

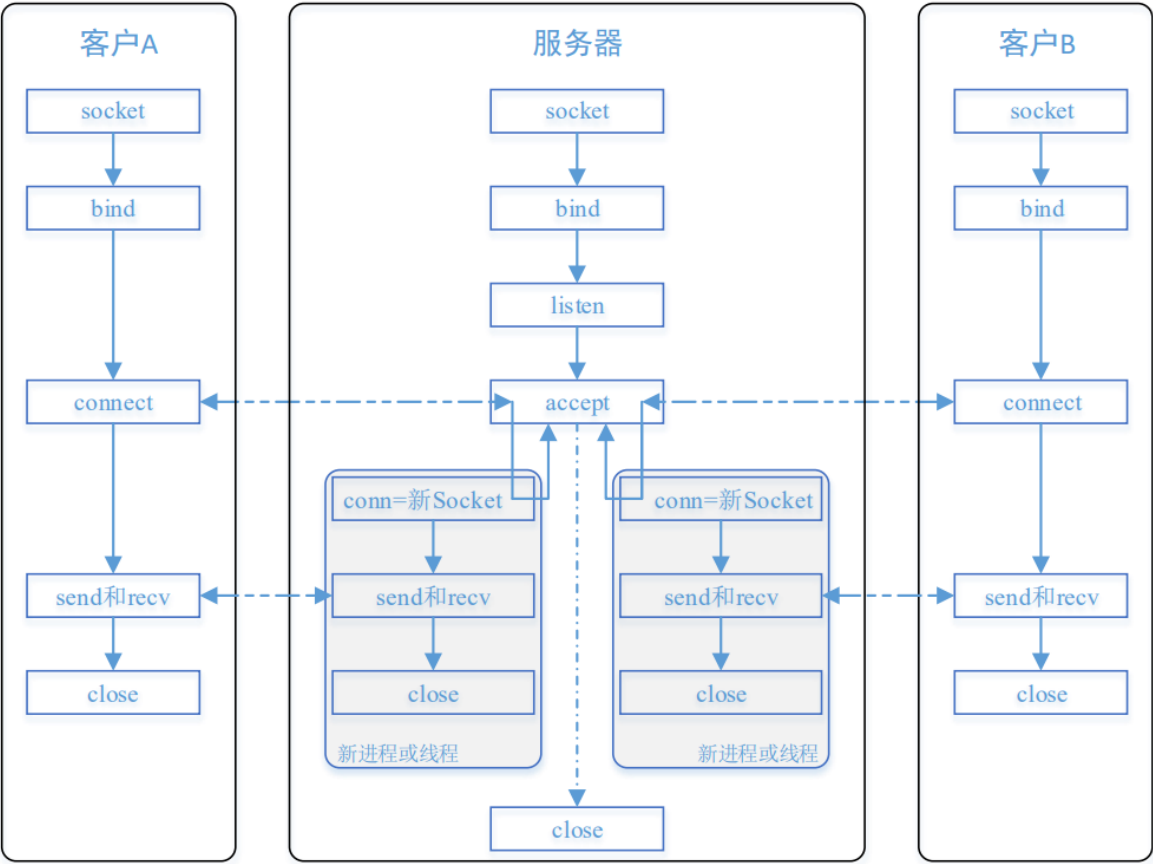
编程作业一：基于TCP协议的网络聊天室

学号：2111698

姓名：于泽林

协议设计

- 基于TCP协议设计，采用C/S架构
- 数据报格式：**username : message**
- 服务器监听自己的**1234**端口，等待客户端的连接
- 连接交由新的线程处理，主线程继续等待新的连接
- 服务端将用户输入的消息转发给其他所有已激活的客户端
- 用户输入“quit”结束连接，线程关闭



各模块功能

- 服务端

加载winsock2环境

```
1 WSADATA wd;
2 if (WSAStartup(MAKEWORD(2, 2), &wd) != 0)
3 {
4     printf("WSAStartup error\n");
5     return 1;
6 }
```

创建流式套接字

```
1 SOCKET welcome = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
2 if (welcome == INVALID_SOCKET)
3 {
4     printf("create socket error\n");
5     return 1;
6 }
```

