计算机网络第一次实验报告

2013536 汤清云

# 实验思路

## **服务器端编程的步骤：**

1：加载套接字库，创建套接字(WSAStartup()/[socket](https://so.csdn.net/so/search?q=socket&spm=1001.2101.3001.7020)())；

2：绑定套接字到一个IP地址和一个端口上(bind())；

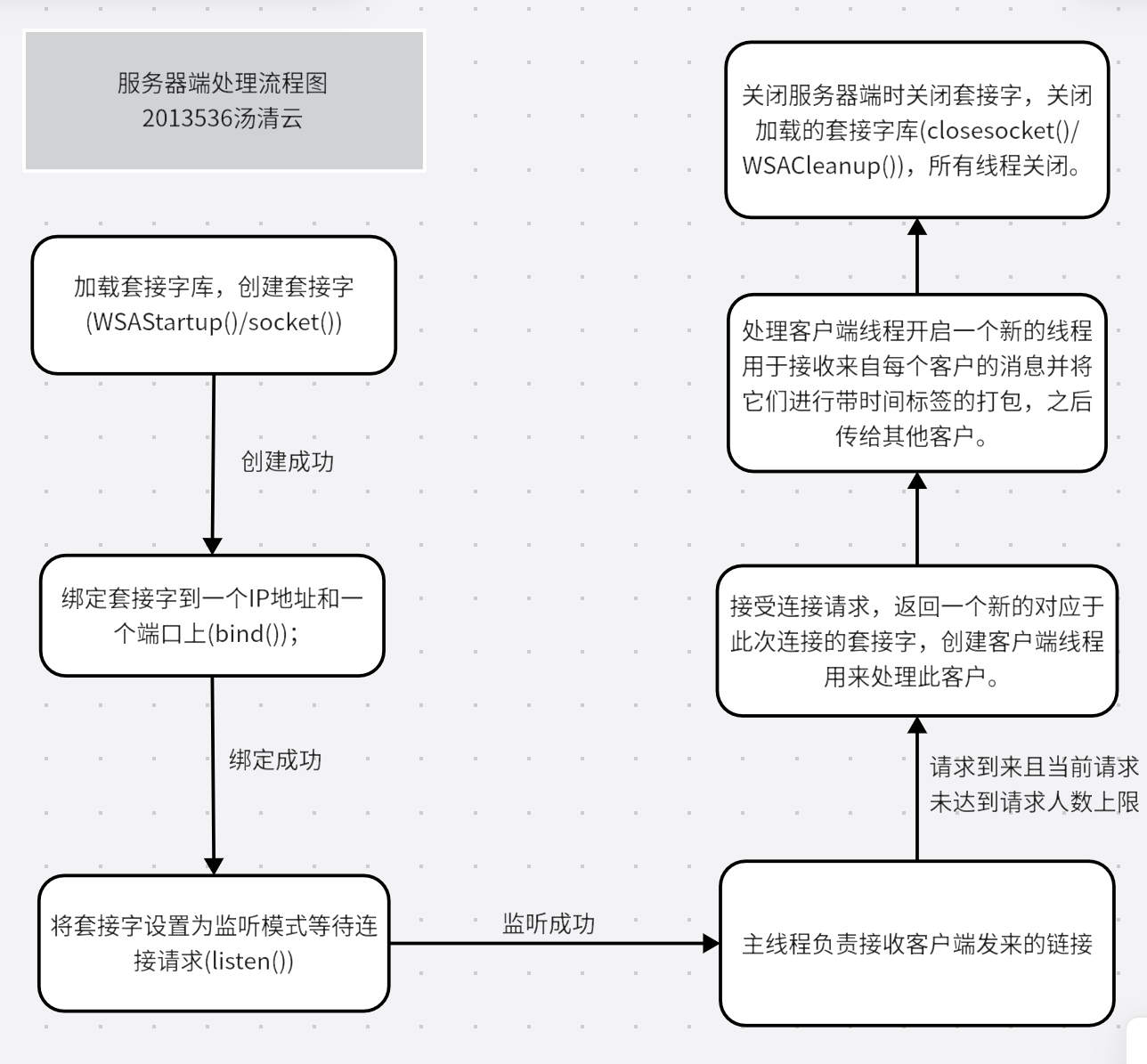
3：将套接字设置为监听模式等待连接请求(listen())；

4：请求到来后，接受连接请求，返回一个新的对应于此次连接的套接字(accept())；

5：用返回的套接字和客户端进行通信(send()/recv())；

6：返回，等待另一连接请求；

7：关闭套接字，关闭加载的套接字库(closesocket()/WSACleanup())。



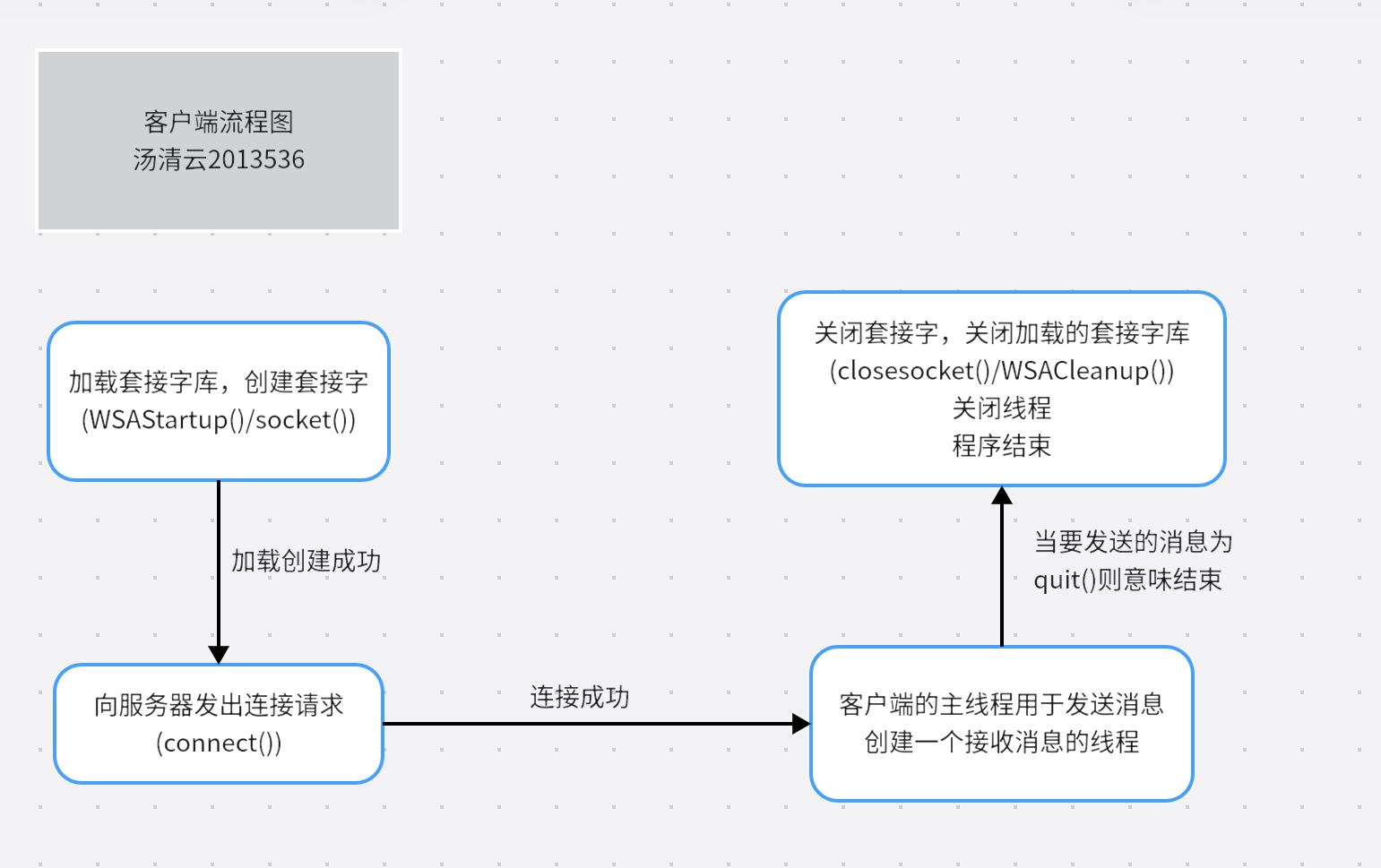
## **客户端编程的步骤：**

1：加载套接字库，创建套接字(WSAStartup()/socket())；

2：向服务器发出连接请求(connect())；

3：和服务器端进行通信(send()/recv())；

4：关闭套接字，关闭加载的套接字库(closesocket()/WSACleanup())。



# 实验过程

## 最初版本代码实现

完成任务：实现单个客户端和单个服务器端的消息互传、并且进行了时间打印操作。

缺点：在发送和接收消息时必须是“一来一回”的形式，且必须先由客户端发起对话。

服务器端代码：

|  |
| --- |
| #include<iostream>  #include<WinSock2.h>  #include<cstring>  #include<ctime>  #pragma comment(lib,"ws2\_32.lib")//socket库  using namespace std;  int main()  {  WSADATA wsaData;  //wVersionRequested是Windows Sockets API提供的调用方可使用的最高版本号。  //高位字节指出副版本(修正)号，低位字节指明主版本号。  WORD mVersionRequested = MAKEWORD(2, 2);  //加载套接字库 flag表示是否成功  int flag = WSAStartup(mVersionRequested, &wsaData);  time\_t now = time(nullptr);  char\* curr\_time = ctime(&now);  if (flag == 0)  {  cout <<curr\_time<< "套接字库加载成功" << endl;  }  else  {  cout << curr\_time<< "套接字库加载失败" << endl;  }  //创建socket，指定地址类型为AF\_INET，流式套接字，TCP协议  SOCKET server = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);  //为之后绑定套接字做准备  SOCKADDR\_IN ServerAddr;  ServerAddr.sin\_family = AF\_INET;  ServerAddr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");//IP地址  ServerAddr.sin\_port = htons(6666);//端口号  // 绑定套接字到一个IP地址和一个端口上  flag=bind(server, (SOCKADDR\*)&ServerAddr, sizeof(SOCKADDR));  now = time(nullptr);  curr\_time = ctime(&now);  if (flag == 0)  {  cout << curr\_time << "绑定成功" << endl;  }  else  {  cout << curr\_time << "绑定失败" << endl;  }  //将套接字设置为监听模式等待连接请求  flag=listen(server, 10);//同一队列中最多十个请求  