

## LAB 2

I. Mục đích .....	2
II. Yêu cầu .....	2
III. Hướng dẫn .....	2
III.1. Tạo một chương trình server đơn giản .....	2
III.1.1. Hướng dẫn lập trình .....	2
III.1.2. Test chương trình: .....	3
III.1.3. Trả lời câu hỏi: .....	4
III.2. Tạo chương trình Server đơn giản, khi Client kết nối vào thì gửi câu chào cho client .....	5
III.2.1. Hướng dẫn lập trình .....	5
III.2.2. Test chương trình .....	5
III.2.3. Trả lời câu hỏi: .....	7
III.3. Xây dựng chương trình client đơn giản .....	7
III.3.1. Hướng dẫn lập trình .....	7
III.3.2. Test chương trình .....	8
III.3.3. Trả lời câu hỏi: .....	9
III.4. Cải tiến chương trình client .....	11
III.4.1. Hướng dẫn lập trình .....	11
III.4.2. Test chương trình .....	11
III.5. Thêm khả năng gửi dữ liệu lên server và nhận lại dữ liệu từ server gửi về .....	12
III.5.1. Hướng dẫn lập trình .....	12
III.6. Thêm khả năng nhận dữ liệu từ client gửi lên và gửi trả dữ liệu ngược lại client .....	12
III.6.1. Hướng dẫn lập trình .....	12
III.6.2. Test chương trình .....	12
III.6.3. Trả lời câu hỏi: .....	14
IV. Bài tập .....	15

## **I. Mục đích**

Ban đầu làm quen với lập trình Socket

Hiểu được cách hoạt động của bộ đệm hệ thống và bộ đệm chương trình

Xây dựng chương trình client-server đơn giản

## **II. Yêu cầu**

Xây dựng các chương trình client-server từ mức đơn giản nhất rồi phức tạp dần

## **III. Hướng dẫn**

### **III.1. Tạo một chương trình server đơn giản**

#### **III.1.1. Hướng dẫn lập trình**

Để lập trình socket ta sử dụng 2 namespace:

```
using System.Net;  
using System.Net.Sockets;
```

Tạo Server EndPoint, EndPoint này sẽ tham chiếu đến địa chỉ IP và Port của Server:

```
IPEndPoint serverEndPoint = new IPEndPoint(IPAddress.Any, 5000);
```

Tạo Server Socket, Socket này sẽ được kết nối với Server EndPoint:

```
Socket serverSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream,  
ProtocolType.Tcp);
```

Lắng nghe kết nối trên Server Socket, khi có client kết nối tới sẽ trả về Socket để trao đổi dữ liệu với client:

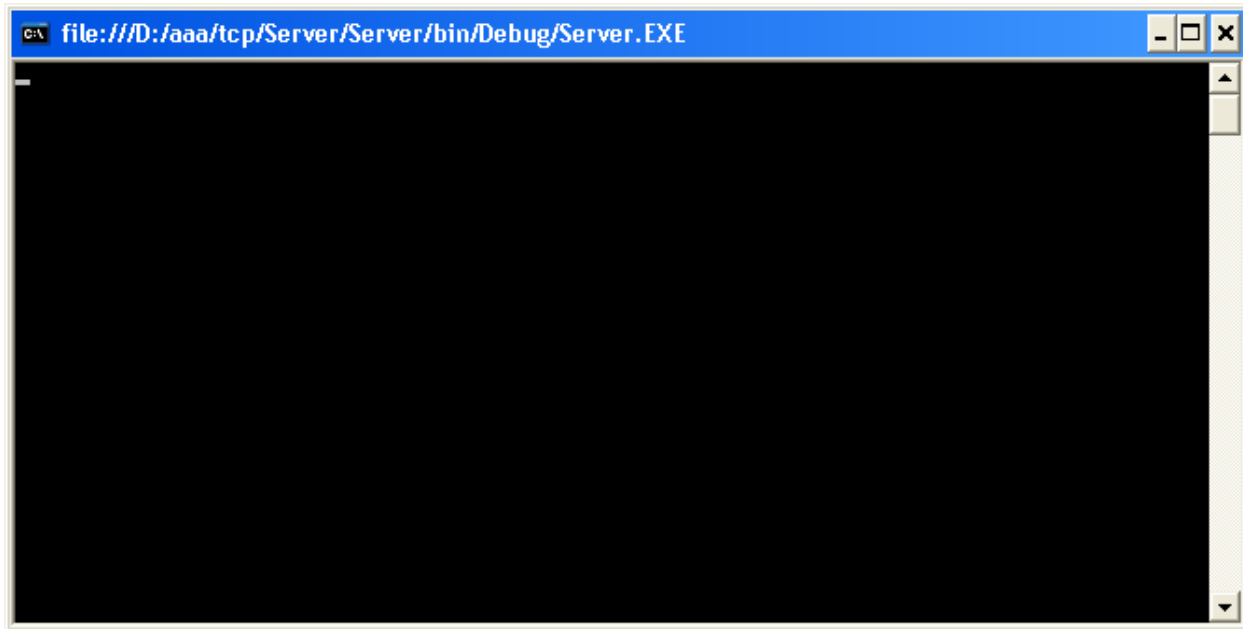
```
serverSocket.Listen(10);  
Socket clientSocket = serverSocket.Accept();
```

Khi client kết nối tới trên server sẽ xuất thông tin của client kết nối tới (địa chỉ IP, Port)