绑定服务器IP和端口

```
1 sockaddr_in addr;
2 memset(&addr, 0, sizeof(addr));
3 addr.sin_family = AF_INET;
4 addr.sin_port = htons(1234);
5 addr.sin_addr.s_addr = inet_addr("127.0.0.1");
6
7 if (bind(welcome, (SOCKADDR*)&addr, sizeof(sockaddr_in)) == SOCKET_ERROR)
8 {
9     printf("bind error\n");
10    return 1;
11 }
```

持续监听

```
1 if (listen(welcome, 5) == SOCKET_ERROR)
2 {
3     printf("listen error\n");
4     return 1;
5 }
```

服务端的核心部分，用于接收客户端的连接并获取用户名，然后分配一个线程

```
1 while (1)
2 {
3     if (linkCount >= MAX_LINK_NUMBER) continue;
4     Client *client = &clients[linkCount];
5     client->s = accept(welcome, (SOCKADDR*)&remoteAddr, &len);
6     if (client->s == INVALID_SOCKET)
7     {
8         printf("accept error\n");
9         continue;
10    }
11    //接收到连接
12    client->idx = linkCount;
13    recv(client->s, client->username, sizeof(client->username), 0); //接
    收用户名
14    printf("接收到一个连接,IP:%s port:%d username:%s\n",
        inet_ntoa(remoteAddr.sin_addr), ntohs(remoteAddr.sin_port), client-
        >username);
15
16
17 }
```

```

18     //单独开启线程处理与客户端的连接
19     DWORD dwThreadId;
20     HANDLE hThread;
21     hThread = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)AnswerThread,
client, 0, &dwThreadId);
22     if (hThread == NULL){    //线程创建失败, 连接取消
23         printf("线程创建出错\n");
24         continue;
25     }
26     else{
27         printf("线程创建成功,服务器继续等待新的连接.....\n");
28         CloseHandle(hThread);    //关闭句柄, 并不是结束线程
29     }
30
31     ++linkCount;
32 }

```

线程代码, 用于接收客户端发来的消息, 然后转发给已激活的所有客户端

```

1  DWORD WINAPI AnswerThread(LPVOID lparam)
2  {
3      printf("线程%d开始处理\n", GetCurrentThreadId());
4
5      Client *client = (Client*)lparam;
6      char recvData[128] = {0};
7      char buffer[256] = {0};
8      char colon[] = ":";
9
10     while (1)
11     {
12         int recvlen = recv(client->s, recvData, sizeof(recvData), 0);
13         if (recvlen > 0)
14         {
15             if (strcmp(recvData, "quit") == 0)
16             {
17                 break;
18             }
19             else{
20                 strcpy(buffer, client->username);
21                 strcat(buffer, colon);
22                 strcat(buffer, recvData);
23                 //给发送者之外的所有客户发送
24                 for (int i = 0; i < linkCount; ++i)
25                 {
26                     if (i != client->idx)
27                         send(clients[i].s, buffer, sizeof(buffer), 0);
28                 }
29             }
30         }
31         else
32         {
33             printf("接收信息失败\n");
34             printf("%d\n", GetLastError());
35         }
36     }

```

```

37
38     closesocket(client->s);
39     printf("线程号%d结束\n", GetCurrentThreadId());
40     return 0;
41 }

```

- 客户端

此线程用于获取标准输入流中的字符流并且通过缓冲发送给服务器端，若用户输入“quit”则返回，同时更改全局变量**Finished**取值，代表着此次聊天结束

```

1  unsigned ThreadSend(void *param)
2  {
3      char buf[128] = {0};
4      while (1)
5      {
6          input(buf, sizeof(buf));
7
8          int sendlen = send(*(SOCKET*)param, buf, sizeof(buf), 0);
9          if (sendlen == SOCKET_ERROR) printf("send message failed\n");
10         if (strcmp(buf, "quit") == 0)
11         {
12             Finished = 1;
13             return 1;
14         }
15     }
16
17     return 0;
18 }

