

数据通信作业-2

姓名: 费扬 学号: 519021910917 日期: 2022.04.18

一、 实验名称及内容

Assignment2: EchoClient&Server

使用 winsock 编程，主要目标任务为：

Task1: Write a stream based echoclient sending messages to the echo server, receiving each message returned by the server. Terminate the connection when "quit" is entered.

即编写一个客户端，接受用户数据，向服务器发送数据并显示服务器返回值。输入 quit 退出程序。

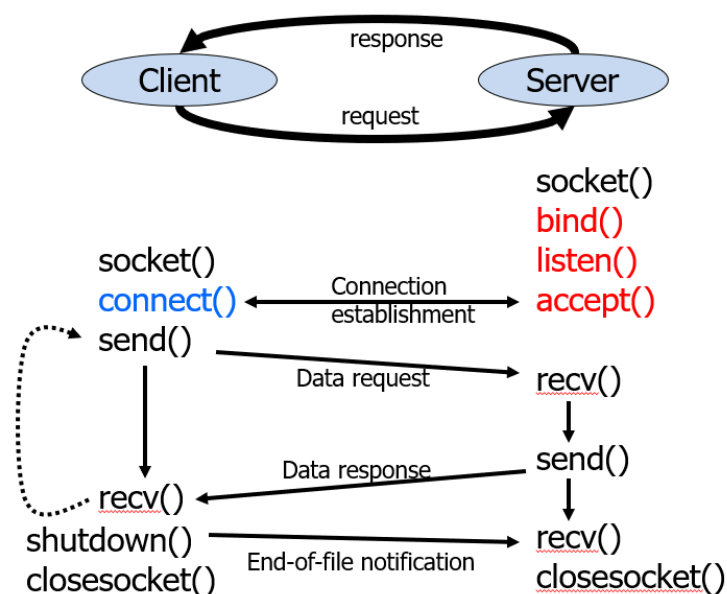
Task2: Write a stream based echoserver printing out the message received from the client, echoing it back, until the client closes the connection.

即编写一个服务器，接受用户的数据并将其发送回客户端。等待客户端关闭通信。

Task3: Modify your solution to Exercise2 to write a stream based echo server, which can simultaneously handle multiple clients connecting to it. No modification of the client code is necessary, but multiple instances of the client should be started. Hint: use Windows threads functions.

即修改上述的服务器使其能同时连接多个客户端，并使用多线程处理客户端数据。

二、 实验过程和结果



本次实验的基本架构图示

Task1:

基本操作步骤：基本框架类似 TCPClient 和 TCPServer，在其基础上修改一点点：

EchoClient:

1. Initialize Winsock.
2. Create a socket.
3. Connect to the server.
4. Send and receive data. (minor change)
5. Disconnect.

```
// Receive until the peer closes the connection
do {
    gets_s(sendbuf);
    if (strcmp(sendbuf, "quit") == 0)
        break;
    iResult = send(ConnectSocket, sendbuf, int(strlen(sendbuf) + 1), 0);
    printf("Bytes Sent: %ld\n", iResult);
    iResult = recv(ConnectSocket, recvbuf, recvbuflen, 0);
    printf("Received: \"%s\"\n", recvbuf);
    if (iResult > 0)
        printf("Bytes received: %d\n", iResult);
    else if (iResult == 0)
        printf("Connection closed\n");
    else
        printf("recv failed with error: %d\n", WSAGetLastError());
} while (iResult > 0);
```

有所变动的地方代码展示

EchoServer:

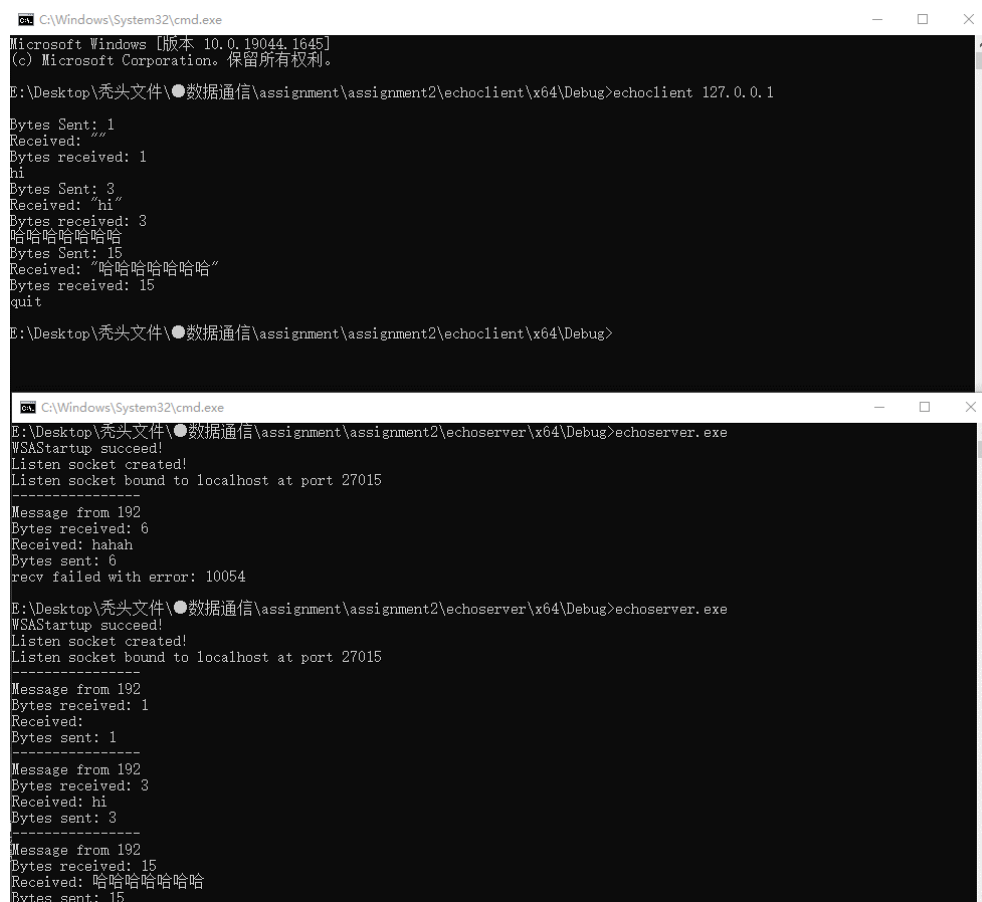
1. Initialize Winsock
2. Create a socket
3. Bind the socket
4. Listen on the socket for a client
5. Accept a connection from a client
6. Receive and send data (minor change)
7. Disconnect

```
do {
    iResult = recv(ClientSocket, recvbuf, recvbuflen, 0);
    if (iResult > 0) {
        printf("Bytes received: %d\n", iResult);
        // Echo the buffer back to the sender
        iSendResult = send(ClientSocket, recvbuf, iResult, 0);
        if (iSendResult == SOCKET_ERROR) {
            printf("send failed with error: %d\n", WSAGetLastError());
            closesocket(ClientSocket);
            WSACleanup();
            return 1;
        }
        printf("Bytes sent: %d\n", iSendResult);
    }
    else if (iResult == 0)
        printf("Connection closing...\n");
    else {
        printf("recv failed with error: %d\n", WSAGetLastError());
        closesocket(ClientSocket);
        WSACleanup();
        return 1;
    }
} while (iResult > 0);
```

