

Assignment - 10 Socket

Programming - File Transfer (Multiple Clients)

Y.BHUPENDRA
2023008358

File Transferring for Multiple Clients in Socket Programming includes:

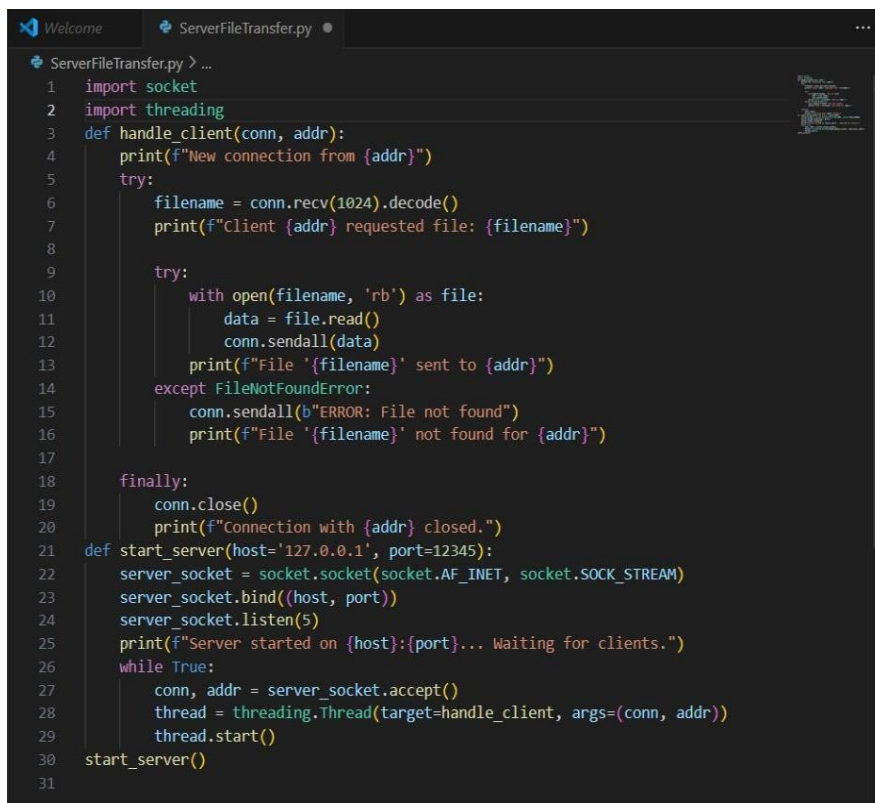
Server

- Listens for incoming client connections.
- Uses **multithreading** to handle multiple clients concurrently.
- Sends a requested file to the connected client.

Client

- Connects to the server.
- Requests a file from the server.
- Receives and saves the file locally.

Server code:



```
1 import socket
2 import threading
3 def handle_client(conn, addr):
4     print(f"New connection from {addr}")
5     try:
6         filename = conn.recv(1024).decode()
7         print(f"Client {addr} requested file: {filename}")
8
9         try:
10             with open(filename, 'rb') as file:
11                 data = file.read()
12                 conn.sendall(data)
13                 print(f"File '{filename}' sent to {addr}")
14         except FileNotFoundError:
15             conn.sendall(b"ERROR: File not found")
16             print(f"File '{filename}' not found for {addr}")
17
18     finally:
19         conn.close()
20         print(f"Connection with {addr} closed.")
21 def start_server(host='127.0.0.1', port=12345):
22     server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
23     server_socket.bind((host, port))
24     server_socket.listen(5)
25     print(f"Server started on {host}:{port}... Waiting for clients.")
26     while True:
27         conn, addr = server_socket.accept()
28         thread = threading.Thread(target=handle_client, args=(conn, addr))
29         thread.start()
30 start_server()
31
```

Clint Code:

```
ClintFileTransefer.py •
ClintFileTransefer.py > ...
1  import socket
2  def request_file(filename, host='127.0.0.1', port=12345):
3      client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
4      client_socket.connect((host, port))
5      print(f"Connected to server at {host}:{port}")
6      client_socket.sendall(filename.encode())
7      with open(f"received_{filename}", 'wb') as file:
8          while True:
9              data = client_socket.recv(1024)
10             if not data:
11                 break
12             file.write(data)
13
14     print(f"File '{filename}' received successfully.")
15     client_socket.close()
16 file_to_request = input("Enter the filename to download: ")
17 request_file(file_to_request)
18
```

Its Works like:

Server Side

1. Creates a TCP socket and binds it to a port.
2. Waits for incoming client connections.
3. When a client connects:

Spawns a new thread using `threading.Thread(target=handle client, args=(conn, addr))`.

