

利用 Socket 设计聊天程序

2112514 辛浩然

实验要求

给出聊天协议的完整说明；

利用 C 或 C++ 语言，使用基本的 Socket 函数完成程序；不允许使用 CSocket 等封装后的类编写程序；

使用流式套接字、采用多线程（或多进程）方式完成程序；

程序应该有基本的对话界面，但可以不是图形界面。程序应该有正常的退出方式。

完成的程序应该支持多人聊天，支持英文和中文聊天；

编写的程序应该结构清晰，具有较好的可读性；

在实验中观察是否有数据丢失，提交可执行文件、程序源码和实验报告。

程序概览

本次实验基于Socket实现了一个简单的多人聊天程序，使用流式套接字、采用多线程，支持多客户端同时在线中英文聊天，能够显示时间戳，包括群聊和私聊功能，设计正常的退出方式。整个流程包括一些相应的反馈。

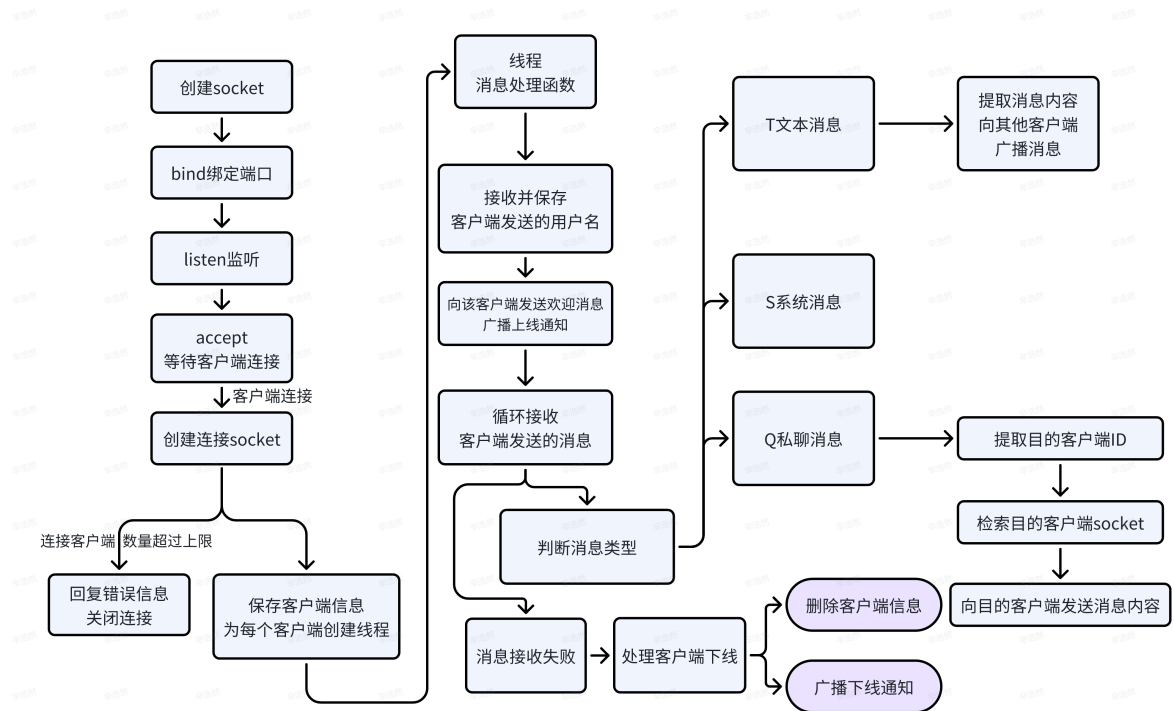
多线程体现在：

- 服务器：为每个连接的客户端创建一个线程，在线程处理函数中处理客户端发送的消息并作出相应行为（如广播等）
- 客户端：
 - 发送消息：主线程循环获取用户输入发送消息；
 - 接收消息：创建线程，循环接收消息。