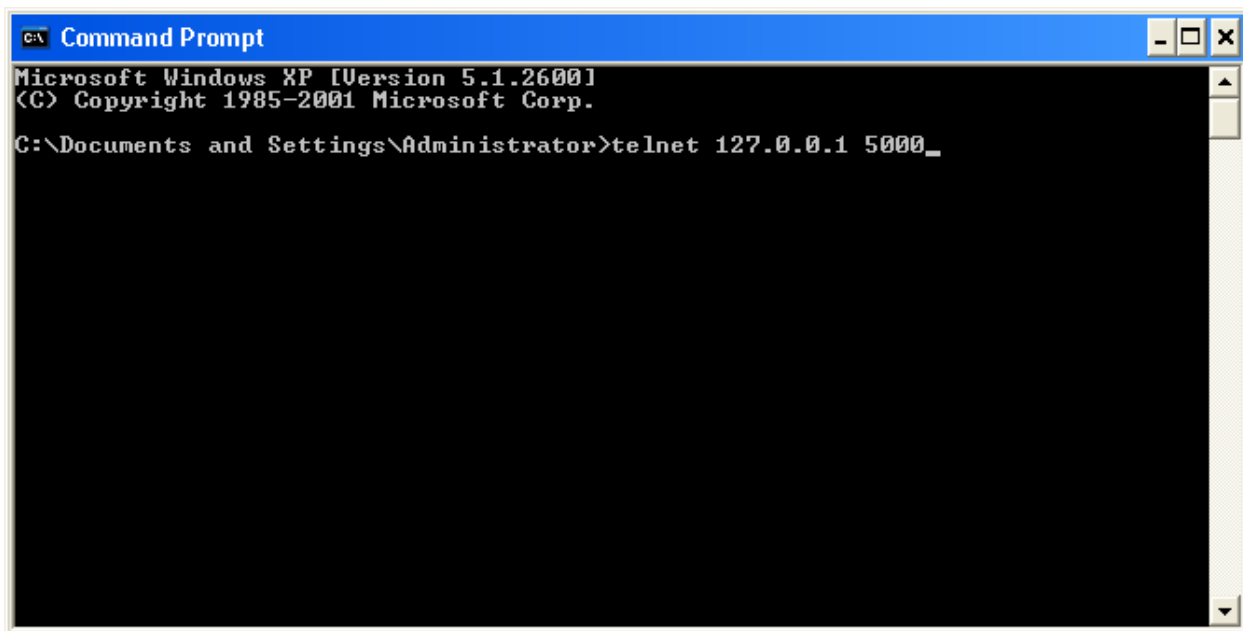
```
EndPoint clientEndPoint = clientSocket.RemoteEndPoint;  
Console.WriteLine(clientEndPoint.ToString());
```

### III.1.2. Test chương trình:

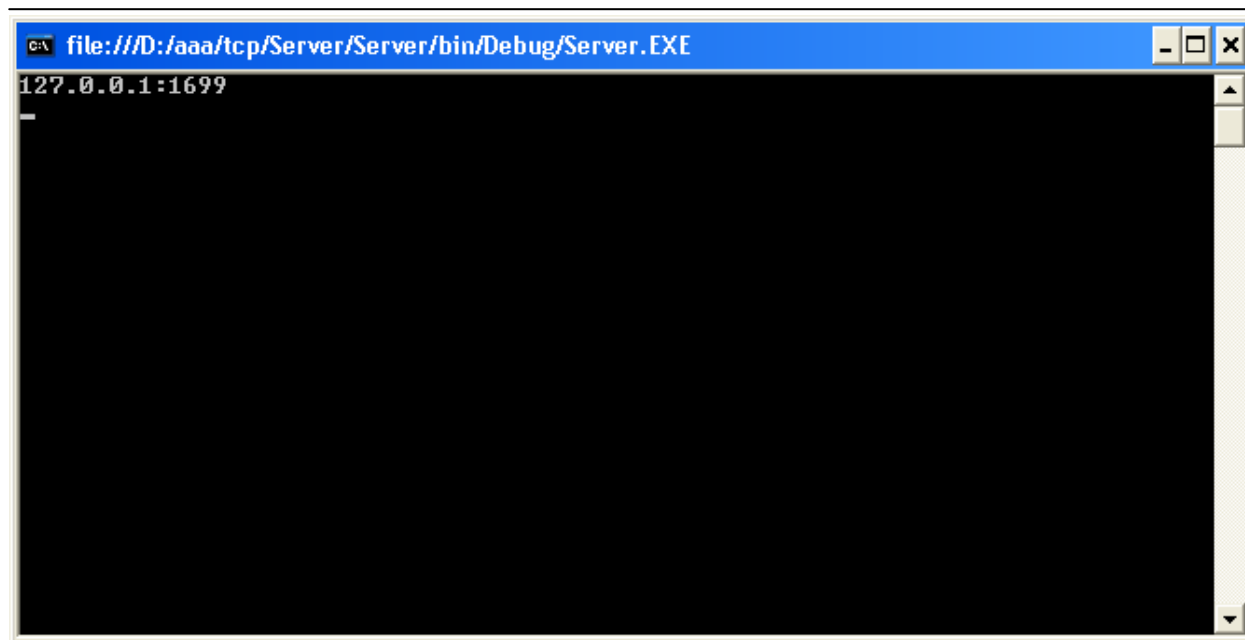
Chạy chương trình Server:



Chạy chương trình CMD và telnet vào server trên với địa chỉ IP loopback và port 5000:



Kết quả trả về trên Server:



### III.1.3. Trả lời câu hỏi:

- 1) Khi telnet vào Server bằng port 5000 tại sao port client dùng để kết nối vào lại không phải là port 5000 ?

*Vì client kết nối đến, hệ điều hành sẽ tự động cấp phát một port ngẫu nhiên cho client để thực hiện kết nối. Port này khác với 5000 để tránh xung đột (nếu client cũng dùng 5000 thì sẽ trùng với server).....*

- 2) Như hình trên, giải thích vì sao có client dùng port giá trị 1699 ?

*Con số 1699 là một port ngẫu nhiên mà hệ điều hành gán cho tiến trình client tại thời điểm đó. Mỗi lần client mở kết nối, OS chọn một port tự do trong khoảng 1024-65535. Vì vậy nó có thể là 1699, 50951, 30000....tùy lúc.....*

- 3) Có phải lúc nào client cũng mở port 1699 để kết nối với Server không ?

*Port client luôn thay đổi, mỗi lần chạy sẽ khác nhau. Nó phụ thuộc vào OS chọn ngẫu nhiên trong dải port ngẫu nhiên. Vì vậy không cố định 1699.*

- 4) Ta có thể chạy 2 chương trình CMD để telnet vào Server trên được không ? Giải thích.