```

将接收到的有效信息打印在控制台上

```

1  unsigned ThreadRecv(void *param)
2  {
3      char buf[128] = {0};
4      while (1)
5      {
6          int recvlen = recv(*(SOCKET*)param, buf, sizeof(buf), 0);
7          if (recvlen == SOCKET_ERROR)
8          {
9              // printf("something missing\n");
10             return 1;
11         }
12         else if (strlen(buf) > 0)
13         {
14             printf("%s\n", buf);
15         }
16     }
17
18     return 0;
19 }

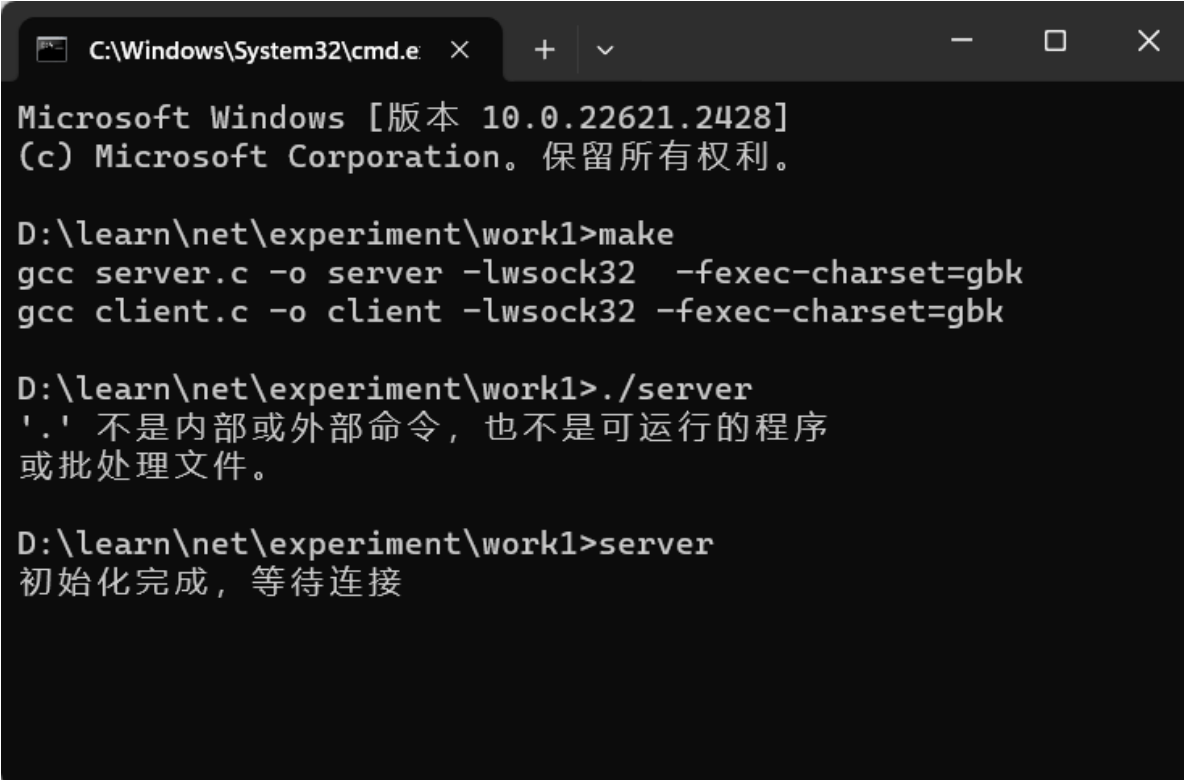
```

主要逻辑实现，开启两个线程，分管接收和发送，同时while循环保证主线程不退出，当且仅当Finished变量不为0时退出循环，然后关闭两个线程，且清理环境并退出程序

```
1 HANDLE hThread_send, hThread_recv;
2 DWORD dwThread_send, dwThread_recv;
3
4 hThread_send = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)ThreadSend,
    &s, 0, &dwThread_send);
5 hThread_recv = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)ThreadRecv,
    &s, 0, &dwThread_recv);
6
7 while (!Finished)
8 {}
9
10 CloseHandle(hThread_send);
11 CloseHandle(hThread_recv);
12
13 closesocket(s);
14 WSACleanup();
```