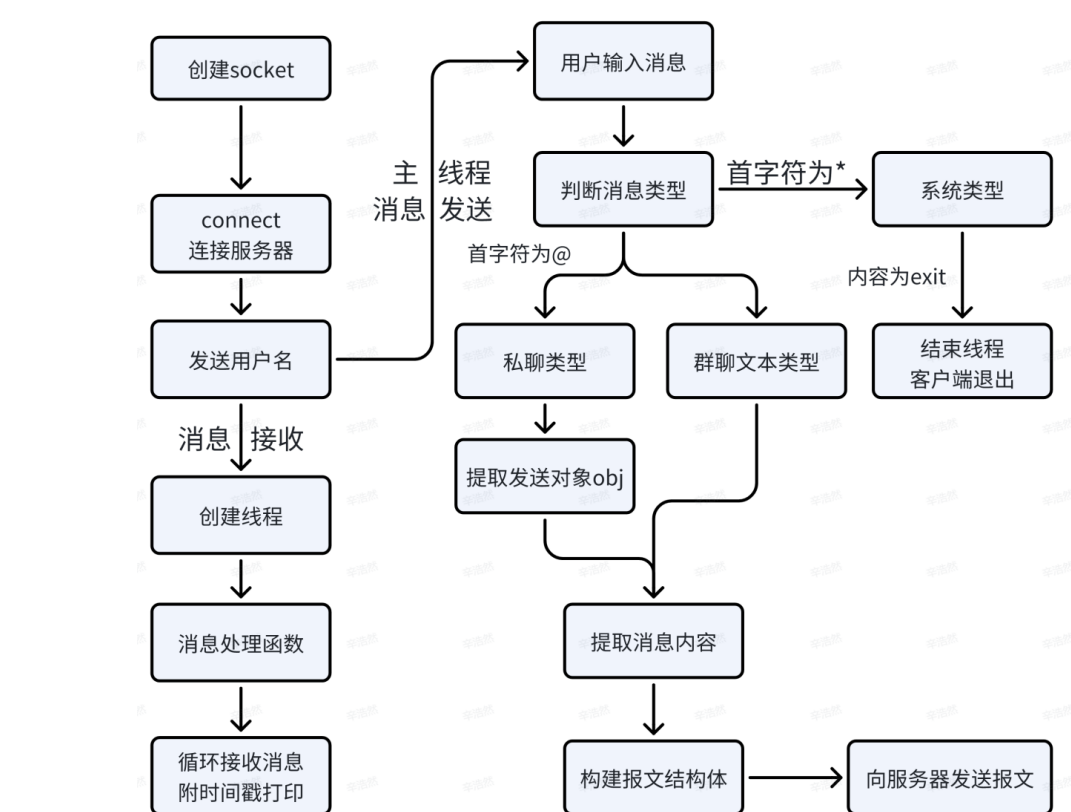
在与助教检查后，意识到：在服务器多线程接收消息时，会读写共享变量。多个线程并发执行时，因此可能同时访问共享的内存位置或资源，这可能导致数据损坏或不确定的行为。虽然在本次实验的情况下，不会出现同时访问共享资源的情形。但我还是改进了代码，在服务器代码中使用互斥锁来保护共享资源，以确保多个线程不会同时访问或修改它们。

程序的基本流程如下：

服务器



客户端



协议设计

首先，设计报文结构体如下，包括消息类型、发送的目标客户端ID和消息内容。

- 消息类型默认为T，即群聊文本类型；S为系统类型，根据 `content` 处理不同的系统命令，如 `exit`；Q为私聊类型，根据 `obj` 确定私聊客户端ID，`content` 为发送消息的内容。
- `obj` 为定长数组，默认为 `00000`，保存私聊模式下目标客户端ID。
- `content` 为消息内容，实际长度动态分配。

```
// 定义报文结构体
struct Message
{
    char type; // 标识消息类型
    char obj[5];
    char content[1]; // 先使用长度为 1 的数组，实际长度会动态分配
};
```

具体而言，协议如下：

1. 服务器端启动并等待客户端连接。
2. 客户端启动并尝试连接到服务器。
3. 客户端成功连接到服务器后，
 - 服务器接受连接并创建一个新的线程来处理该客户端的请求。
 - 客户端首先将用户的昵称将被发送到服务器。
 - 服务器线程处理函数首先接收客户端昵称，然后循环接收客户端发来的消息并针对性地处理。
 - 服务器主线程继续等待新的客户端连接。
4. 客户端创建另一个线程，该线程不断循环接收来自服务器的消息并将其显示在控制台上。
5. 在主线程中，客户端可以循环输入消息，然后根据消息的内容构建消息结构体（`Message`）。
 - 如果消息以 '*' 开头，它被识别为系统消息；
 - 如果消息以 '@' 开头，它被识别为私聊消息；
 - 其他情况下为群聊文本消息。

对于三种不同模式的消息，分别设计不同协议：

系统消息协议

- 客户端识别消息类型为系统消息后，提取后面输入的内容作为 `content`
- 如果 `content` 为 `exit`，客户端关闭线程，关闭套接字，释放资源，结束程序。
- 服务器无法接收到客户端发送的消息，处理客户端退出：
 - 删除客户端信息
 - 广播下线通知

群聊文本消息协议

- 客户端识别消息类型为文本消息后，提取后面输入的内容作为 `content`，构建 `message` 结构体。
- 客户端通过socket将 `message` 发送到服务器。
- 服务器接收到来自客户端的消息，根据消息类型识别为文本消息。
 - 提取消息内容，遍历客户端列表，获取所有上线客户端的socket，广播至其他客户端。

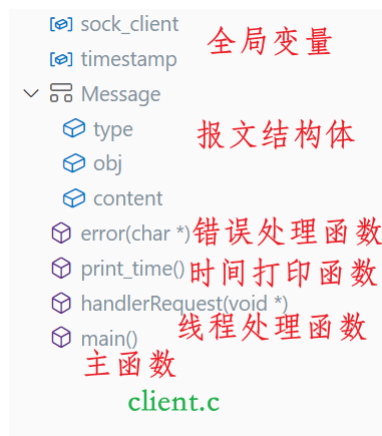
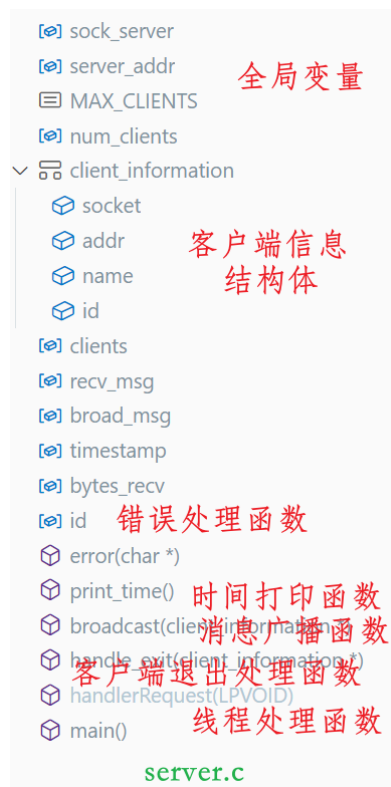
私聊消息协议