有所变动的地方代码展示

代码见打包文件中 echoclient.cpp 和 echoserver.cpp，不予展示：

之后在命令行操作输出：



```
C:\Windows\System32\cmd.exe
Microsoft Windows [版本 10.0.19044.1645]
(c) Microsoft Corporation. 保留所有权利。

E:\Desktop\秃头文件\●数据通信\assignment\assignment2\echoclient\x64\Debug>echoclient 127.0.0.1

Bytes Sent: 1
Received: ""
Bytes received: 1
hi
Bytes Sent: 3
Received: "hi"
Bytes received: 3
哈哈哈哈哈
Bytes Sent: 15
Received: "哈哈哈哈哈"
Bytes received: 15
quit

E:\Desktop\秃头文件\●数据通信\assignment\assignment2\echoclient\x64\Debug>

C:\Windows\System32\cmd.exe
E:\Desktop\秃头文件\●数据通信\assignment\assignment2\echoserver\x64\Debug>echoserver.exe
WSAStartup: succeed!
Listen socket created!
Listen socket bound to localhost at port 27015
-----
Message from 192
Bytes received: 6
Received: hahah
Bytes sent: 6
recv failed with error: 10054

E:\Desktop\秃头文件\●数据通信\assignment\assignment2\echoserver\x64\Debug>echoserver.exe
WSAStartup: succeed!
Listen socket created!
Listen socket bound to localhost at port 27015
-----
Message from 192
Bytes received: 1
Received:
Bytes sent: 1
-----
Message from 192
Bytes received: 3
Received: hi
Bytes sent: 3
-----
Message from 192
Bytes received: 15
Received: 哈哈哈哈哈
Bytes sent: 15
```

这里可以看到首先打开服务器，建立连接，传输信息，中英文编码都无问题，最后使用 quit 退出即可。

Task2:

Tip: No modification of the client code is necessary, but multiple instances of the client should be started.

基本操作步骤：

1. 在 EchoServer 的基础上修改为 EchoServer2
2. Adding doService()
3. Initialize Winsock
4. Create a socket
5. Bind the socket
6. Listen on the socket for a client
7. Accept a connection from multiple clients
8. Receive and send data
9. Disconnect

```

void doService(void* ClientSocketParam) {
    char recvbuf[DEFAULT_BUFLEN];
    int recvbuflen = DEFAULT_BUFLEN;
    int iResult;
    int iSendResult;
    SOCKET ClientSocket = (SOCKET)ClientSocketParam;

    do {
        memset(recvbuf, 0, DEFAULT_BUFLEN);
        iResult = recv(ClientSocket, recvbuf, recvbuflen, 0);
        if (iResult > 0) {
            sockaddr_in info;
            int len = sizeof(info);
            getsockname(ClientSocket, reinterpret_cast<sockaddr*>(&info), &len);
            printf("-----\n");
            printf("Message from %d\n", ClientSocket);
            // printf("Message from %s %d\n", inet_ntoa(info.sin_addr), ntohs(info.sin_port));
            printf("Bytes received: %d\n", iResult);
            printf("Received: %s\n", recvbuf);

            // Echo the buffer back to the sender
            iSendResult = send(ClientSocket, recvbuf, iResult, 0);

            if (iSendResult == SOCKET_ERROR) {
                printf("send failed with error: %d\n", WSAGetLastError());
                closesocket(ClientSocket);
                WSACleanup();
                return;
            }
            printf("Bytes sent: %d\n", iSendResult);

            // handle quit
            if (strcmp(recvbuf, "quit") == 0) {
                printf("Connection has been quited!\n");
                break;
            }
        }
        else if (iResult == 0)
            printf("Connection closing...\n");
        else {
            printf("recv failed with error: %d\n", WSAGetLastError());
            closesocket(ClientSocket);
            WSACleanup();
            return;
        }
    }
}

