CNlite-main — jupyter_mac.command — -zsh — 80×24
(base) sreevaatsav@Sreevaatsavs-MacBook-Pro CNlite-main % ls
client.py server.py
(base) sreevaatsav@Sreevaatsavs-MacBook-Pro CNlite-main % python server.py
LISTENING AS 10.59.232.84 : 6666
('10.59.193.75', 64704) IS CONNECTED
FILENAME IS movies_info.csv
Length of the data recieved :- 8170
File recieved and read
FILE TRANSFER DONE
DISCONNECTED
(base) sreevaatsav@Sreevaatsavs-MacBook-Pro CNlite-main %

Terminal output at server (receiver's side)


Terminal output at client(sender's side)



client.py

Users > sreevaatsav > Downloads > CNlite-main > client.py > ...

```python
import socket
import os

BUFFER_SIZE = 9216
# The reciever's (IPv4) address (of wireless connection)

HOST = '''Device's IP address'''

# Port number
PORT = 6666

# Filename of the file to be transferred
filename = "movies_info.csv"
def client_prog():

    # Creating a socket for the client to send the file
    client_socket = socket.socket()

    # Connecting to the server with TCP protocol
    print("CONNECTING TO", HOST, ":", PORT)
    client_socket.connect((HOST, PORT))
    print("CONNECTED")

    # reading the file
    file = open(filename, "r")
    data = file.read()
    print("Length of data beign sent :-", len(data), "Bytes")
    # We use the utf-8 encoding and decoding to transfer data via TCPs

    # Sending file name to the server(reciever), so that it can save the data into that file
    client_socket.send(filename.encode("utf-8"))
    # Reading the message Server sends
    msg1 = client_socket.recv(BUFFER_SIZE).decode("utf-8")
    print("SERVER SAYS :- ", msg1)

    # Sending the data
    client_socket.send(data.encode("utf-8"))
    # Reading the message Server sends
    msg2 = client_socket.recv(BUFFER_SIZE).decode("utf-8")
    print("SERVER SAYS :- ", msg2)

    # Closing file after all the files contents have been sent
    print("ALL THE CONTENTS OF THE FILES HAS BEEN SENT")
    file.close()

    # Closing the client socket
    client_socket.close()
    print("DISCONNETED")

if __name__ == "__main__":
    client_prog()
```

## Client code

The above screenshot is the client's program, we first set IPv4 address of the host and then buffer size for transferring the data and set the port numbers same for both the sender and receiver.

**All the data transfer is done with utf-8 encoding and decoding.**

Then we connect client with the server and 1st send the filename to server and read the reply of server.

Thereafter, we send the data with the max buffer preset by us via the same TCP connection. After the server receives the complete data, we close the connection.

## Server code

The next screenshot is of the server's program, here again, set the constant variables (IP address, max buffer to receive, and port number) and then wait until the client connects to it.

After a successful connection, it 1st gets the filename, then we create a new file in write mode with the same filename and send the acknowledgment messages, after that, we get the data from client and write that encoded data by decoding them into the file.

Then ,we close the connection

server.py

Users > sreevaatsav > Downloads > CNlite-main > server.py > ...

```python
import socket
import os

# Device's IP address

SERVER_HOST = '''Device's IP address'''
# SERVER_HOST = "0.0.0.0"
SERVER_PORT = 6666
BUFFER_SIZE = 9216


def server_prog():
    # Creating a server socket to recieve the data
    server_socket = socket.socket()

    # making a server socket
    server_socket.bind((SERVER_HOST, SERVER_PORT))

    # 6 unaccepted connections that the system will allow before refusing new connections.
    server_socket.listen(6)
    print("LISTENING AS ",SERVER_HOST,":",SERVER_PORT)

    # Accepting the connection made
    client_socket, ADDRESS = server_socket.accept()

    print(ADDRESS, "IS CONNECTED")

    filename = client_socket.recv(BUFFER_SIZE).decode("utf-8")
    print("FILENAME IS ", filename)
    file = open(filename, "w")
    # After the filename was succesfully recieved, server opens a new file with same filename and sends a message
    client_socket.send("FILE NAME RECIEVED".encode("utf-8"))

    # Recieving the data from sender(client socket)
    data = client_socket.recv(BUFFER_SIZE).decode("utf-8")
    print("Length of the data recieved :- ",len(data))
    # Writing that daat into the file opened
    file.write(data)
    client_socket.send("DATA RECIEVED".encode("utf-8"))

    # Closing the sockets after sucessfull data transfer
    print("File recieved and read")
    client_socket.close()
    print("FILE TRANSFER DONE")
    server_socket.close()
    print("DISCONNECTED")


if __name__ == "__main__":
    server_prog()
```