- 客户端识别消息类型为私聊消息，提取用户输入的目标id作为 `obj`、消息内容作为 `content`，构建 `message` 结构体。
- 客户端通过socket将 `message` 发送到服务器；
- 服务器接收到来自客户端的消息，根据消息类型识别为私聊消息。
 - 在接收到的报文中提取目标id，在客户端列表中检索，找到目标客户端及其socket；
 - 提取消息内容；
 - 将消息内容发送给目标客户端。

具体实现

该聊天程序源码包括 `server.c` 和 `client.c` 文件，编译得到可执行程序 `server` 和 `client`。

源码文件大纲如下：



接下来描述具体实现过程。

服务器

首先，初始化 `Winsock` 库，设定 `Socket` 最高版本为2.2；

- 在 `Socket API` 调用的每一步都设置相应的错误处理，如果返回值非预期，则调用错误处理函数，打印错误信息，关闭创建的 `Socket`，释放 `Socket DLL` 资源，退出程序。

```
// 初始化Winsock库
WSADATA wsaData; // 存储Winsock库的初始化信息
if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0)
{
    error("WSAStartup");
}
```

随后，创建服务器套接字 `sock_server`，`AF_INET` 表示IPv4地址族，`SOCK_STREAM` 表示使用流式套接字，`IPPROTO_TCP` 表示使用TCP协议。

```
// 创建服务器套接字
sock_server = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
if (sock_server == INVALID_SOCKET)
{
    error("Socket creation");
}
```

初始化服务器地址结构体 `server_addr`，设置服务器IP地址和端口。然后使用 `bind` 函数将服务器套接字 `sock_server` 绑定到 `server_addr` 所指定的IP地址和端口号上。使用 `listen` 函数监听客户端的连接请求。

```
struct sockaddr_in server_addr;

memset(&server_addr, 0, sizeof(server_addr));
server_addr.sin_family = AF_INET;
// 设置服务器 IP 地址，将 IP 地址转为二进制整数
server_addr.sin_addr.s_addr = inet_addr("127.127.127.127");
// 设置服务器端口，小端转大端
server_addr.sin_port = htons(8000);

// 服务器套接字 sock_server 被绑定到 server_addr 所指定的IP地址和端口号上
if (bind(sock_server, (struct sockaddr *)&server_addr,
sizeof(server_addr)))
{
    error("Socket bind");
}

// 监听客户端连接请求
if (listen(sock_server, MAX_CLIENTS))
{
    error("Listen");
}
```

为了能够记录连接的多客户端，创建结构体，保存不同客户端信息：

```

struct client_information // 客户端信息结构体
{
    SOCKET socket;          // 客户端套接字
    struct sockaddr_in addr; // 客户端地址信息
    char *name;             // 客户端昵称
    int id;                 // 客户端id
};
struct client_information clients[MAX_CLIENTS];

```

创建 `while` 循环，不断等待客户端的连接请求，持续接受新的客户端连接。

- `accept` 函数会阻塞程序，直到有客户端尝试连接到服务器。一旦有连接请求到达，`accept` 函数会创建一个新的套接字，该套接字代表与客户端建立的连接，同时也会填充结构体以包含客户端的地址信息。
- 如果此时超过最大可连接客户端数量，则发送错误信息给客户端，中断连接；
- 对于每个连接的客户端，服务器页面打印时间戳和连接信息，计数器 `num_clients` 加一；
 - 使用 `CreateThread` 函数创建一个新线程，线程函数为 `handlerRequest`，并将客户端消息传递给线程函数。这个函数用于处理特定客户端的消息。每个线程对应一个客户端连接，因此可以同时处理多个客户端的消息，而不会相互干扰。

```

while (1)
{
    // 对每个客户端，等待连接请求，创建新的连接socket
    int client_addr_len = sizeof(clients[num_clients].addr);
    struct client_information tmp;
    tmp.socket = accept(sock_server, (struct sockaddr *)&tmp.addr,
&client_addr_len);
    clients[num_clients].socket = tmp.socket;
    clients[num_clients].addr = tmp.addr;

    if (clients[num_clients].socket == INVALID_SOCKET)
    {
        printf("Socket accept error");
        break;
    }
    else if (num_clients >= MAX_CLIENTS)
    {
        printf("Maximum number of clients reached. Not accepting new
connections.\n");
        // 发送错误消息给客户端
        const char *error_message = "Server is at maximum capacity. Try
again later.";
    }
}

```

```

        send(clients[num_clients].socket, error_message,
strlen(error_message), 0);

        // 关闭与该客户端的连接
        closesocket(clients[num_clients].socket);
    }
    else
    {
        id++;
        clients[num_clients].id = id;
        print_time();
        // 打印连接信息
        printf("[%s] Client connected from: %s:%d\n", timestamp,
inet_ntoa(clients[num_clients].addr.sin_addr),
ntohs(clients[num_clients].addr.sin_port));
        // 创建线程
        HANDLE hThreads = CreateThread(NULL, 0, handlerRequest,
&clients[num_clients], 0, NULL);
        if (hThreads == NULL)
        {
            printf("Thread creation error");
            closesocket(clients[num_clients].socket);
        }
        CloseHandle(hThreads);
        num_clients++;
    }
}