```

进行了修改的地方：doService()

```

printf("Listen socket bound to localhost at port 27015\n");

freeaddrinfo(result);

iResult = listen(ListenSocket, SOMAXCONN);
if (iResult == SOCKET_ERROR) {
    printf("listen failed with error: %d\n", WSAGetLastError());
    closesocket(ListenSocket);
    WSACleanup();
    return 1;
}

// Accept a client socket
while (1) {
    ClientSocket = accept(ListenSocket, NULL, NULL);
    if (ClientSocket == INVALID_SOCKET) {
        printf("accept failed with error: %d\n", WSAGetLastError());
        closesocket(ListenSocket);
        WSACleanup();
        return 1;
    }
    _beginthread(doService, 2048, reinterpret_cast<void*>(ClientSocket));
}

// Receive until the peer shuts down the connection

WSACleanup();

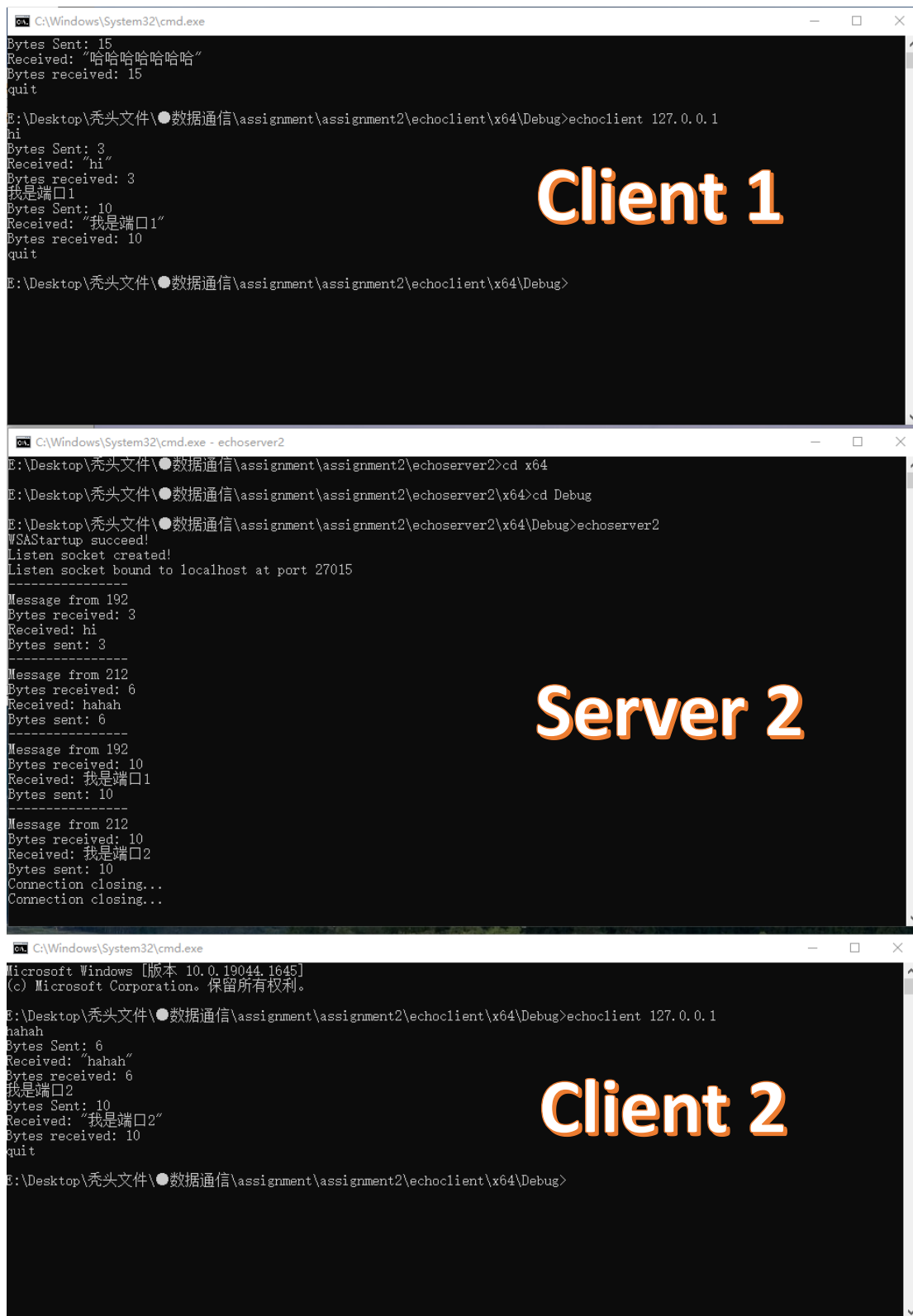
return 0;
}

