Socket编程实验附录

目录

基于python的可视化版本

一、系统概述

运行环境

python

编译

pycharm自带的对于Python程序的编译方式

使用方法

先运行server.py程序，后运行client.py程序

实现环境

pycharm编程环境

程序文件列表

- socket_GUI
  - client.py
  - server.p

二、主要数据结构

```
self.s = None            // 套接字
self.address = None      // 服务端地址和端口
self.status              // 服务器和客户端连接状态
...                      // 此处省略GUI的诸多静态标签和按键布局
```

三、主要算法描述

1. 初始化套接字，并且尝试连接

服务器端：

- 利用socket库，创建一个套接字，对其指定协议类型
- 绑定访问的端口以及ip地址。
- 开启监听模式，等待用户端连接。

用户端：

- 利用socket库，创建一个套接字，对其指定协议类型
- 绑定访问的端口以及ip地址。
- 尝试连接，设置异常处理，没有连接到服务器端时抛出异常。

2. 服务器实现连接

服务器端：

- 连接成功后，返回客户端地址和一个新的 socket 连接

用户端：

- 学习

3. 服务器和用户端数据传输

服务器端：

- 开始监听，用户端发来的消息。
- 收到消息后，输入需要给用户端返回的消息。当消息为空的时候，不能发送。
- ** 注意：py3中数据传输的格式需要进行编码格式的转换。

用户端：

- 输入需要给用户端返回的消息。当消息为空的时候，不能发送。
- 开始监听，服务器端发来的消息。
- ** 注意：py3中数据传输的格式需要进行编码格式的转换。

基于C++的非可视化版本

一、系统概述

运行环境

C++

编译

利用visual studio2019编译

使用方法

先运行server.sln工程文件，后运行client.sln工程文件

实现环境

visual studio2019

程序文件列表

- TCP
  - Client
    - Client
    - Debug
    - Client.sln
  - Server
    - Server
    - Debug
    - Server.sln
- UDP
  - Client
    - Client
    - Debug
    - Client.sln
  - Server
    - Server
    - Debug
    - Server.sln

二、主要数据结构

```
SOCKET sclient = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP); // AF_INET :ipv4, SOCK_STREAM: socket类型, IPPROTO_TCP: TCP协议
serAddr.sin_family = AF_INET;       // 地址族
serAddr.sin_port = htons(8888);     // 端口号，高低次序转换
```

三、主要算法描述

1. 初始化套接字，并且尝试连接

服务器端：

- 利用socket库，创建一个套接字，对其指定协议类型
- 绑定访问的端口以及ip地址。
- 开启监听模式，等待用户端连接。

用户端：

- 利用socket库，创建一个套接字，对其指定协议类型
- 绑定访问的端口以及ip地址。
- 尝试连接，设置异常处理，没有连接到服务器端时抛出异常。

2. 服务器实现连接

服务器端：

- 连接成功后，返回客户端地址和一个新的 socket 连接

用户端：

- 学习

3. 服务器和用户端数据传输

服务器端：

- 开始监听，用户端发来的消息。
- 收到消息后，输入需要给用户端返回的消息。当消息为空的时候，不能发送。
- ** 注意：C++中数据传输的格式需要进行编码格式的转换。

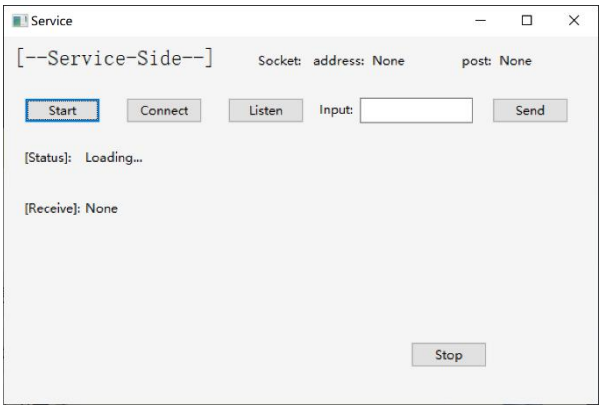
用户端：

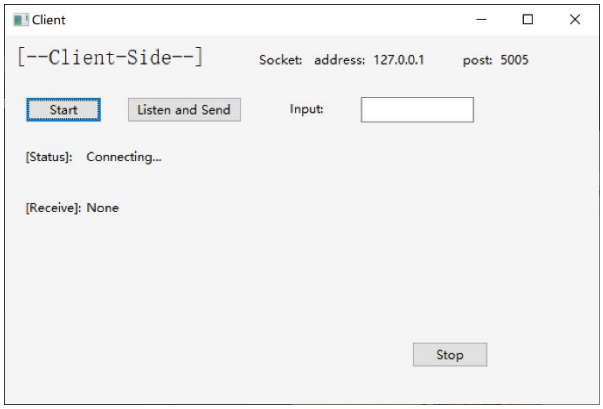
- 输入需要给用户端返回的消息。当消息为空的时候，不能发送。
- 开始监听，服务器端发来的消息。
- ** 注意：C++中数据传输的格式需要进行编码格式的转换。