```

当客户端连接到服务器时，每个客户端都会被分配到一个独立的线程，该线程负责处理特定客户端的消息。 `handlerRequest` 函数是线程处理函数，它接受一个 `LPVOID` 类型的参数，通常用于传递客户端信息。在这个函数中，参数 `lparam` 被强制转换为指向 `struct client_information` 结构的指针，以便获取有关客户端的信息。

```

DWORD WINAPI handlerRequest(LPVOID lparam)
{
    struct client_information *clients_info = (struct client_information
*)lparam; // 客户端信息

    bytes_rcv = recv(clients_info->socket, recv_msg, sizeof(recv_msg) - 1,
0);
    recv_msg[bytes_rcv] = '\0';

    // 存放客户端的昵称
    clients_info->name = (char *)malloc(strlen(recv_msg) + 1);
    strcpy(clients_info->name, recv_msg);
}

```



```

    char sendBuf[] = "Welcome to the chat room. Enter your message and
    press Enter to send. Enter '*' and the content you wish to use for
    executing system commands. If you wanna chat to somebody, please input like
    '@00001:xxxxx'(length of id must be five)";
    send(clients_info->socket, sendBuf, strlen(sendBuf), 0);

    // 向所有已连接的其他客户端广播上线通知
    sprintf(broad_msg, "Client(%d) %s (%s:%d) is online!", clients_info-
    >id, clients_info->name, inet_ntoa(clients_info->addr.sin_addr),
    ntohs(clients_info->addr.sin_port));
    broadcast(clients_info);

    while (1)
    {
        // 接受客户端消息
        bytes_recv = recv(clients_info->socket, recv_msg, sizeof(recv_msg)
        - 1, 0);
        if (bytes_recv == SOCKET_ERROR || bytes_recv == 0)
        {
            handle_exit(clients_info); // 处理客户端退出
            break;                     // 退出循环，结束线程
        }
        recv_msg[bytes_recv] = '\0';

        // 打印从客户端接收到的消息
        print_time();
        printf("[%s] Received from %s(%s:%d): %s\n", timestamp,
        clients_info->name, inet_ntoa(clients_info->addr.sin_addr),
        ntohs(clients_info->addr.sin_port), recv_msg);

        // 判断消息类型
        char message_type = recv_msg[0];

        // 提取内容
        char *message_content = &recv_msg[1]; // 从第二个字符开始是消息内容

        if (message_type == 'T')
        {
            // 将接收到的消息发送给所有已连接的客户端
            sprintf(broad_msg, "(%d)%s(%s:%d): %s", clients_info->id,
            clients_info->name, inet_ntoa(clients_info->addr.sin_addr),
            ntohs(clients_info->addr.sin_port), message_content + 5);
            broadcast(clients_info);
        }
        else if (message_type == 'Q')
        {

```

```

        int obj = (message_content[0] - '0') * 10000 +
        (message_content[1] - '0') * 1000 + (message_content[2] - '0') * 100 +
        (message_content[3] - '0') * 10 + (message_content[4] - '0');
        if (obj > 0)
        {
            sprintf(broad_msg, "[private](%d) %s(%s:%d): %s",
clients_info->id, clients_info->name, inet_ntoa(clients_info-
>addr.sin_addr), ntohs(clients_info->addr.sin_port), message_content + 5);
            int i;

            EnterCriticalSection(&cs);    // 进入临界区
            for (i = 0; i < num_clients; i++)
            {
                if (clients[i].id == obj)
                    break;
            }
            if (i == num_clients)
            {
                char sendBuf[] = "Sent failed!Input the right ID!";
                send(clients_info->socket, sendBuf, strlen(sendBuf),
0);

                continue;
            }
            send(clients[i].socket, broad_msg, strlen(broad_msg), 0);
            LeaveCriticalSection(&cs);    // 离开临界区
        }
        else
        {
            char sendBuf[] = "Sent failed!Input the right ID!";
            send(clients_info->socket, sendBuf, strlen(sendBuf), 0);
            continue;
        }
    }
    return 0;
}

```

下面具体分析：

- 获取客户端信息：
 - 从传递给线程的参数 `lparam` 中获取客户端信息，并将其存储在 `clients_info` 变量中，包括客户端的 `socket` 和地址信息。
- 处理客户端连接的反馈：
 - 使用 `recv` 函数接收客户端发送的首条消息，即客户端的昵称。昵称保存在服务器，用于标识不同的客户端。

- 服务器向客户端发送欢迎消息。
- 服务器使用 `broadcast` 函数向所有已连接的客户端广播消息，宣布这个新客户端上线了。通知包括客户端的昵称和地址信息。
 - `broadcast` 函数遍历所有已连接的客户端，将一条消息广播给除消息的发送者外的所有已连接的客户端。

```

// 广播消息给所有客户端
void broadcast(struct client_information *clients_info)
{
    EnterCriticalSection(&cs); // 进入临界区
    // 遍历所有已连接的客户端
    for (int i = 0; i < num_clients; i++)
    {
        if (clients_info->socket != clients[i].socket &&
            clients_info->name != NULL)
        {
            // 发送消息
            send(clients[i].socket, broad_msg,
                strlen(broad_msg), 0);
        }
    }
    LeaveCriticalSection(&cs); // 离开临界区
}