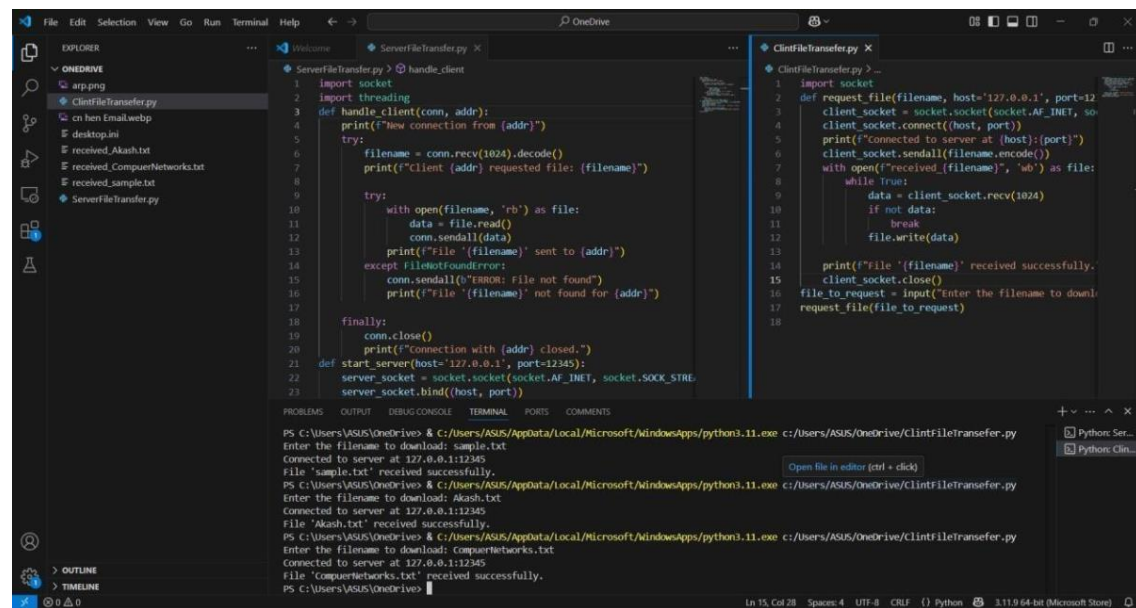
This allows handling multiple clients simultaneously.

4. Receives the filename from the client.
5. Checks if the file exists:
 - If found, reads and sends the file.
 - If not, sends an error message.

Client Side

1. Connects to the server.
2. Sends the filename it wants to download.
3. Receives the file and saves it locally.
4. Handles errors if the file is not found.

OUTPUT:



```
ServerFileTransfer.py
1 import socket
2 import threading
3 def handle_client(conn, addr):
4     print(f"New connection from {addr}")
5     try:
6         filename = conn.recv(1024).decode()
7         print(f"Client {addr} requested file: {filename}")
8
9         try:
10             with open(filename, "rb") as file:
11                 data = file.read()
12                 conn.sendall(data)
13                 print(f"File '{filename}' sent to {addr}")
14             except FileNotFoundError:
15                 conn.sendall(b"ERROR: File not found")
16                 print(f"File '{filename}' not found for {addr}")
17         finally:
18             conn.close()
19     except:
20         print(f"Connection with {addr} closed.")
21
22 def start_server(host="127.0.0.1", port=12345):
23     server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
24     server_socket.bind((host, port))
25     server_socket.listen(1)
26     while True:
27         conn, addr = server_socket.accept()
28         thread = threading.Thread(target=handle_client, args=(conn, addr))
29         thread.start()
30
31 if __name__ == "__main__":
32     start_server()
```

```
ClientFileTransfer.py
1 import socket
2 def request_file(filename, host="127.0.0.1", port=12345):
3     client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
4     client_socket.connect((host, port))
5     print(f"Connected to server at {host}:{port}")
6     client_socket.sendall(filename.encode())
7     with open(f"received_{filename}", "wb") as file:
8         while True:
9             data = client_socket.recv(1024)
10             if not data:
11                 break
12             file.write(data)
13
14     print(f"File '{filename}' received successfully.")
15     client_socket.close()
16
17 file_to_request = input("Enter the filename to download: ")
18 request_file(file_to_request)
```

```
PS C:\Users\ASUS\OneDrive> cd C:\Users\ASUS\OneDrive\Local\Microsoft\WindowsApps\python3.11.exe
PS C:\Users\ASUS\OneDrive> c:\Users\ASUS\OneDrive\Local\Microsoft\WindowsApps\python3.11.exe c:/Users/ASUS/OneDrive/ServerFileTransfer.py
Enter the filename to download: sample.txt
Connected to server at 127.0.0.1:12345
File 'sample.txt' received successfully.
PS C:\Users\ASUS\OneDrive> c:\Users\ASUS\OneDrive\Local\Microsoft\WindowsApps\python3.11.exe c:/Users/ASUS/OneDrive/ClientFileTransfer.py
Enter the filename to download: Akash.txt
Connected to server at 127.0.0.1:12345
File 'Akash.txt' received successfully.
PS C:\Users\ASUS\OneDrive> c:\Users\ASUS\OneDrive\Local\Microsoft\WindowsApps\python3.11.exe c:/Users/ASUS/OneDrive/ServerFileTransfer.py
Enter the filename to download: ComputerNetworks.txt
Connected to server at 127.0.0.1:12345
File 'ComputerNetworks.txt' received successfully.
PS C:\Users\ASUS\OneDrive>
```