Experiment No. 11

Date - 16/06/2021

Concurrent File Server

Aim:

To write a program to develop a concurrent file server which will provide the file requested by the client along with the server (PID). If no file exists, the server sends appropriate messages to the client.

THEORY:

There are two main classes of servers, iterative and concurrent. An *iterative* server iterates through each client, handling it one at a time. A *concurrent* server handles multiple clients at the same time. The simplest technique for a concurrent server is to call the fork function, creating one child process for each client. An alternative technique is to use *threads* instead (i.e., light-weight processes).

The following list describes the concurrent server process -

1. When a connection request arrives in the main process of a concurrent server, it schedules a child process and forwards the connection to the child process.
2. The child process takes the connection from the main process.
3. The child process receives the client request, processes it, and returns a reply to the client.
4. The connection is closed, and the child process terminates or signals to the main process that it is available for a new connection.

Algorithm

a) Server

1. Start
2. Import modules socket, threading
3. Initialize port and ip variables
4. Create a socket s
5. Bind socket with ip and port
6. Create the thread for handling client requests
7. Send the thread id to the requested client
8. Server receives the file name from the client
9. If file exists the file contents are send to requested client
10. If not response message is send
11. Server is shut downed
12. Stop

b) Client

1. Start
2. Import modules socket, os
3. Initialize port and ip variables
4. Create a socket s
5. Connect socket with ip and port
6. User inputs the filename and the response is send to server
7. Client receives the file contents from the server and copies to a new file
8. Socket is closed
9. Stop

Program

a) Server

```
import socket
import threading
import os

class Server:
    def __init__(self):
        self.s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
        self.accept_connections()


    def accept_connections(self):
        ip = socket.gethostbyname(socket.gethostname())
        port = int(input('Enter desired port -> '))

        self.s.bind((ip,port))
        self.s.listen(100)

        print('Running on IP: '+ip)
        print('Running on Port: '+str(port))

        while 1:
            c, addr = self.s.accept()
            print(c)


threading.Thread(target=self.handle_client,args=(c,addr,)).start()

    def handle_client(self,c,addr):
        data = c.recv(1024).decode()
        c.send(str(addr).encode('utf-8'))

        if not os.path.exists(data):
            c.send("file-doesn't-exist".encode())

        else:
            c.send("file-exists".encode())
            print('Sending',data)
            if data != '':
                file = open(data,'rb')
                data = file.read(1024)
```

```
                while data:
                    c.send(data)
                    data = file.read(1024)


                c.shutdown(socket.SHUT_RDWR)
                c.close()


server = Server()
```