用户使用文档

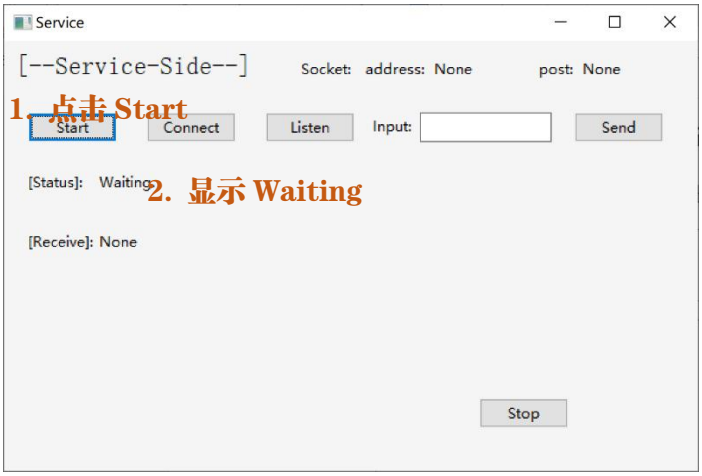
I. 基于 Python 的 GUI 可视化程序

1. 依次运行 server.py 和 client.py 出现如下窗口:

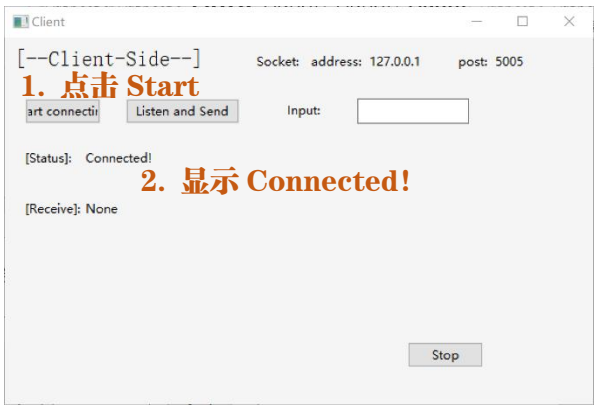




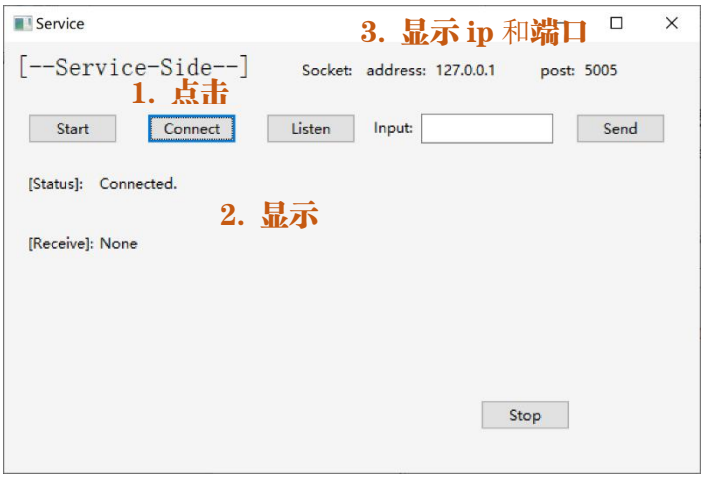
2. 点击 Service 窗口中的 Start 按钮，开始连接。[Status] 显示【waiting】状态。



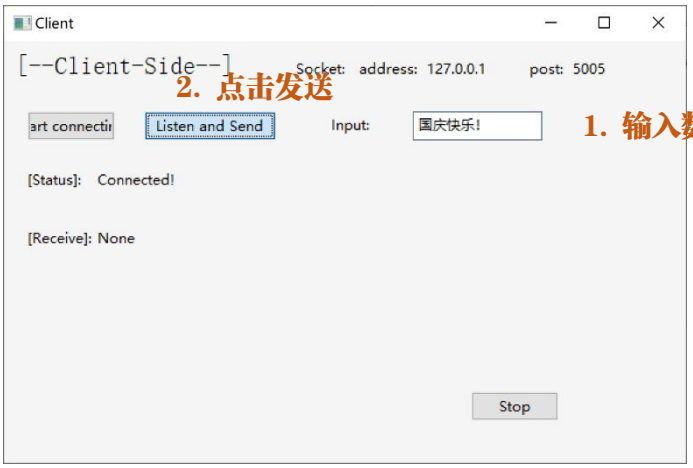
3. 点击 Client 窗口中的 Start 按钮，开始连接。
若成功 [Status] 显示【Connected!】状态; 若不成功; [Status] 显示【Connected!】状态。