```

- 进入消息处理循环，接收客户端发送的消息，并处理它们。
 - 接收客户端消息：使用 `recv` 函数接收客户端发送的消息，并存储在 `recv_msg` 中。如果接收到的消息出现错误（如客户端断开连接），则会使用 `handle_exit` 函数处理客户端退出，并退出循环，结束线程。
 - `handle_exit` 函数：打印断连信息，广播下线通知，在客户端数组中移除相应的客户端，计数器 `num_clients` 减一，并释放相关资源。

```

// 处理客户端退出
void handle_exit(struct client_information *clients_info)
{
    EnterCriticalSection(&cs); // 进入临界区
    // 打印客户端退出信息
    print_time();
    printf("[%s] Client disconnected: %s:%d\n", timestamp,
        inet_ntoa(clients_info->addr.sin_addr), ntohs(clients_info->
            >addr.sin_port));
    sprintf(broad_msg, "Client(%d) %s(%s:%d) is offline!",
        clients_info->id, clients_info->name, inet_ntoa(clients_info->
            >addr.sin_addr), ntohs(clients_info->addr.sin_port));
    broadcast(clients_info);
}

```

```

// 释放客户端名称内存
free(clients_info->name);
closesocket(clients_info->socket);
// 查找要删除的客户端
int i;
for (i = 0; i < num_clients; i++)
{
    if (clients[i].socket == clients_info->socket)
        break;
}
// 将要删除的客户端信息从数组中移除
if (i < num_clients)
{
    memmove(&clients[i], &clients[i + 1], (num_clients - i
- 1) * sizeof(struct client_information));
    num_clients--;
}
LeaveCriticalSection(&cs); // 离开临界区
}

```

- 处理客户端消息：打印从客户端接收到的消息，包括时间戳、客户端的昵称、IP地址、端口号以及消息内容。
 - 根据消息的类型（这里是 'T'，表示文本消息），决定采取的行为。
 - 如果接收到的消息是文本消息，服务器会提取消息内容，将消息格式化，包括发送者的昵称、IP地址、端口号以及消息内容，并使用 `broadcast` 函数广播给所有已连接的客户端。
 - 如果接收到的消息是私聊消息，服务器：
 - 提取目标客户端ID，在客户端信息数组中查找目标客户端及其socket；
 - 提取消息内容，向目标客户端发送消息内容。

服务器退出时，关闭套接字，清理Winsock库，结束程序。

▲ 服务器互斥锁的实现

首先，定义一个全局的 `CRITICAL_SECTION`：

```
CRITICAL_SECTION cs;
```

在 `main` 函数中进行初始化：

```

int main(){
    // ...
    InitializeCriticalSection(&cs); // 初始化互斥锁
}

```

```

while (1) {
    // ...
    // 创建线程
    // ...
}

DeleteCriticalSection(&cs);    // 释放互斥锁

closesocket(sock_server);
WSACleanup();
return 0;
}

```

然后，在需要保护临界区的地方使用互斥锁。在程序中，以下地方是需要加锁的：

1. `num_clients` 变量的读写操作，因为它在多个线程之间共享。
2. 对 `clients` 数组的读写操作，因为不同线程可能同时访问客户端信息。

在每个需要保护的地方，使用如下的方式添加互斥锁（具体参见前面服务器完整代码）：

```

EnterCriticalSection(&cs); // 进入临界区

// 执行需要保护的操作

LeaveCriticalSection(&cs); // 离开临界区

```

客户端

客户端同样首先初始化 `Winsock` 库并创建 `Socket`，将 `Socket` 绑定服务器的IP和端口。

```

WSADATA wsaData;
if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0)
{
    error("WSAStartup");
}

// 创建socket
sock_client = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
if (sock_client == INVALID_SOCKET)
{
    error("Socket creation");
}

struct sockaddr_in server_addr;

```

```
memset(&server_addr, 0, sizeof(server_addr));
server_addr.sin_family = AF_INET;
server_addr.sin_addr.s_addr = inet_addr("127.127.127.127"); // 设置服务器地址
server_addr.sin_port = htons(8000); // 设置服务器端口
```

随后，客户端要求用户输入昵称，并保存起来。

```
// 输入昵称
printf("Please enter your name: ");
char name[256];
fgets(name, sizeof(name), stdin);
name[strlen(name) - 1] = '\0';
```

客户端向服务器发送连接请求，连接成功后，首先发送客户端设定的昵称。

```
// 连接服务器
if (connect(sock_client, (SOCKADDR *)&server_addr, sizeof(server_addr)))
{
    error("Connect");
}

// 发送昵称
int bytes_sent = send(sock_client, name, strlen(name), 0);
if (bytes_sent == SOCKET_ERROR)
{
    error("Send");
}
```

之后，客户端可以随时对服务器收发信息。首先创建一个线程，处理从服务器接收的消息。

```
// 创建线程
HANDLE hThread = CreateThread(NULL, 0,
(LPTHREAD_START_ROUTINE)handlerRequest, NULL, 0, NULL);
```

线程处理函数如下。创建 `while` 循环接收客户端发来的信息，附加时间戳打印在页面上，形如： `[时间戳] 服务器消息内容` 。

```
void handlerRequest(void *lpParam)
{
    char buffer[256];
```

```

while (1)
{
    int bytes_received = recv(sock_client, buffer, sizeof(buffer), 0);
    if (bytes_received == SOCKET_ERROR)
    {
        error("Receive");
    }
    else
    {
        buffer[bytes_received] = '\0';
        print_time();
        printf("[%s] %s\n", timestamp, buffer);
    }
}
}