*Server đã Listen(10) nghĩa là nó chấp nhận tối đa 10 kết nối đồng thời trong hàng đợi. Mỗi client kết nối sẽ được cấp một socket riêng với một port client riêng. Do đó có thể mở 2 CMD. (hoặc nhiều hơn). để telnet đến cùng một server port 5000.....*

## **III.2. Tạo chương trình Server đơn giản, khi Client kết nối vào thì gửi câu chào cho client**

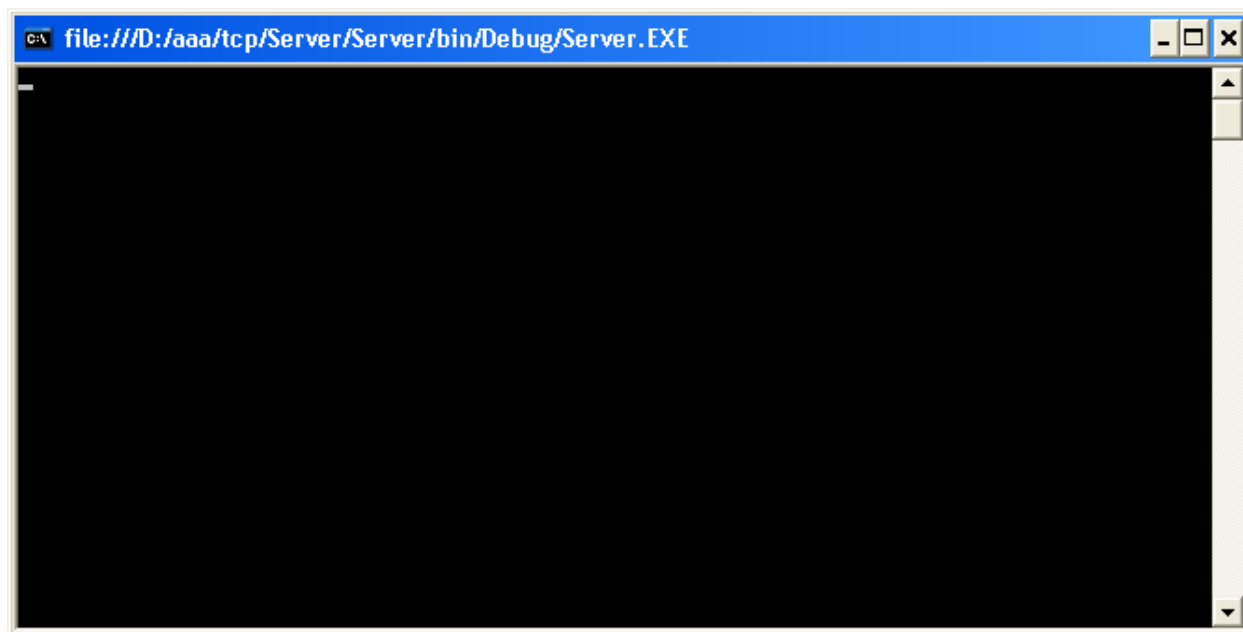
### **III.2.1. Hướng dẫn lập trình**

Sử dụng chương trình Server đơn giản ở trên, thêm phần gửi dữ liệu xuống client

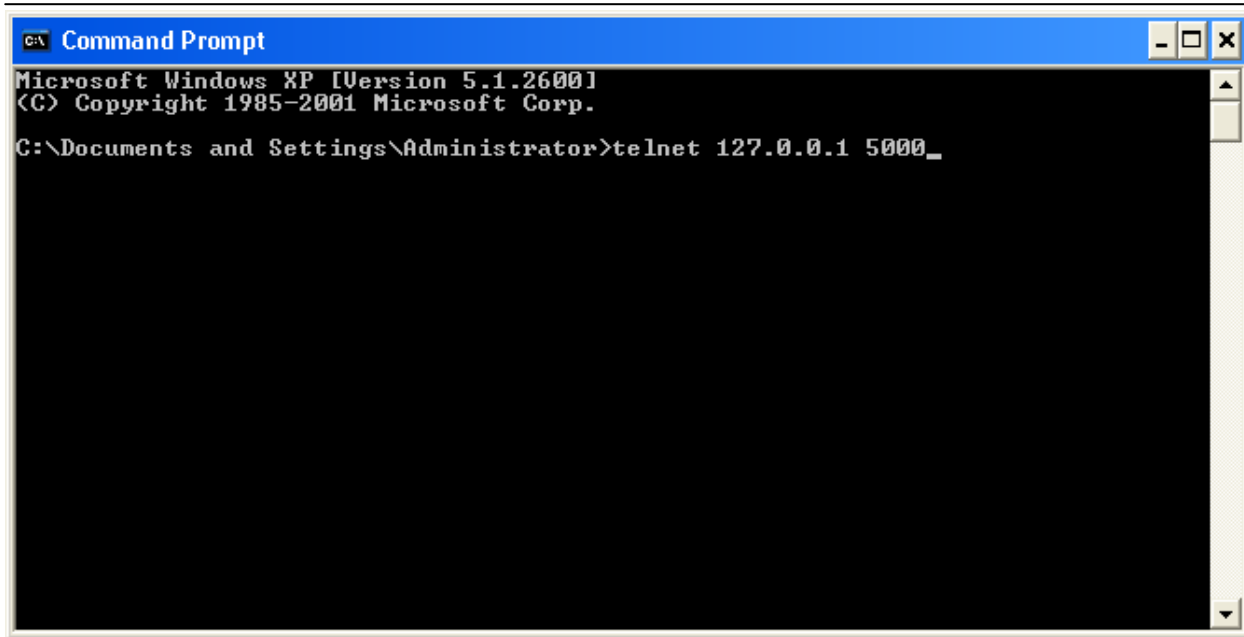
```
byte[] buff;  
string hello = "Hello Client";  
buff = Encoding.ASCII.GetBytes(hello);  
clientSocket.Send(buff, 0, buff.Length, SocketFlags.None);
```

### **III.2.2. Test chương trình**

Chạy chương trình Server:

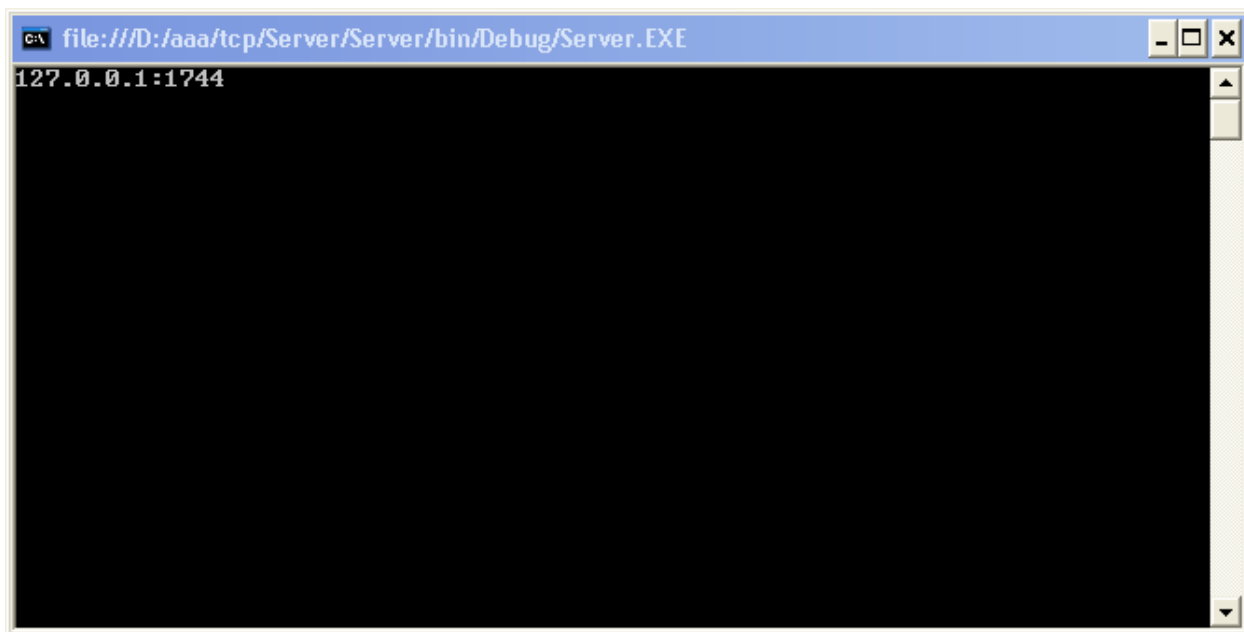


Chạy chương trình CMD và telnet vào server trên với địa chỉ IP loopback và port 5000:



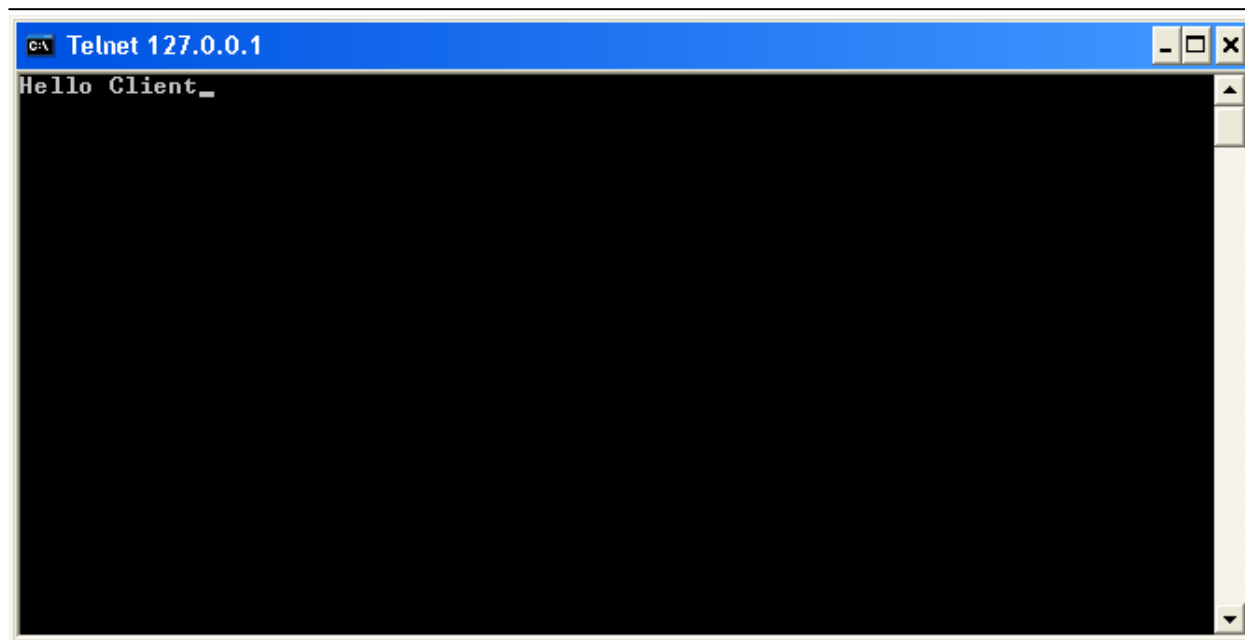
```
C:\ Command Prompt
Microsoft Windows XP [Version 5.1.2600]
Copyright 1985-2001 Microsoft Corp.
C:\Documents and Settings\Administrator>telnet 127.0.0.1 5000_
```

Kết quả trả về trên Server:



```
file:///D:/aaa/tcp/Server/Server/bin/Debug/Server.EXE
127.0.0.1:1744
```

Kết quả trả về trên client :



### III.2.3. Trả lời câu hỏi:

- 1) Tạo sao trong đoạn code viết thêm vào ta không cần phải khởi tạo độ lớn ban đầu cho buffer ?

*Encoding.ASCII.GetBytes(hello) sẽ tự động tạo ra một mảng byte đúng bằng độ dài chuỗi "Hello Client". Do đó không cần phải khai báo trước buff = new byte[1024]; như khi làm việc với dữ liệu nhận (Receive).*

- 2) Giải thích ý nghĩa các tham số của phương thức clientSocket.Send .

*buff: mảng type chứa dữ liệu cần gửi. 0: vị trí bắt đầu trong mảng buff (offset), buff.Length: số byte cần gửi (độ dài của dữ liệu). SocketFlags.None: cờ điều khiển việc gửi. None nghĩa là không dùng cờ đặc biệt (gửi bình thường).*

### III.3. Xây dựng chương trình client đơn giản

#### III.3.1. Hướng dẫn lập trình

Để lập trình socket ta using 2 namespace:

```
using System.Net;
using System.Net.Sockets;
```

Tạo Server EndPoint, EndPoint này sẽ tham chiếu đến địa chỉ IP và Port của Server:

```
IPEndPoint serverEndPoint = new IPEndPoint(IPAddress.Loopback, 5000);
```

Tạo socket dùng để kết nối với server:

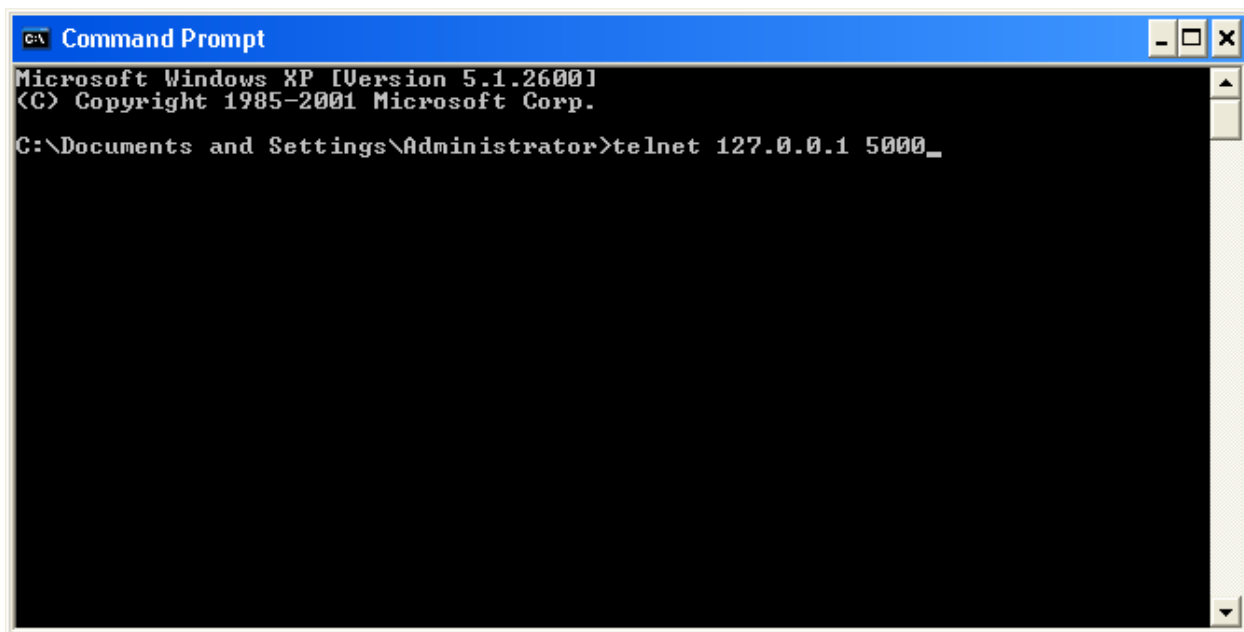
```
Socket serverSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);
Console.WriteLine("Dang ket noi voi server...");
serverSocket.Connect(serverEndPoint);
```

Nếu kết nối thành công, nhận câu chào từ server và hiển thị câu chào lên màn hình:

```
if (serverSocket.Connected)
{
    Console.WriteLine("Ket noi thanh cong voi server ...");
    byteReceive = serverSocket.Receive(buff, 0, buff.Length, SocketFlags.None);
    str = Encoding.ASCII.GetString(buff, 0, byteReceive)
    Console.WriteLine(str);
}
```

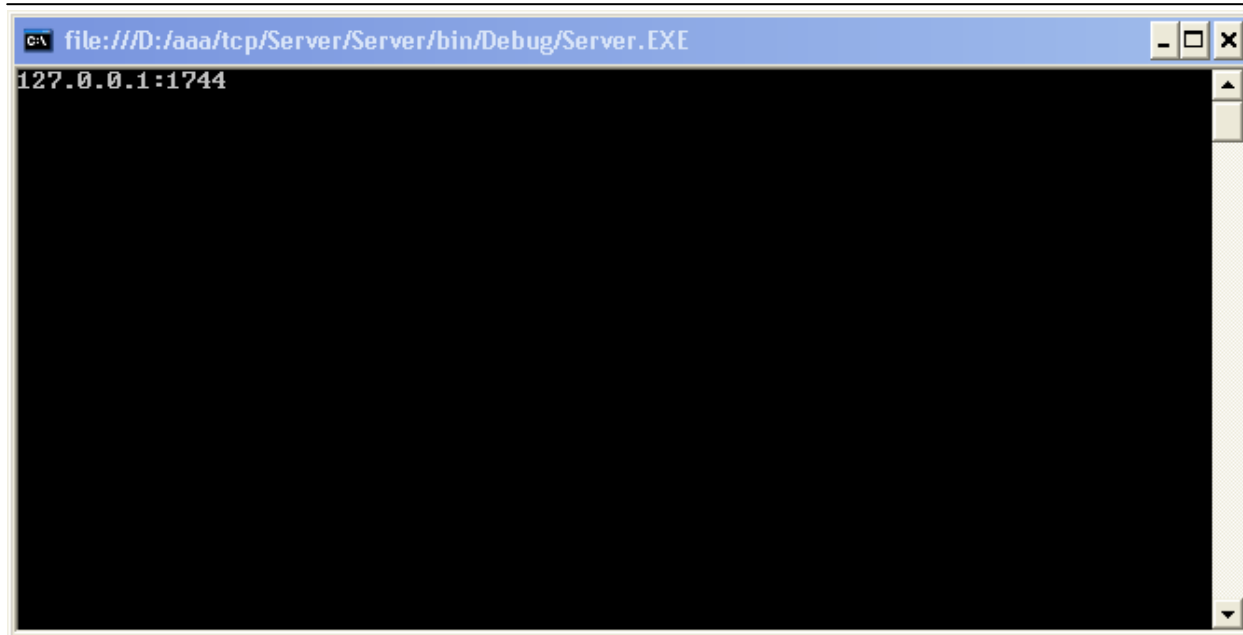
### III.3.2. Test chương trình

Chạy chương trình Server:

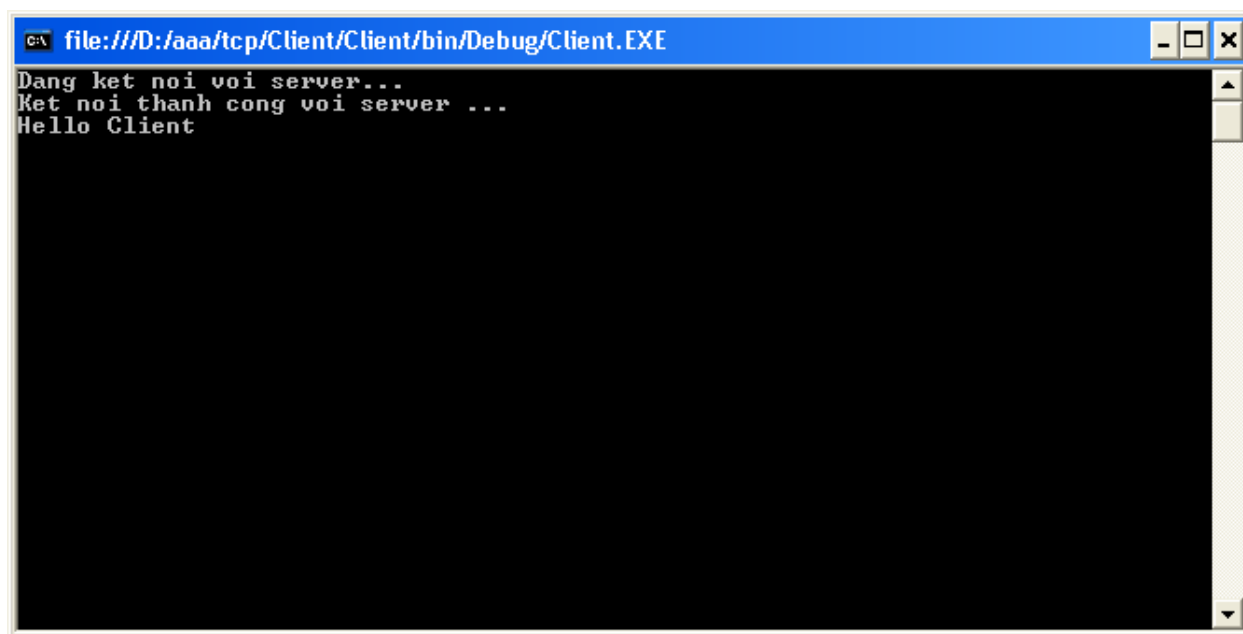


Kết quả trả về trên Server:





Kết quả trả về trên client :



### III.3.3. Trả lời câu hỏi:

- 1) Khi client kết nối đến server mà server chưa chạy hoặc là đường truyền mạng bị rớt thì có hiện tượng gì xảy ra ?, giải thích.

.....

.....

2) Từ đoạn code ở chương trình client:

```
str = Encoding.ASCII.GetString(buff, 0, byteReceive)
```

Nếu ta thay byteReceive bằng buff.Length thì kết quả sẽ như thế nào? Giải thích.

.....

.....

### III.4. Cải tiến chương trình client

#### III.4.1. Hướng dẫn lập trình

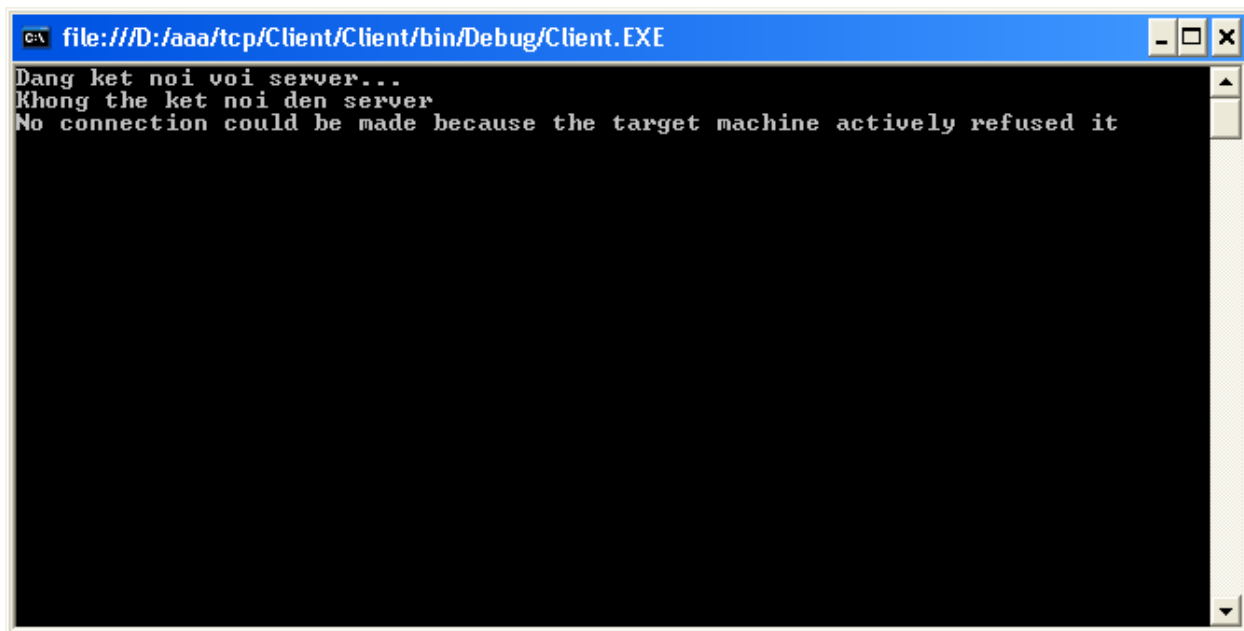
Chương trình client này khi kết nối đến server mà server chưa bật hoặc đường truyền bị lỗi thì chương trình sẽ bị lỗi.

Sửa lại chương trình bằng cách thêm try-catch vào:

```
try
{
    serverSocket.Connect(serverEndPoint);
}
catch (SocketException se)
{
    Console.WriteLine("Khong the ket noi den server");
    return;
}
```

#### III.4.2. Test chương trình

Chạy chương trình ở client mà không chạy server:



### **III.5. Thêm khả năng gửi dữ liệu lên server và nhận lại dữ liệu từ server gửi về**

#### **III.5.1. Hướng dẫn lập trình**

Việc này được thực hiện bằng cách thêm vòng lặp vô hạn

```
while (true)
{
    str = Console.ReadLine();
    buff = Encoding.ASCII.GetBytes(str);
    serverSocket.Send(buff, 0, buff.Length, SocketFlags.None);
    buff = new byte[1024];
    byteReceive = serverSocket.Receive(buff, 0, buff.Length, SocketFlags.None);
    str = Encoding.ASCII.GetString(buff, 0, byteReceive);
    Console.WriteLine(str);
}
```