now = time(nullptr);  curr\_time = ctime(&now);  if (flag == 0)  {  cout << curr\_time << "监听成功" << endl;  }  else  {  cout << curr\_time << "监听失败" << endl;  }  //请求到来后，接受连接请求，返回一个新的对应于此次连接的套接字  int len = sizeof(SOCKADDR);  SOCKET accepter = accept(server, (SOCKADDR\*)&ServerAddr, &len);  //用返回的套接字和客户端进行通信(send()/recv())；  char recvtext[500];  char sendtext[500];  //获取当前时间  cout << "——————————聊天开始————————" << endl;  while (1)  {  recv(accepter, recvtext, 500, 0);  now = time(nullptr);  curr\_time = ctime(&now);  cout << curr\_time;  cout << "收到来自客户端的信息："<<endl;  cout << recvtext << endl;  cin.getline(sendtext, 499);  if (!(strcmp(sendtext, "quit()")))//退出聊天信息  {  break;  }  send(accepter, sendtext, sizeof(sendtext), 0);  now = time(nullptr);  curr\_time = ctime(&now);  cout << curr\_time << "信息\""<<sendtext<<"\"发送成功" << endl;  }  //关闭套接字，关闭加载的套接字库(closesocket()/WSACleanup())  closesocket(accepter);  closesocket(server);  WSACleanup();  } |