```

而对于客户端的发送消息，在主线程中，进入 `while` 循环，等待用户输入信息。

- 读取用户输入信息，保存到 `message` 数组中。如果用户输入的消息以换行符(`\n`)结尾，将这个换行符替换为字符串终止符(`\0`)。
- 动态分配内存来构建消息结构体(`msg`)。

```

// 定义报文结构体
struct Message
{
    char type; // 标识消息类型
    char obj[5];
    char content[1]; // 先使用长度为 1 的数组，实际长度会动态分配
};

```

- 消息结构体包括消息类型、目标对象内容。
- 使用 `malloc` 函数分配足够的内存来存储消息结构体及其内容。
- 设置消息结构体的类型(`msg->type`)。
 - 默认情况下，消息的类型被设置为文本（'T'）。
- 如果用户输入的消息以 `*` 字符开头，将消息的类型设置为系统消息（'S'）并将内容从用户输入消息中复制（跳过第一个字符 `*`）。
- 如果用户输入的信息以 `@` 字符开头，则消息的类型为私聊类型
 - 提取目标客户端id和消息文本内容，构造 `message` 结构体
- 其他情况下，将消息内容直接从用户输入消息中复制
- 如果消息的类型是系统消息（'S'）且消息内容是"exit"，则：

- 调用 `TerminateThread` 函数终止处理服务器消息的线程（`hThread`）。
- 关闭线程句柄，确保线程资源被正确释放。
- 释放动态分配的消息结构体内存。
- 退出循环。
- 最后，使用 `send` 函数将消息结构体发送给服务器。
 - 如果发送成功，打印发送时间、消息类型和内容，并释放动态分配的消息结构体内存。

```
while (1)
{
    char message[256];

    if (fgets(message, sizeof(message), stdin) != NULL)
    {
        size_t len = strlen(message);
        if (len > 0 && message[len - 1] == '\n')
        {
            message[len - 1] = '\0';
        }
    }

    // 构建消息结构体
    size_t message_len = strlen(message);
    struct Message *msg = (struct Message *)malloc(sizeof(struct Message) +
message_len - 1);
    if (!msg)
    {
        error("Memory allocation");
    }

    msg->type = 'T'; // 默认为文本类型
    msg->obj[0] = '\0';
    msg->obj[1] = '\0';
    msg->obj[2] = '\0';
    msg->obj[3] = '\0';
    msg->obj[4] = '\0';
    if (message[0] == '*')
    {
        msg->type = 'S'; // 类型为系统类型
        strcpy(msg->content, message + 1); // 文本内容，跳过第一个字符 "*"
    }
    else if (message[0] == '@')
    {
        msg->type = 'Q'; // 类型为私聊类型
    }
}
```



```

char *at_position = strchr(message, '@');
char *colon_position = strchr(message, ':');
if (at_position && colon_position && at_position < colon_position)
{
    // 计算要提取的字符的长度
    int length = colon_position - at_position - 1;

    if (length == 5)
    {
        strncpy(msg->obj, at_position + 1, length);
        strcpy(msg->content, message + 7);
    }
    else
    {
        printf("Input error.\n");
        free(msg);
        continue;
    }
}
else
{
    printf("Input error.\n");
    free(msg);
    continue;
}
}
else
{
    if (strlen(message) < 1)
    {
        printf("Please input the message.\n");
        free(msg);
        continue;
    }
    strcpy(msg->content, message);
}

if (msg->type == 'S' && strcmp(msg->content, "exit") == 0)
{
    TerminateThread(hThread, 0);
    CloseHandle(hThread);
    free(msg); // 释放动态分配的内存
    break;
}

// 发送消息结构体

```

```

    int bytes_sent = send(sock_client, (const char *)msg, sizeof(struct
Message) + message_len - 1, 0);

    if (bytes_sent == SOCKET_ERROR)
    {
        error("Send");
    }
    else
    {
        print_time();
        printf("[%s] Sent %d bytes to the server: %s\n", timestamp,
bytes_sent, msg);
    }

    free(msg); // 释放动态分配的内存
}

```

客户端退出后，关闭套接字，释放相关资源。

程序测试

编译写好的程序：

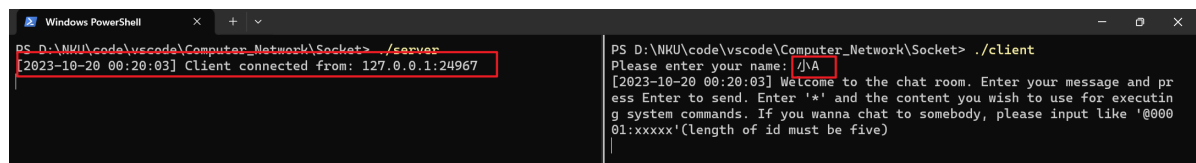
```

PS D:\NKU\code\vscode\Computer_Network\Socket> gcc server.c -o server -lwsck32
PS D:\NKU\code\vscode\Computer_Network\Socket> gcc client.c -o client -lwsck32