```

允许多个 client

代码见打包文件中 echoserver2.cpp，不予展示：

之后在命令行操作输出如下：



```
C:\Windows\System32\cmd.exe
Bytes Sent: 15
Received: "哈哈哈哈哈"
Bytes received: 15
quit

E:\Desktop\秃头文件\数据通信\assignment\assignment2\echoclient\x64\Debug>echoclient 127.0.0.1
hi
Bytes Sent: 3
Received: "hi"
Bytes received: 3
我是端口1
Bytes Sent: 10
Received: "我是端口1"
Bytes received: 10
quit

E:\Desktop\秃头文件\数据通信\assignment\assignment2\echoclient\x64\Debug>

C:\Windows\System32\cmd.exe - echoserver2
E:\Desktop\秃头文件\数据通信\assignment\assignment2\echoserver2>cd x64
E:\Desktop\秃头文件\数据通信\assignment\assignment2\echoserver2\x64>cd Debug
E:\Desktop\秃头文件\数据通信\assignment\assignment2\echoserver2\x64\Debug>echoserver2
WSAStartup succeed!
Listen socket created!
Listen socket bound to localhost at port 27015
-----
Message from 192
Bytes received: 3
Received: hi
Bytes sent: 3
-----
Message from 212
Bytes received: 6
Received: hahah
Bytes sent: 6
-----
Message from 192
Bytes received: 10
Received: 我是端口1
Bytes sent: 10
-----
Message from 212
Bytes received: 10
Received: 我是端口2
Bytes sent: 10
Connection closing...
Connection closing...

C:\Windows\System32\cmd.exe
Microsoft Windows [版本 10.0.19044.1645]
(c) Microsoft Corporation. 保留所有权利。

E:\Desktop\秃头文件\数据通信\assignment\assignment2\echoclient\x64\Debug>echoclient 127.0.0.1
hahah
Bytes Sent: 6
Received: "hahah"
Bytes received: 6
我是端口2
Bytes Sent: 10
Received: "我是端口2"
Bytes received: 10
quit

E:\Desktop\秃头文件\数据通信\assignment\assignment2\echoclient\x64\Debug>
```

这里可以看到在启动了服务器 echoserver2 后，可以分别通过两个 client 的窗口进行对话传输，而 server 中可以看到来自两个端口的消息传输，检测中英文编码，检测端口对应无误，最后 quit 退出即可。

三、问题与思考

1. gets_s 函数的使用：

背景：gets()函数不安全。C11 标准委员会已经将其废除，建议能不用尽量不用。

解释：gets()函数的作用:它读取整行输入，直至遇到换行符，然后丢弃换行符，储存其余字符，并在其末尾添加一个空字符使其成为一个字符串。听起来挺安全的，问题在于 gets() 函数不检查函数边界，有多少字符它就给你输入多少，这就造成了一个问题：缓冲区溢出（buffer overflow）。这意味着：如果他们有可能擦掉程序中的其他数据（即把数据放到了存储别的数据的地方并将其覆盖）这样就很容易出现问题。

gets()的替代品：fgets(), get_s(), s_gets(), 也就是我们这里使用的 gets_s()。

而在使用中出现“E0304 没有与参数列表匹配的重载函数实例”这一问题，经过查找资料研究发现是字符串存储的一些问题，需要修改几个地方：

```
char sendbuf[DEFAULT_BUflen] = "\0";
char recvbuf[DEFAULT_BUflen] = "\0";

do {
    gets_s(sendbuf, 1000);
} while (1);

#ifdef UNICODE
```

这样可以保证不出现上述报错，这里 gets_s 函数的第二个参数指定了最大数组大小。