界面展示

make编译源代码,./server运行服务端



```
C:\Windows\System32\cmd.e x + v
Microsoft Windows [版本 10.0.22621.2428]
(c) Microsoft Corporation。保留所有权利。

D:\learn\net\experiment\work1>make
gcc server.c -o server -lwsck32 -fexec-charset=gbk
gcc client.c -o client -lwsck32 -fexec-charset=gbk

D:\learn\net\experiment\work1>./server
'.' 不是内部或外部命令，也不是可运行的程序
或批处理文件。

D:\learn\net\experiment\work1>server
初始化完成，等待连接
```

./client运行客户端

```
C:\Windows\System32\cmd.e  X  +  v  -  □  X

Microsoft Windows [版本 10.0.22621.2428]
(c) Microsoft Corporation。保留所有权利。

D:\learn\net\experiment\work1>client
connecting.....
连接成功
Please input your username : |
```

聊天界面

```
C:\Windows\System32\cmd.e  X  +  v  -  □  X  C:\Windows\System32\cmd.e  X  +  v  -  □  X  X

D:\learn\net\experiment\work1>make
gcc server.c -o server -lwsock32 -fexec
gcc client.c -o client -lwsock32 -fexec

D:\learn\net\experiment\work1>./server
'.' 不是内部或外部命令，也不是可运行的程序
或批处理文件。

D:\learn\net\experiment\work1>server
初始化完成，等待连接
接收到一个连接,IP:127.0.0.1 port:55170 u
线程创建成功,服务器继续等待新的连接.....
线程25044开始处理
接收到一个连接,IP:127.0.0.1 port:55198 u
线程创建成功,服务器继续等待新的连接.....
线程25088开始处理

C:\Windows\System32\cmd.e  X  +  v  -  □  X

Microsoft Windows [版本 10.0.22621.2428]
(c) Microsoft Corporation。保留所有权利。

D:\learn\net\experiment\work1>client
connecting.....
连接成功
Please input your username : otto
大家好啊，我是说的道理
xuan:原来你也玩原神

C:\Windows\System32\cmd.e  X  +  v  -  □  X

Microsoft Windows [版本 10.0.22621.2428]
(c) Microsoft Corporation。保留所有权利。

D:\learn\net\experiment\work1>client
connecting.....
连接成功
Please input your username : xuan
otto:大家好啊，我是说的道理
xuan:原来你也玩原神
```

退出

```
C:\Windows\System32\cmd.e  X  +  v  -  □  X  C:\Windows\System32\cmd.e  X  +  v  -  □  X  X

gcc client.c -o client -lwsock32 -fexec-

D:\learn\net\experiment\work1>./server
'.' 不是内部或外部命令，也不是可运行的程序
或批处理文件。

D:\learn\net\experiment\work1>server
初始化完成，等待连接
接收到一个连接,IP:127.0.0.1 port:55170 u
线程创建成功,服务器继续等待新的连接.....
线程25044开始处理
接收到一个连接,IP:127.0.0.1 port:55198 u
线程创建成功,服务器继续等待新的连接.....
线程25088开始处理
线程号25044结束
线程号25088结束

C:\Windows\System32\cmd.e  X  +  v  -  □  X

Microsoft Windows [版本 10.0.22621.2428]
(c) Microsoft Corporation。保留所有权利。

D:\learn\net\experiment\work1>client
connecting.....
连接成功
Please input your username : otto
大家好啊，我是说的道理
xuan:原来你也玩原神
quit

D:\learn\net\experiment\work1>

C:\Windows\System32\cmd.e  X  +  v  -  □  X

Microsoft Windows [版本 10.0.22621.2428]
(c) Microsoft Corporation。保留所有权利。

D:\learn\net\experiment\work1>client
connecting.....
连接成功
Please input your username : xuan
otto:大家好啊，我是说的道理
xuan:原来你也玩原神
quit

D:\learn\net\experiment\work1>
```

问题与思考

本次实验采用纯C语言实现，在很多方面尤其是字符串操作相关的实现上与之前熟悉的C++有很大不同，并且对于多线程的实现上也存在诸多技术难点。我的思考在于本工程采用服务器客户端架构，如果采取P2P架构的话又该如何设计。