```

首先启动服务器，然后启动客户端。

输入用户名后，连接服务器成功。



```

Windows PowerShell
PS D:\NKU\code\vscode\Computer_Network\Socket> ./server
[2023-10-20 00:20:03] Client connected from: 127.0.0.1:24967

PS D:\NKU\code\vscode\Computer_Network\Socket> ./client
Please enter your name: J\A
[2023-10-20 00:20:03] Welcome to the chat room. Enter your message and pr
ess Enter to send. Enter '*' and the content you wish to use for executin
g system commands. If you wanna chat to somebody, please input like '@000
01:xxxxx'(length of id must be five)

```

开启多个客户端，已上线客户端能收到新客户端的上线提醒。

<pre>PS D:\NKU\code\vscode\Computer_Network\Socket> ./server [2023-10-20 00:20:03] Client connected from: 127.0.0.1:24967 [2023-10-20 00:20:48] Client connected from: 127.0.0.1:24994 [2023-10-20 00:20:58] Client connected from: 127.0.0.1:57717</pre>	<pre>PS D:\NKU\code\vscode\Computer_Network\Socket> ./client Please enter your name: 小A [2023-10-20 00:20:03] Welcome to the chat room. Enter your message and press Enter to send. Enter '*' and the content you wish to use for executing system commands. If you wanna chat to somebody, please input like '@00001:xxxxx'(length of id must be five) [2023-10-20 00:20:48] Client(2) 小B (127.0.0.1:24994) is online! [2023-10-20 00:20:58] Client(3) 小C (127.0.0.1:57717) is online!</pre>
服务器信息打印	其他用户上线提醒
<pre>PS D:\NKU\code\vscode\Computer_Network\Socket> ./client Please enter your name: 小B [2023-10-20 00:20:03] Welcome to the chat room. Enter your message and press Enter to send. Enter '*' and the content you wish to use for executing system commands. If you wanna chat to somebody, please input like '@00001:xxxxx'(length of id must be five) [2023-10-20 00:20:58] Client(3) 小C (127.0.0.1:57717) is online!</pre>	<pre>PS D:\NKU\code\vscode\Computer_Network\Socket> ./client Please enter your name: 小C [2023-10-20 00:20:58] Welcome to the chat room. Enter your message and press Enter to send. Enter '*' and the content you wish to use for executing system commands. If you wanna chat to somebody, please input like '@00001:xxxxx'(length of id must be five)</pre>

多客户端之间能够共同聊天，能够传输中英文文本，能够正确构造 `message` 报文结构体，没有发现数据的丢失。

<pre>PS D:\NKU\code\vscode\Computer_Network\Socket> ./server [2023-10-20 00:20:03] Client connected from: 127.0.0.1:24967 [2023-10-20 00:20:48] Client connected from: 127.0.0.1:24994 [2023-10-20 00:20:58] Client connected from: 127.0.0.1:57717 [2023-10-20 00:21:45] Received from 小A(127.0.0.1:24967): T00000hello [2023-10-20 00:21:52] Received from 小B(127.0.0.1:24994): T00000早上好 [2023-10-20 00:21:55] Received from 小C(127.0.0.1:57717): T00000晚上好 [2023-10-20 00:22:03] Received from 小A(127.0.0.1:24967): T00000下午好啊！！！！</pre>	<pre>PS D:\NKU\code\vscode\Computer_Network\Socket> ./client Please enter your name: 小A [2023-10-20 00:20:03] Welcome to the chat room. Enter your message and press Enter to send. Enter '*' and the content you wish to use for executing system commands. If you wanna chat to somebody, please input like '@00001:xxxxx'(length of id must be five) [2023-10-20 00:20:48] Client(2) 小B (127.0.0.1:24994) is online! [2023-10-20 00:20:58] Client(3) 小C (127.0.0.1:57717) is online! hello [2023-10-20 00:21:45] Sent 11 bytes to the server: T00000hello [2023-10-20 00:21:52] (2)小B(127.0.0.1:24994): 早上好 [2023-10-20 00:21:55] (3)小C(127.0.0.1:57717): 晚上好 下午好啊！！！！ [2023-10-20 00:22:03] Sent 22 bytes to the server: T00000下午好啊！！！！</pre>
服务器打印聊天信息	发送给服务器的信息构成：信息类型+target id(默认为0)+文本内容
能够共同聊天	
<pre>PS D:\NKU\code\vscode\Computer_Network\Socket> ./client Please enter your name: 小B [2023-10-20 00:20:03] Welcome to the chat room. Enter your message and press Enter to send. Enter '*' and the content you wish to use for executing system commands. If you wanna chat to somebody, please input like '@00001:xxxxx'(length of id must be five) [2023-10-20 00:20:58] Client(3) 小C (127.0.0.1:57717) is online! [2023-10-20 00:21:45] (1)小A(127.0.0.1:24967): hello 早上好 [2023-10-20 00:21:52] Sent 12 bytes to the server: T00000早上好 [2023-10-20 00:21:55] (3)小C(127.0.0.1:57717): 晚上好 [2023-10-20 00:22:03] (1)小A(127.0.0.1:24967): 下午好啊！！！！</pre>	<pre>PS D:\NKU\code\vscode\Computer_Network\Socket> ./client Please enter your name: 小C [2023-10-20 00:20:58] Welcome to the chat room. Enter your message and press Enter to send. Enter '*' and the content you wish to use for executing system commands. If you wanna chat to somebody, please input like '@00001:xxxxx'(length of id must be five) [2023-10-20 00:21:45] (1)小A(127.0.0.1:24967): hello [2023-10-20 00:21:52] (2)小B(127.0.0.1:24994): 早上好 晚上好 [2023-10-20 00:21:55] Sent 12 bytes to the server: T00000晚上好 [2023-10-20 00:22:03] (1)小A(127.0.0.1:24967): 下午好啊！！！！</pre>