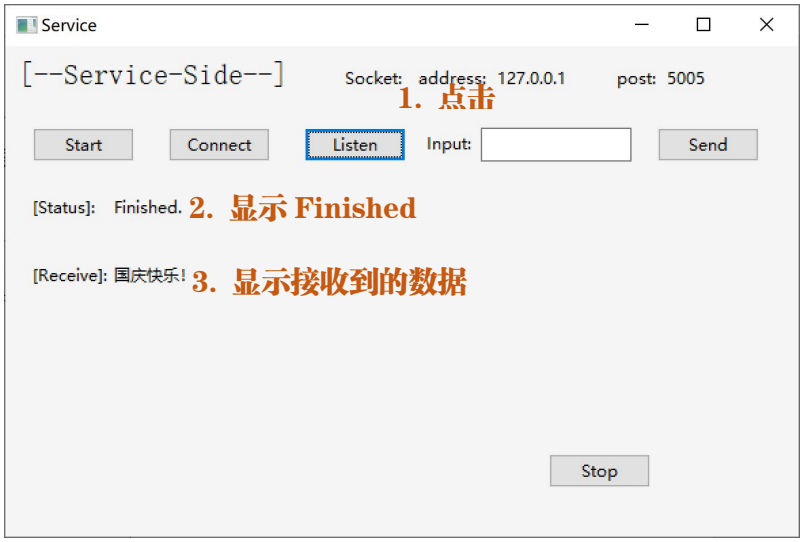
4. 点击 Service 窗口中的 Connect 按钮，开始连接。[Status] 显示【waiting】状态。Socket: address 和 post 显示连接的 ip 地址和端口号。



5. 在 Client 窗口的 Input 输入框中，输入想要发送的数据信息，比如:“国庆快乐!”，然后点击 Listen and Send 按钮。

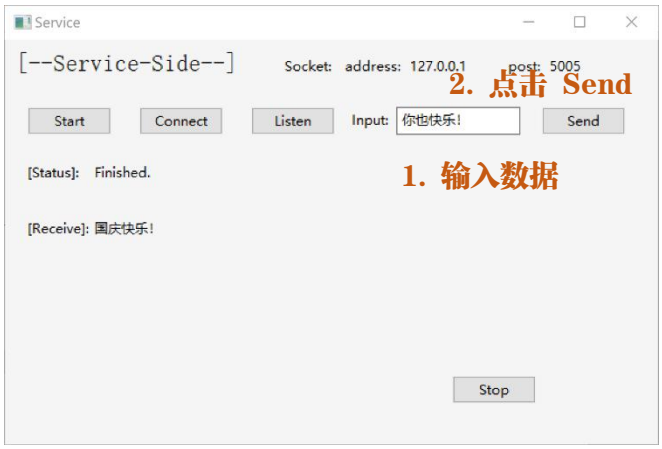


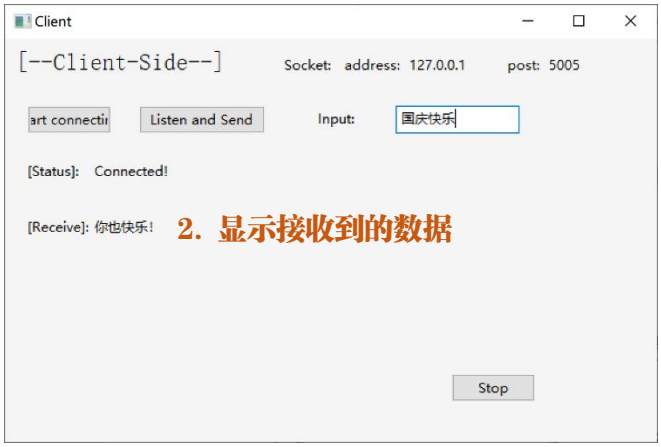
6. 在 Server 窗口，点击 Listen 按钮，接收用户端发送的数据。[Status] 显示【Finished】。[Receive]显示接收到的数据。



7.在 Service 窗口中的 Input 输入框中，输入想要发送的数据信息，比如:“你也快乐!”。点击 send 按钮。

在 Client 窗口可以接收到服务端发送的数据。





II. 基于 C++ 的非可视化程序

1. 依次打开 Server.sln 和 Client.sln。

   在 Client 窗口输入想要传输的数据，在 Server 窗口会显示接收到的数据。

   在 Server 窗口输入想要传输的数据，在 Client 窗口会显示接收到的数据。