服务端：

```
#include<iostream>
#include<string.h>
#include<string>
#include<map>
#include<queue>
#include<WinSock2.h>
#include<WS2tcpip.h>
#pragma comment(lib,"ws2_32.lib")

#include <typeinfo>


#define DEFAULT_BUFLEN 100
#define DEFAULT_PORT 27015

using namespace std;
map<SOCKET, int> user_map;  // 其中的int如果为1则为在线
queue<string> message_queue; /*如果遇到消息并发的情形，需要添加消息队列
                               单开一个处理并发消息转发的线程*/

                               // 为每一个连接到此端口的用户创建一个线程
DWORD WINAPI handlerRequest(LPVOID lparam)
{
    SOCKET ClientSocket = (SOCKET)(LPVOID)lparam;
    user_map[ClientSocket] = 1;

    char user_name[10];
    strcpy_s(user_name, to_string(ClientSocket).data());
    send(ClientSocket, user_name, 10, 0);

    SYSTEMTIME systime = { 0 };
    GetLocalTime(&systime);
    cout << endl << systime.wYear << "年" << systime.wMonth << "月" <<
systime.wDay << "日";
    cout << systime.wHour << "时" << systime.wMinute << "分" << systime.wSecond
<< "秒" << endl;
    cout << "Log: 用户--" << ClientSocket << "--加入聊天！" << endl;
    cout << "-----------------------------------------------------" << endl;

    // 循环接受客户端数据
    int recvResult;
    int sendResult;
    int flag = 1;
    do {
        char recvBuf[DEFAULT_BUFLEN] = "";
        char sendBuf[DEFAULT_BUFLEN] = "";
        recvResult = recv(ClientSocket, recvBuf, DEFAULT_BUFLEN, 0);
        if (recvResult > 0) {
            SYSTEMTIME Logtime = { 0 };
            GetLocalTime(&Logtime);
```

```
            strcpy_s(sendBuf, "用户--");
            string ClientID = to_string(ClientSocket);
            strcat_s(sendBuf, ClientID.data()); // data函数直接转换为char*
            strcat_s(sendBuf, "--: ");
            strcat_s(sendBuf, recvBuf);
            // message_queue.push(sendBuf); //这里将消息存储到队列中

            cout << endl << Logtime.wYear << "年" << Logtime.wMonth << "月" <<
Logtime.wDay << "日";
            cout << Logtime.wHour << "时" << Logtime.wMinute << "分" <<
Logtime.wSecond << "秒" << endl;
            cout << "Log: 用户--" << ClientSocket << "--的消息: " << recvBuf <<
endl;
            cout << "-----------------------------------------------------" <<
endl;
            for (auto it : user_map) {
                if (it.first != ClientSocket && it.second == 1) {
                    sendResult = send(it.first, sendBuf, DEFAULT_BUFLEN, 0);
                    if (sendResult == SOCKET_ERROR)
                        cout << "send failed with error: " << WSAGetLastError()
<< endl;
                }
            }
        }
        else {
            flag = 0;
        }
    } while (recvResult != SOCKET_ERROR && flag != 0);

    GetLocalTime(&systime);
    cout << endl << systime.wYear << "年" << systime.wMonth << "月" <<
systime.wDay << "日";
    cout << systime.wHour << "时" << systime.wMinute << "分" << systime.wSecond
<< "秒" << endl;
    cout << "Log: 用户--" << ClientSocket << "--离开了聊天（quq）" << endl;
    cout << "-----------------------------------------------------" << endl;

    closesocket(ClientSocket);
    return 0;
}

int main()
{

    //----------------------
    // 初始化Winsock
    WSADATA wsaData;
    int iResult;
    iResult = WSAStartup(MAKEWORD(2, 2), &wsaData);
    if (iResult != NO_ERROR) {
        cout << "WSAStartup failed with error: " << iResult << endl;
        return 1;
    }
    //----------------------
    // 创建一个监听的SOCKET
```

```
    // 如果有connect的请求就新创建一个线程
    SOCKET ListenSocket;