### **III.6. Thêm khả năng nhận dữ liệu từ client gửi lên và gửi trả dữ liệu ngược lại client**

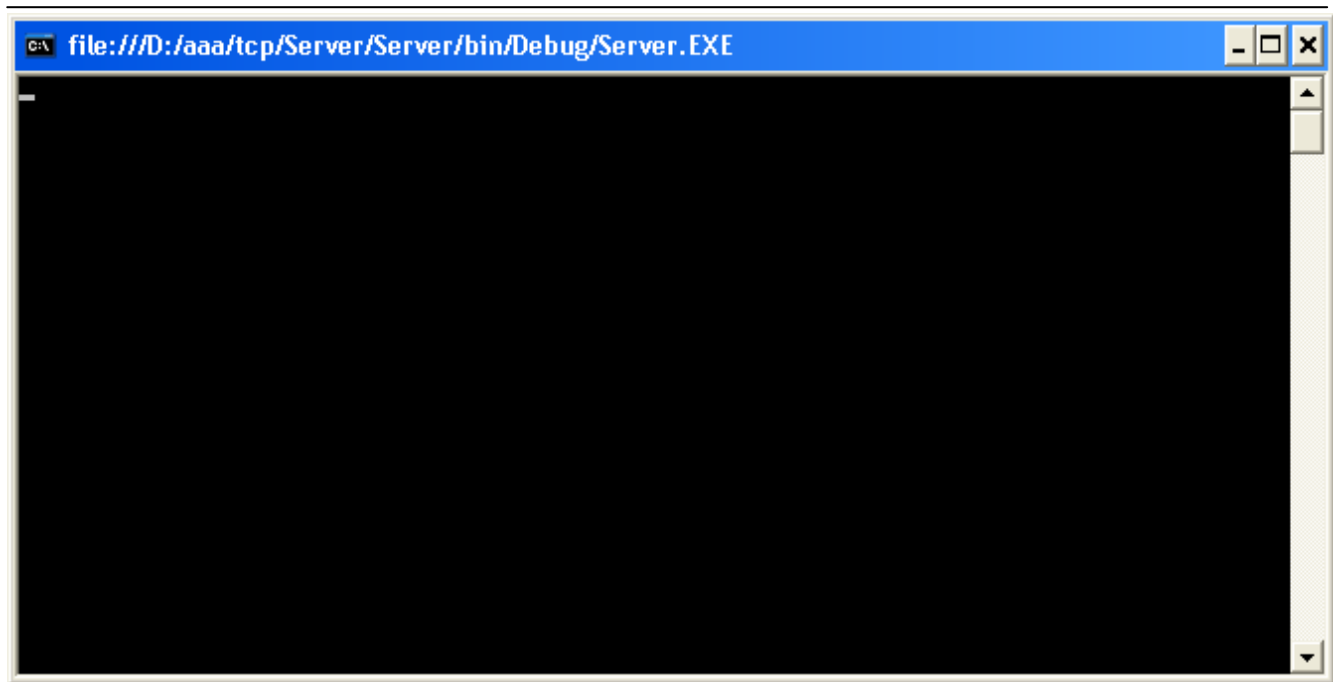
#### **III.6.1. Hướng dẫn lập trình**

Việc này được thực hiện bằng cách thêm vòng lặp vô hạn

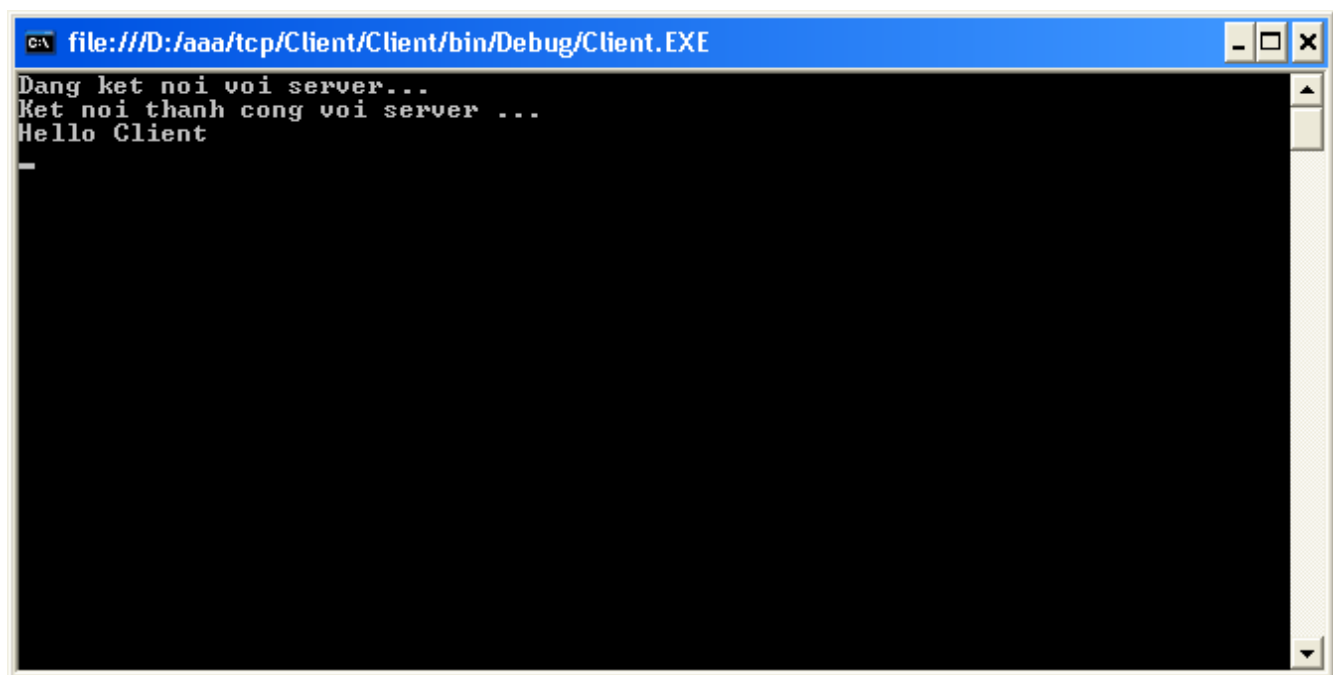
```
while (true)
{
    buff = new byte[1024];
    byteReceive = clientSocket.Receive(buff, 0, buff.Length, SocketFlags.None);
    str = Encoding.ASCII.GetString(buff, 0, byteReceive);
    Console.WriteLine(str);
    clientSocket.Send(buff, 0, byteReceive, SocketFlags.None);
}
```

