简单客户端服务器模型（C++、python 和 go 语言示例）

工作中用到了 C/S 模型，所做的也无非是给服务器发数据，但开发阶段会遇到程序自身的回环测试，需要用到简单的服务端以便验证数据发送的正确性。

写软件用 C++，跑测试用 python，这段时间也刚好看 go 语言，所以都要有 demo。以下三组程序实现的功能相同，这里一起做下总结。

一、C++实现

Boost.Asio 是一个跨平台的 C++库，它用现代 C++方法为网络和底层 I/O 程序提供了一致的异步 I/O 模型。 为了跨平台，我用 boost 库实现，具体如下。

服务端代码：

```
/*
    File      : svr.cpp
    Author    : Mike
    E-Mail    : Mike_Zhang@live.com
*/

#include <iostream>
#include <boost/asio.hpp>

using boost::asio::ip::tcp;
enum {max_length = 1024};

typedef boost::shared_ptr<tcp::socket> socket_ptr;

int main()
{
        boost::asio::io_service io_service;
        tcp::acceptor a(io_service, tcp::endpoint(tcp::v4(), atoi("12345")));
        for (;;)
        {
                socket_ptr sock(new tcp::socket(io_service));
                a.accept(*sock);
                char data[max_length];
                boost::system::error_code error;
                size_t length = sock->read_some(boost::asio::buffer(data), error);
                data[length] = 0;
                std::cout<<data<<std::endl;
                sock->close();
        }
        return 0;
}
```

客户端代码：

```
/*
    File      : cli.cpp
    Author    : Mike
    E-Mail    : Mike_Zhang@live.com
*/
#include <iostream>
#include <boost/asio.hpp>

using boost::asio::ip::tcp;
enum { max_length = 1024 };
```

```
int main(int argc, char* argv[])
{
        boost::asio::io_service io_service;
        tcp::resolver resolver(io_service);
        tcp::resolver::query query(tcp::v4(), "127.0.0.1","12345");
        tcp::resolver::iterator iterator = resolver.resolve(query);

        tcp::socket s(io_service);
        s.connect(*iterator);

        std::cout << "Please input: ";
        char request[max_length];
        std::cin.getline(request, max_length);
        size_t request_length = strlen(request);
        boost::asio::write(s, boost::asio::buffer(request, request_length));
        return 0;
}
```

二、python实现

服务端代码：

```
'''
     File      : svr.py
     Author    : Mike
     E-Mail    : Mike_Zhang@live.com
'''
import socket,os
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.bind(('127.0.0.1', 12345))
sock.listen(5)
while True:
        connection,address = sock.accept()
        buf = connection.recv(1024)
        print buf
        connection.close()
```

客户端代码：

```
'''
     File      : cli.py
     Author    : Mike
     E-Mail    : Mike_Zhang@live.com
'''
import socket
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect(('127.0.0.1', 12345))
#sock.send('Test\n')
sock.send(raw_input("Please input : "))
sock.close()
```

三、go 语言实现

服务端代码：

```
/*
     File      : svr.go
     Author    : Mike
     E-Mail    : Mike_Zhang@live.com
*/
```

```
package main

import(
        "net"
        "fmt"
        "bufio"
)

func main() {
        client,err := net.Listen("tcp","127.0.0.1:12345")
        if err != nil {
                fmt.Printf("Error : %s\n",err.String())
        }
        for {
                if c, err := client.Accept(); err == nil {
                        defer c.Close()
                        line, _ := bufio.NewReader(c).ReadString('\n')
                        fmt.Println(line)
                }
        }
}
```

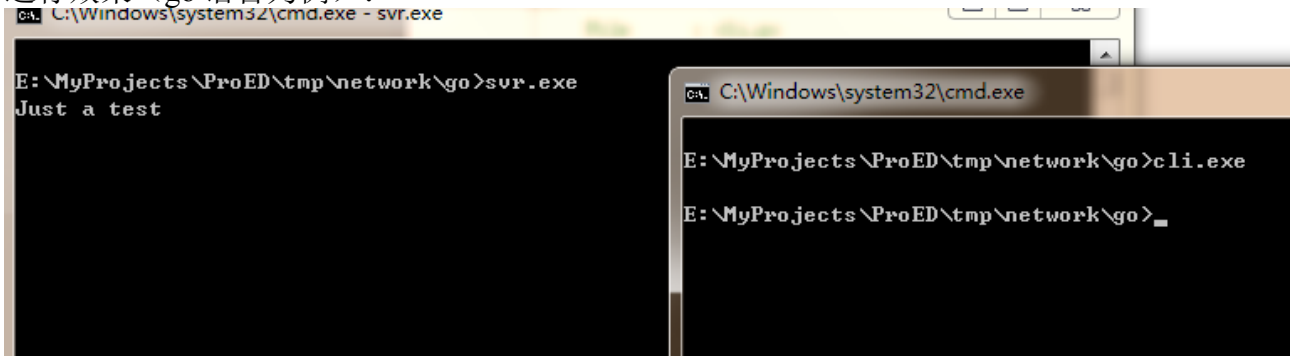
客户端代码：

```
/*
     File       : cli.go
     Author     : Mike
     E-Mail     : Mike_Zhang@live.com
*/
package main

import(
        "net"
        "fmt"
)

func main() {
        conn, err := net.Dial("tcp", "127.0.0.1:12345")
        if err != nil {
                fmt.Printf("Error : %s\n",err.String())
        }
        conn.Write([]byte("Just a test"))
}
```

运行效果（go 语言为例）：



好，就这些了，希望对你有帮助。