    ListenSocket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    if (ListenSocket == INVALID_SOCKET) {
        cout << "socket failed with error: " << WSAGetLastError() << endl;
        WSACleanup();
        return 1;
    }
    //----------------------
    // 用于bind函数绑定的IP地址和端口号
    sockaddr_in service;
    service.sin_family = AF_INET;
    inet_pton(AF_INET, "127.0.0.1", &service.sin_addr.s_addr);
    service.sin_port = htons(27015);
    iResult = bind(ListenSocket, (SOCKADDR*)&service, sizeof(service));
    if (iResult == SOCKET_ERROR) {
        wprintf(L"bind failed with error: %ld\n", WSAGetLastError());
        closesocket(ListenSocket);
        WSACleanup();
        return 1;
    }
    //----------------------
    // 监听即将到来的请求信号
    if (listen(ListenSocket, 5) == SOCKET_ERROR) {
        cout << "listen failed with error: " << WSAGetLastError() << endl;
        closesocket(ListenSocket);
        WSACleanup();
        return 1;
    }
    //----------------------
    // 对于每个新的请求使用多线程处理
    cout << "Waiting for client to connect..." << endl;

    while (1) {
        sockaddr_in addrClient;
        int len = sizeof(sockaddr_in);
        // 接受成功返回与client通讯的Socket
        SOCKET AcceptSocket = accept(ListenSocket, (SOCKADDR*)&addrClient,
&len);
        if (AcceptSocket == INVALID_SOCKET) {
            cout << "accept failed with error: " << WSAGetLastError() << endl;
            closesocket(ListenSocket);
            WSACleanup();
            return 1;
        }
        else {
            // 创建线程，并且传入与client通讯的套接字
            HANDLE hThread = CreateThread(NULL, 0, handlerRequest,
(LPVOID)AcceptSocket, 0, NULL);
            CloseHandle(hThread); // 关闭对线程的引用
        }
    }

    // 关闭服务端SOCKET
    iResult = closesocket(ListenSocket);
```

```
    if (iResult == SOCKET_ERROR) {
        cout << "close failed with error: " << WSAGetLastError() << endl;
        WSACleanup();
        return 1;
    }

    WSACleanup();
    return 0;
}
```

客户端：

```
#include<iostream>
#include<string>
#include<WinSock2.h>
#include<WS2tcpip.h>
#include<thread>
#pragma comment(lib,"ws2_32.lib")

#define DEFAULT_BUFLEN 100
#define DEFAULT_PORT 27015
//#define _WINSOCK_DEPRECATED_NO_WARNINGS 1  //VS2015后启用旧函数

using namespace std;
string quit_string = "quit";
int flag = 1;
char user_name[10]; //接受服务端分发的用户名

DWORD WINAPI Recv(LPVOID lparam_socket) {
    int recvResult;
    SOCKET* recvSocket = (SOCKET*)lparam_socket; //一定要使用指针型变量，因为要指向
connect socket的位置

    while (1) {
        char recvbuf[DEFAULT_BUFLEN] = "";
        recvResult = recv(*recvSocket, recvbuf, DEFAULT_BUFLEN, 0);
        if (recvResult > 0 && flag == 1) {
            SYSTEMTIME systime = { 0 };
            GetLocalTime(&systime);
            cout << endl << endl << systime.wYear << "年" << systime.wMonth <<
"月" << systime.wDay << "日";
            cout << systime.wHour << "时" << systime.wMinute << "分" <<
systime.wSecond << "秒" << endl;
            cout << "收到消息：";
            cout << recvbuf << endl;
            cout << "--------------------------------------------------------" <<
endl;
            cout << "请输入你的消息："; // 提示可以继续发送消息了
        }
        else {
            closesocket(*recvSocket);
            return 1;
        }
    }
}
```

```
DWORD WINAPI Send(LPVOID lparam_socket) {

    // 接受消息直到quit退出聊天
    // flag为是否退出聊天的标志
    int sendResult;
    SOCKET* sendSocket = (SOCKET*)lparam_socket;

    while (1)