#### **III.6.2. Test chương trình**

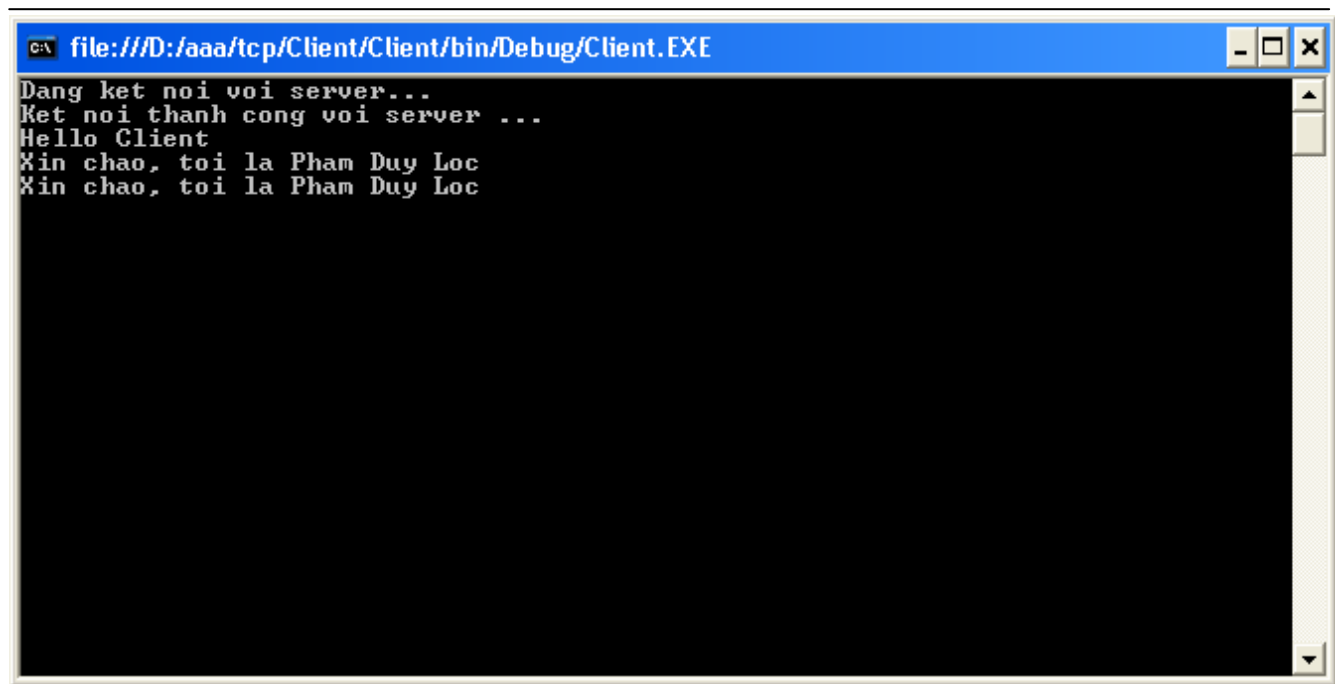
Chạy chương trình server:



Chạy chương trình client:

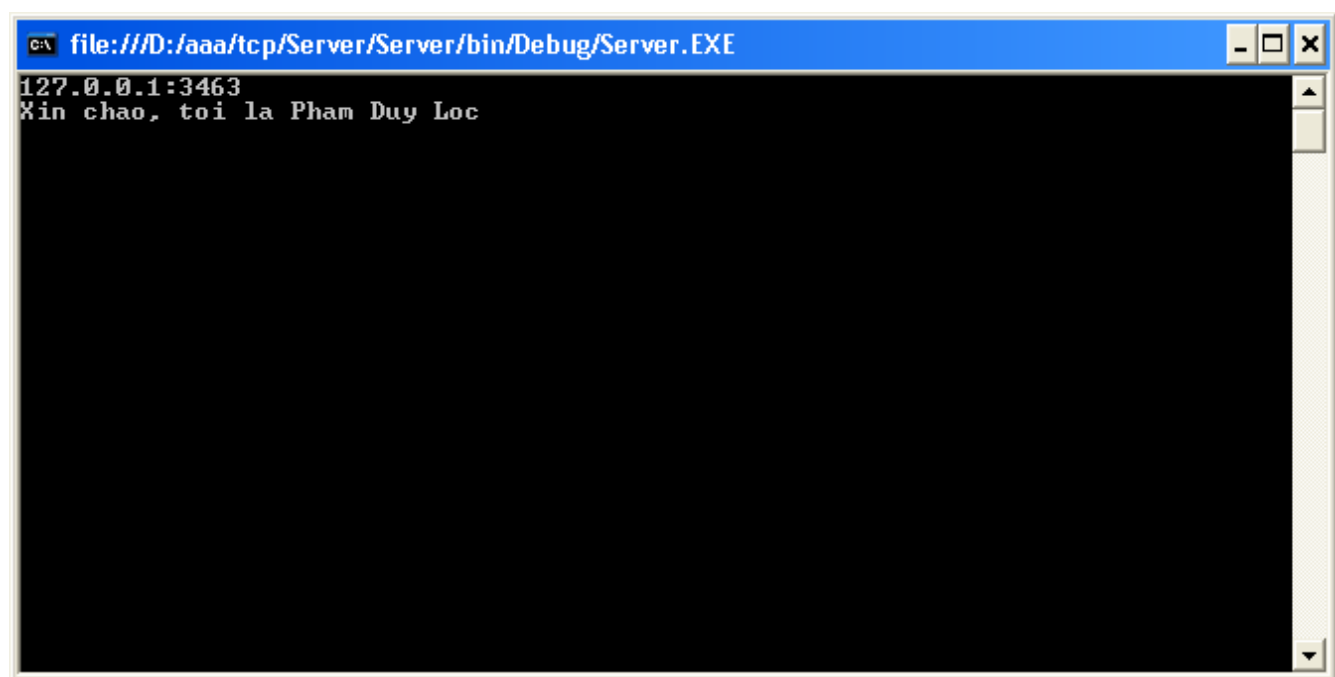


Nhập dữ liệu và gửi lên server:



```
file:///D:/aaa/tcp/Client/Client/bin/Debug/Client.EXE
Dang ket noi voi server...
Ket noi thanh cong voi server ...
Hello Client
Xin chao, toi la Pham Duy Loc
Xin chao, toi la Pham Duy Loc
```

Kết quả hiển thị trên server:



```
file:///D:/aaa/tcp/Server/Server/bin/Debug/Server.EXE
127.0.0.1:3463
Xin chao, toi la Pham Duy Loc
```

### III.6.3. Trả lời câu hỏi:

- 1) Khi thay đổi kích thước bộ đệm từ 1024 xuống 10 thì có hiện tượng gì xảy ra ?. Giải thích

.....

.....

2) Nếu ta không reset lại bộ đếm thì có hiện tượng gì xảy ra? Giải thích.

.....

.....

#### **IV. Bài tập**

- 1) Chương trình trên khi tắt client thì trên server sẽ bị lỗi, hãy cải tiến lại chương để khi client tắt đột ngột trên server sẽ thông báo client đã ngắt kết nối
- 2) Cải tiến chương trình: khi client nhập chữ “exit” thì sẽ tắt client, trên server thông báo client đã ngắt kết nối
- 3) Viết chương trình thực hiện các phép tính số học, việc thực hiện tính toán được thực hiện trên server, phía client sẽ nhập vào các toán hạng và các phép tính