Experiment : 11
Date : 16/06/2021
Author : Bonnie Simon

# Concurrent File Server

## AIM

To write a program to develop a concurrent file server which will provide the file requested by the client if it exists.

## THEORY

Unlike a sequential server, a concurrent server has to be able to serve more than one client at a time. For example, a chat server may be serving a specific client for hours-it cannot wait till it stops serving a client before it serves the next one.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.

## ALGORITHM

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

Client algorithm:

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

# PROGRAM

Client.py

```
import socket
import os

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

        def server_connection(self):
        self.target_ip ='127.0.1.1'
        self.target_port =3000

        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: ')
        self.s.send(file_name.encode())
```

```
        confirmation = self.s.recv(1024)
        if confirmation.decode() == "file-doesn't-exist":
                print("File not found")

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

        else:
                write_name = 'from_server '+file_name
                c=self.s.recv(1024).decode()
                print(c)
                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,'File received.')


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

client = Client()
```

Server.py

```
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 ='127.0.1.1' #socket.gethostbyname(socket.gethostname())
        port = 3000#int(input('Enter desired port --> '))

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

        print('IP: '+ip)
        print('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()

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

        else:
        c.send("file-exists".encode())
        c.send(bytes(str(addr),'utf-8'))
        print(addr)
        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()
```

# OUTPUT


# RESULT

The python program to implement a concurrent file server has been executed and verified successfully.