    {
        //----------------------
        // 发送消息
        char sendBuf[DEFAULT_BUFLEN] = "";
        cout << "请输入你的消息：";
        cin.getline(sendBuf, DEFAULT_BUFLEN);   // 保证可以输入空格，getline函数设置好了以换行符为结束

        if (string(sendBuf) == quit_string) {
            flag = 0;
            closesocket(*sendSocket);
            cout << endl << "即将退出聊天" << endl;
            return 1;
        }
        else {
            sendResult = send(*sendSocket, sendBuf, DEFAULT_BUFLEN, 0);
            if (sendResult == SOCKET_ERROR) {
                cout << "send failed with error: " << WSAGetLastError() << endl;
                closesocket(*sendSocket);
                WSACleanup();
                return 1;
            }
            else {
                SYSTEMTIME systime = { 0 };
                GetLocalTime(&systime);
                cout << endl << endl << systime.wYear << "年" << systime.wMonth << "月" << systime.wDay << "日";
                cout << systime.wHour << "时" << systime.wMinute << "分" << systime.wSecond << "秒" << endl;
                cout << "消息已成功发送" << endl;
                cout << "------------------------------------------------------" << endl;
            }
        }
    }
}


int main() {

    //----------------------
    //使用iResult的值来表征各个步骤是否操作成功
    int iResult;
    WSADATA wsaData;
    SOCKET ConnectSocket = INVALID_SOCKET;
```

```
    int recvbuflen = DEFAULT_BUFLEN;
    int sendbuflen = DEFAULT_BUFLEN;

    //----------------------
    // 初始化 Winsock,输出信息详细描述
    iResult = WSAStartup(MAKEWORD(2, 2), &wsaData);
    if (iResult != NO_ERROR) {
        cout << "WSAStartup failed with error: " << iResult << endl;
        return 1;
    }

    //----------------------
    // 客户端创建SOCKET内存来连接到服务端
    ConnectSocket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    if (ConnectSocket == INVALID_SOCKET) {
        cout << "socket failed with error: " << WSAGetLastError() << endl;
        WSACleanup();
        return 1;
    }

    //----------------------
    // 创建sockaddr_in结构，再转换成SOCKADDR的结构
    // 要连接的服务端的IP地址、端口号
    struct sockaddr_in clientService;
    clientService.sin_family = AF_INET;
    inet_pton(AF_INET, "127.0.0.1", &clientService.sin_addr.s_addr);
    clientService.sin_port = htons(DEFAULT_PORT);

    //----------------------
    // Connect连接到服务端
    iResult = connect(ConnectSocket, (SOCKADDR*)&clientService,
sizeof(clientService));
    if (iResult == SOCKET_ERROR) {
        cout << "connect failed with error: " << WSAGetLastError() << endl;
        closesocket(ConnectSocket);
        WSACleanup();
        return 1;
    }

    recv(ConnectSocket, user_name, 10, 0);

    // 打印进入聊天的标志
    cout << "              Welcome    User    " << user_name << endl;
    cout << "*****************************************************" << endl;
    cout << "             Use quit command to quit" << endl;
    cout << "-----------------------------------------------------" << endl;

    //----------------------
    // 创建两个线程，一个接受线程，一个发送线程
    HANDLE hThread[2];
    hThread[0] = CreateThread(NULL, 0, Recv, (LPVOID)&ConnectSocket, 0, NULL);
    hThread[1] = CreateThread(NULL, 0, Send, (LPVOID)&ConnectSocket, 0, NULL);

    WaitForMultipleObjects(2, hThread, TRUE, INFINITE);
    CloseHandle(hThread[0]);
```

```
    CloseHandle(hThread[1]);

    // 关闭socket
    iResult = closesocket(ConnectSocket);
    WSACleanup();
    return 0;
}
```