17 Concurrent File Server

17.1 Aim

Develop concurrent file server which will provide the file requested by client if it exists. If not server sends appropriate message to the client. Server should also send its process ID (PID) to clients for display along with file or the message.*

17.2 Theory

- **FTP** : The file transfer protocol is a standard network protocol used for sending files over a network between a server and a client.It uses a client-server architecture where the client connects to a server and requests a file which if exists is send back to the client by the server.FTP server may run in active mode or passive mode which determines how the data connection is established.In **active mode** the client starts listening for incoming data connections from the server on some port M.It then sends the ftp command PORT M to the server.The server then opens a data channel to the client from its port 20. In **passive mode** the client is behind a firewall and unable to accept any incoming tcp connections.In this situation,the client sends the PASV command to the server to which the server sends back the server IP address and Port number.The client uses this IP address and PORT number to open a data channel between the client and server.

17.3 Algorithm

Algorithm 1 Server

```
Open a TCP socket
Bind the TCP socket to HOST,PORT
Listen on the socket for connections
for ;; do
    Accept incoming connection
    Create a thread to process the incoming connection.The thread will take a filename as input from the client and sends back the file if the file exists along with the server PID
end for
```

Algorithm 2 Client

```
Open a TCP socket
Connect to the TCP socket at HOST,PORT
Take a file name as input from the user and send the filename to the TCP socket
Save the first response from the server as server Process ID and print it
Print the second response from the server and save the second response as a file
Exit
```

17.4 Code

Server

```python
import socket
import threading
import os

HOST = '127.0.0.1'
PORT = 65432

def userCallBack(conn,addr):
    while True:
        filename = conn.recv(1024)
        print(filename)
        filename = filename.decode()
        print(filename)
        if filename == "exit":
            exit(0)
        data = ""
        try:
            if filename != "":
                file = open("./files/"+filename,'r')
                data = file.read()
                if data == "":
                    data = "File Contains no data"
        except FileNotFoundError:
            data = "File Not Found !"
        pid = os.getpid()
        conn.sendall(str(pid).encode())
        conn.sendall(data.encode())
s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.bind((HOST,PORT))
s.listen()

while True:

    conn,addr = s.accept()
    print("Connected by :",addr)
    t = threading.Thread(target=userCallBack,args=(conn,addr))
    t.start()
```

Client

```python
import socket

HOST = "127.0.0.1"
PORT = 65432

s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect((HOST,PORT))
while True:
    filename = input("Enter the name of the file :")

    s.sendall(filename.encode())
    if filename == "exit":
        exit(0)
    server_pid = s.recv(1024)
    server_pid = server_pid.decode()
    data = s.recv(1024)
    data = data.decode()
    print("Server PID :",server_pid)
    if data != "File Not Found !":
        f = open("./downloaded/"+filename,'w')
        f.write(data)
        print("\nFile Data \n",data)
    else:
        print("\nFile Not Found")
```

17.5 Output

```
Server PID : 3060

File Data
 helloworld

Enter the name of the file :abc
Server PID : 3060

File Not Found
```

17.6 Result

Concurrent File Server was implemented in python3