客户端代码：

|  |
| --- |
| #include<iostream>  #include<winsock2.h>  #include<cstring>  #include<ctime>  #pragma comment(lib,"ws2\_32.lib")//socket库  using namespace std;  int main()  {  WSADATA wsaData;  //wVersionRequested是Windows Sockets API提供的调用方可使用的最高版本号。  //高位字节指出副版本(修正)号，低位字节指明主版本号。  WORD mVersionRequested = MAKEWORD(2, 2);  //加载套接字库 flag表示是否成功  int flag=WSAStartup(mVersionRequested, &wsaData);  //获取当前时间  time\_t now = time(nullptr);  char\* curr\_time = ctime(&now);  if (flag == 0)  {  cout <<curr\_time<< "套接字库加载成功" << endl;  }  else  {  cout <<curr\_time<< "套接字库加载失败" << endl;  }  //创建socket，指定地址类型为AF\_INET，流式套接字，TCP协议  SOCKET client = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);  //为之后连接和传输数据做准备  SOCKADDR\_IN ClientAddr;  ClientAddr.sin\_family = AF\_INET;  ClientAddr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");//IP地址  ClientAddr.sin\_port = htons(6666);//端口号  //进行连接  flag=connect(client, (SOCKADDR\*)&ClientAddr, sizeof(SOCKADDR));  now = time(nullptr);  curr\_time = ctime(&now);  if (flag == 0)  {  cout << curr\_time << "建立连接成功" << endl;  }  else  {  cout << curr\_time << "建立连接失败" << endl;  }  //发送消息  char clienttext[500] = {};  char servertext[500] = {};  cout << "——————————聊天开始————————" << endl;  while (1)  {    cin.getline(clienttext, 499);  if (!(strcmp(clienttext, "quit()")))//退出聊天信息  {  break;  }  send(client, clienttext, sizeof(clienttext), 0);  now = time(nullptr);  curr\_time = ctime(&now);  cout << curr\_time << "信息\""<<clienttext<<"\"发送成功" << endl;  recv(client, servertext, sizeof(servertext), 0);  now = time(nullptr);  curr\_time = ctime(&now);  cout << curr\_time;  cout << "收到来自服务器端的消息：" << endl;  cout<< servertext << endl;  }  //关闭套接字\套接字库  send(client, "聊天结束\n", sizeof("聊天结束\n"), 0);  closesocket(client);  WSACleanup();  return 0;  } |

## 提高版本代码实现

完成任务：在这一版本中，我学习并且使用了多线程以达到单个客户端/服务器端可以发送多条消息，而不仅仅拘泥于“一来一回”的交互。

在这一版本中，我发现在客户端退出聊天时服务器端不停打印空字符串，经过检查后发现是由于缺少对客户端传来消息的检查。因此新增对字符串的检查，当长度<0时说明对面不再传来消息，因此退出聊天。