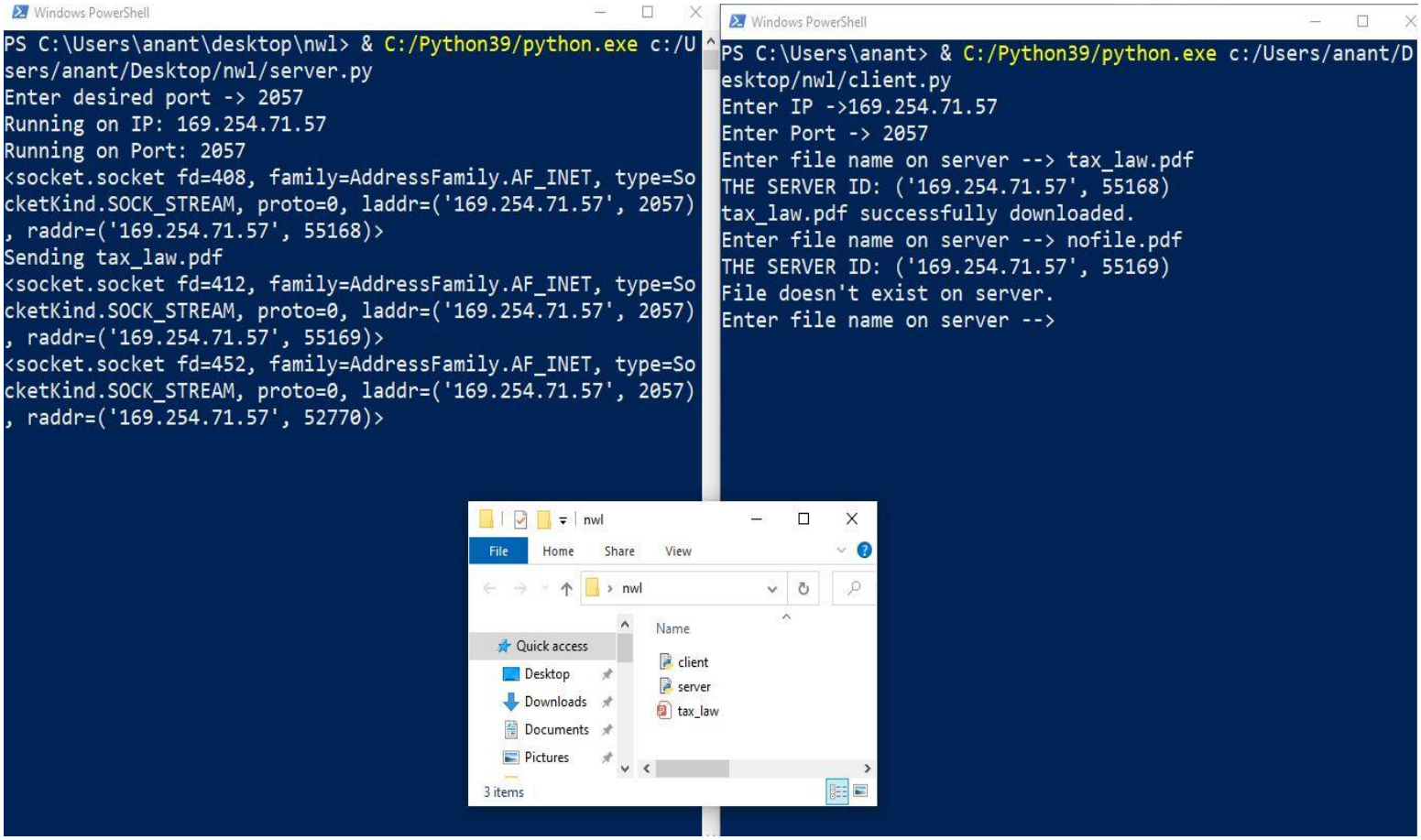
b) Client

```
import socket
import os

class Client:
    def __init__(self):
        self.s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
        self.connect_to_server()

    def connect_to_server(self):
        self.target_ip = input('Enter IP ->')
        self.target_port = input('Enter Port -> ')

        self.s.connect((self.target_ip,int(self.target_port)))

        self.main()

    def reconnect(self):
        self.s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
        self.s.connect((self.target_ip,int(self.target_port)))

    def main(self):
        while 1:
            file_name = input('Enter file name on server --> ')
            self.s.send(file_name.encode())
            id = self.s.recv(1024).decode('utf-8')
            print("THE SERVER ID: "+str(id))

            confirmation = self.s.recv(1024)
```

```
            if confirmation.decode() == "file-doesn't-exist":
                print("File doesn't exist on server.")

                self.s.shutdown(socket.SHUT_RDWR)
                self.s.close()
                self.reconnect()

            else:
                write_name = 'from_server '+file_name
                if os.path.exists(write_name): os.remove(write_name)

                with open(write_name,'wb') as file:
                    while 1:
                        data = self.s.recv(1024)

                        if not data:
                            break

                        file.write(data)

                print(file_name,'successfully downloaded.')


                self.s.shutdown(socket.SHUT_RDWR)
                self.s.close()
                self.reconnect()

client = Client()
```

Output

Server | Client



Result

Successfully executed a program to develop a concurrent file server which will provide the file requested by the client along with the server (PID).