服务器端代码：

|  |
| --- |
| #include<iostream>  #include<WinSock2.h>  #include<cstring>  #include<ctime>  #pragma comment(lib,"ws2\_32.lib")//socket库  using namespace std;  int main()  {  WSADATA wsaData;  //wVersionRequested是Windows Sockets API提供的调用方可使用的最高版本号。  //高位字节指出副版本(修正)号，低位字节指明主版本号。  WORD mVersionRequested = MAKEWORD(2, 2);  //加载套接字库 flag表示是否成功  int flag = WSAStartup(mVersionRequested, &wsaData);  time\_t now = time(nullptr);  char\* curr\_time = ctime(&now);  if (flag == 0)  {  cout <<curr\_time<< "套接字库加载成功" << endl;  }  else  {  cout << curr\_time<< "套接字库加载失败" << endl;  }  //创建socket，指定地址类型为AF\_INET，流式套接字，TCP协议  SOCKET server = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);  //为之后绑定套接字做准备  SOCKADDR\_IN ServerAddr;  ServerAddr.sin\_family = AF\_INET;  ServerAddr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");//IP地址  ServerAddr.sin\_port = htons(6666);//端口号  // 绑定套接字到一个IP地址和一个端口上  flag=bind(server, (SOCKADDR\*)&ServerAddr, sizeof(SOCKADDR));  now = time(nullptr);  curr\_time = ctime(&now);  if (flag == 0)  {  cout << curr\_time << "绑定成功" << endl;  }  else  {  cout << curr\_time << "绑定失败" << endl;  }  //将套接字设置为监听模式等待连接请求  flag=listen(server, 10);//同一队列中最多十个请求    now = time(nullptr);  curr\_time = ctime(&now);  if (flag == 0)  {  cout << curr\_time << "监听成功" << endl;  }  else  {  cout << curr\_time << "监听失败" << endl;  }  //请求到来后，接受连接请求，返回一个新的对应于此次连接的套接字  int len = sizeof(SOCKADDR);  SOCKET accepter = accept(server, (SOCKADDR\*)&ServerAddr, &len);  //创建一个接受消息的线程  DWORD WINAPI servEventThread(LPVOID IpParameter);//服务器端处理线程  //不需要句柄所以直接关闭  CloseHandle(CreateThread(NULL, 0, servEventThread, (LPVOID)&accepter, 0, 0));  //用返回的套接字和客户端进行通信(send()/recv())；  char sendtext[500];  //获取当前时间  cout << "——————————聊天开始————————" << endl;  while (1)  {  cin.getline(sendtext, 499);  //获取当前时间  time\_t now = time(nullptr);  string curr\_time = ctime(&now);  if (!(strcmp(sendtext, "quit()")))//退出聊天信息  {  cout << "您已经选择结束聊天。" << endl;  return 0;  }  else  {  send(accepter, sendtext, sizeof(sendtext), 0);  now = time(nullptr);  curr\_time = ctime(&now);  //strcat(curr\_time, sendtext);  //curr\_time += string(sendtext);  cout << curr\_time << "信息\"" << sendtext << "\"发送成功" << endl;  memset(sendtext, 0, sizeof(sendtext));  }  }  //关闭套接字，关闭加载的套接字库(closesocket()/WSACleanup())  closesocket(accepter);  closesocket(server);  WSACleanup();  }  DWORD WINAPI servEventThread(LPVOID IpParameter)//服务器接收消息的线程  {  SOCKET accepter = \*(SOCKET\*)IpParameter;//获取服务器端的SOCKET参数  time\_t now = time(nullptr);  char\* curr\_time = ctime(&now);  char recvtext[500];  while (1)  {  int recvlength = recv(accepter, recvtext, 500, 0);  if (recvlength <= 0)  {  cout << "对方已经退出聊天。" << endl;  break;  }  else  {  now = time(nullptr);  curr\_time = ctime(&now);  cout << curr\_time;  cout << "收到来自客户端的信息：" << endl;  cout << recvtext << endl;  memset(recvtext, 0, sizeof(recvtext));  }  }  return 0;  } |

## 最终版本代码实现

完成任务：在之前基础上完成了多客户端交互的功能。

服务器端：

|  |
| --- |
| #include<iostream>  #include<WinSock2.h>  #include<cstring>  #include<ctime>  #pragma comment(lib,"ws2\_32.lib")//socket库  using namespace std;  DWORD WINAPI servEventThread(LPVOID& IpParameter);//服务器端收消息线程  DWORD WINAPI ClientThread(LPVOID IpParameter);//服务器端多客户端线程函数声明  int main()  {  WSADATA wsaData;  //wVersionRequested是Windows Sockets API提供的调用方可使用的最高版本号。  //高位字节指出副版本(修正)号，低位字节指明主版本号。  WORD mVersionRequested = MAKEWORD(2, 2);  //加载套接字库 flag表示是否成功  int flag = WSAStartup(mVersionRequested, &wsaData);  time\_t now = time(nullptr);  char\* curr\_time = ctime(&now);  if (flag == 0)  {  cout <<curr\_time<< "套接字库加载成功" << endl;  }  else  {  cout << curr\_time<< "套接字库加载失败" << endl;  }  //创建socket，指定地址类型为AF\_INET，流式套接字，TCP协议  SOCKET server = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);  //为之后绑定套接字做准备  SOCKADDR\_IN ServerAddr;  ServerAddr.sin\_family = AF\_INET;  ServerAddr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");//IP地址  ServerAddr.sin\_port = htons(6666);//端口号  // 绑定套接字到一个IP地址和一个端口上  flag=bind(server, (SOCKADDR\*)&ServerAddr, sizeof(SOCKADDR));  now = time(nullptr);  curr\_time = ctime(&now);  if (flag == 0)//判断绑定  {  cout << curr\_time << "绑定成功" << endl;  }  else  {  cout << curr\_time << "绑定失败" << endl;  }  //为了实现多个客户，给服务端sokect绑定一个事件对象，用来接收客户端链接的事件  WSAEVENT servEvent = WSACreateEvent();//创建一个人工重设为传信的事件对象  //绑定事件对象，并且监听所有事件  WSAEventSelect(server, servEvent, FD\_ALL\_EVENTS);  //将套接字设置为监听模式等待连接请求  flag=listen(server, 10);//同一队列中最多十个请求    now = time(nullptr);  curr\_time = ctime(&now);  if (flag == 0)  {  cout << curr\_time << "监听成功" << endl;  }  else  {  cout << curr\_time << "监听失败" << endl;  }  //请求到来后，接受连接请求，返回一个新的对应于此次连接的套接字  int len = sizeof(SOCKADDR);  SOCKET accepter = accept(server, (SOCKADDR\*)&ServerAddr, &len);  if (accepter == INVALID\_SOCKET)  cout << "没有收到连接请求。" << endl;  HANDLE hThread = CreateThread(NULL, 0, ClientThread, (LPVOID)&accepter, 0, 0);  //主线程负责接收客户端连接  while (1)  {  //SOCKET accepter = accept(server, (SOCKADDR\*)&ServerAddr, &len);  //创建线程用来负责多客户端  //if (accepter != INVALID\_SOCKET)  //{  //HANDLE hThread = CreateThread(NULL, 0, ClientThread, (LPVOID)&accepter, 0, 0);  //CloseHandle(hThread);//关闭对线程的引用  //}  //accepter = INVALID\_SOCKET;//接收一个关闭一个  }  //不需要句柄所以直接关闭  //CloseHandle(CreateThread(NULL, 0, servEventThread, (LPVOID)&accepter, 0, 0));    //关闭套接字，关闭加载的套接字库(closesocket()/WSACleanup())    closesocket(server);  WSACleanup();  }  DWORD WINAPI servEventThread(LPVOID IpParameter)  {  SOCKET accepter = \*(SOCKET\*)IpParameter;//获取服务器端的SOCKET参数  time\_t now = time(nullptr);  char\* curr\_time = ctime(&now);  char recvtext[500];  //cout << "进入此函数" << endl;  while (1)  {  int recvlength = recv(accepter, recvtext, 500, 0);  if (recvlength < 0)  {  cout << "对方已经退出聊天。" << endl;  break;  }  else  {  now = time(nullptr);  curr\_time = ctime(&now);  cout << curr\_time;  cout << "收到来自客户端的信息：" << endl;  cout << recvtext << endl;  memset(recvtext, 0, sizeof(recvtext));  }  }  return 0;  }  DWORD WINAPI ClientThread(LPVOID IpParameter)//用来跟多个客户端交互。  {  SOCKET accepter = \*(SOCKET\*)IpParameter;//获取服务器端的SOCKET参数  //再开个线程收数据  HANDLE hThread = CreateThread(NULL, 0, servEventThread, (LPVOID)&accepter, 0, 0);  CloseHandle(hThread);  //用返回的套接字和客户端进行通信(send()/recv())；  cout << "——————————聊天开始————————" << endl;  char sendtext[500];  //获取当前时间  //cout << "我在这" << endl;  while (1)  {  cin.getline(sendtext, 499);  //获取当前时间  time\_t now = time(nullptr);  string curr\_time = ctime(&now);  if (!(strcmp(sendtext, "quit()")))//退出聊天信息  {  cout << "您已经选择结束聊天。" << endl;  return 0;  }  else  {  send(accepter, sendtext, sizeof(sendtext), 0);  now = time(nullptr);  curr\_time = ctime(&now);  cout << curr\_time << "信息\"" << sendtext << "\"发送成功" << endl;  memset(sendtext, 0, sizeof(sendtext));  }  }  } |

# 实验中遇到的问题

1. 使用vs2019进行编程，在设置ip地址时使用函数inet\_addr会报错；其原因为vs2013及以上版本增加了新函数以绑定ip，在编译时默认使用新的函数，故而报错。

解决方法：项目 -> 属性 -> C/C++ ->常规->SDL检查，将“是”改为“否”，即可。