能够正确进入私聊模式，服务器能够成功检索到目标客户端，将消息单独发给目标客户端。

<pre>PS D:\NKU\code\vscode\Computer_Network\Socket> ./server [2023-10-20 00:20:03] Client connected from: 127.0.0.1:24967 [2023-10-20 00:20:48] Client connected from: 127.0.0.1:24994 [2023-10-20 00:20:58] Client connected from: 127.0.0.1:57717 [2023-10-20 00:21:45] Received from 小A(127.0.0.1:24967): T0000hello [2023-10-20 00:21:52] Received from 小B(127.0.0.1:24994): T0000早上好 [2023-10-20 00:21:55] Received from 小C(127.0.0.1:57717): T0000晚上好 [2023-10-20 00:22:03] Received from 小A(127.0.0.1:24967): T0000下午好啊 !!!! [2023-10-20 00:23:50] Received from 小A(127.0.0.1:24967): Q0000我在单独和你聊天 [2023-10-20 00:23:59] Received from 小B(127.0.0.1:24994): Q00001哈哈哈哈哈我收到了</pre>	<pre>PS D:\NKU\code\vscode\Computer_Network\Socket> ./client Please enter your name: 小A [2023-10-20 00:20:03] Welcome to the chat room. Enter your message and press Enter to send. Enter '*' and the content you wish to use for executing system commands. If you wanna chat to somebody, please input like '@00001:xxxxx'(length of id must be five) [2023-10-20 00:20:48] Client(2) 小B (127.0.0.1:24994) is online! [2023-10-20 00:20:58] Client(3) 小C (127.0.0.1:57717) is online! hello [2023-10-20 00:21:45] Sent 11 bytes to the server: T0000hello [2023-10-20 00:21:52] (2)小B(127.0.0.1:24994): 早上好 [2023-10-20 00:21:55] (3)小C(127.0.0.1:57717): 晚上好 下午好啊!!!! [2023-10-20 00:22:03] Sent 22 bytes to the server: T0000下午好啊!!!! @00002:我在单独和你聊天 [2023-10-20 00:23:50] Sent 29 bytes to the server: Q00002我在单独和你聊天 [2023-10-20 00:23:59] [private](2) 小B(127.0.0.1:24994): 哈哈哈哈哈我收到了</pre> <p>输入@进入私聊模式，配上id及聊天内容 能够和指定的用户聊天，而其他用户收不到</p>
<pre>PS D:\NKU\code\vscode\Computer_Network\Socket> ./client Please enter your name: 小B [2023-10-20 00:20:48] Welcome to the chat room. Enter your message and press Enter to send. Enter '*' and the content you wish to use for executing system commands. If you wanna chat to somebody, please input like '@00001:xxxxx'(length of id must be five) [2023-10-20 00:20:58] Client(3) 小C (127.0.0.1:57717) is online! [2023-10-20 00:21:45] (1)小A(127.0.0.1:24967): hello 早上好 [2023-10-20 00:21:52] Sent 12 bytes to the server: T0000早上好 [2023-10-20 00:21:55] (3)小C(127.0.0.1:57717): 晚上好 [2023-10-20 00:22:03] (1)小A(127.0.0.1:24967): 下午好啊!!!! [2023-10-20 00:23:50] [private](1) 小A(127.0.0.1:24967): 我在单独和你聊天 @00001:哈哈哈哈哈我收到了 [2023-10-20 00:23:59] Sent 29 bytes to the server: Q00001哈哈哈哈哈我收到了</pre>	<pre>PS D:\NKU\code\vscode\Computer_Network\Socket> ./client Please enter your name: 小C [2023-10-20 00:20:58] Welcome to the chat room. Enter your message and press Enter to send. Enter '*' and the content you wish to use for executing system commands. If you wanna chat to somebody, please input like '@00001:xxxxx'(length of id must be five) [2023-10-20 00:21:45] (1)小A(127.0.0.1:24967): hello 早上好 [2023-10-20 00:21:52] (2)小B(127.0.0.1:24994): 早上好 [2023-10-20 00:21:55] Sent 12 bytes to the server: T0000晚上好 [2023-10-20 00:22:03] (1)小A(127.0.0.1:24967): 下午好啊!!!!</pre>

客户端输入*exit能够成功退出，其他用户能够收到下线提醒。

<pre>PS D:\NKU\code\vscode\Computer_Network\Socket> ./server [2023-10-20 00:20:03] Client connected from: 127.0.0.1:24967 [2023-10-20 00:20:48] Client connected from: 127.0.0.1:24994 [2023-10-20 00:20:58] Client connected from: 127.0.0.1:57717 [2023-10-20 00:21:45] Received from 小A(127.0.0.1:24967): T0000hello [2023-10-20 00:21:52] Received from 小B(127.0.0.1:24994): T0000早上好 [2023-10-20 00:21:55] Received from 小C(127.0.0.1:57717): T0000晚上好 [2023-10-20 00:22:03] Received from 小A(127.0.0.1:24967): T0000下午好啊 !!!! [2023-10-20 00:23:50] Received from 小A(127.0.0.1:24967): Q00002我在单独和你聊天 [2023-10-20 00:23:59] Received from 小B(127.0.0.1:24994): Q00001哈哈哈哈哈我收到了 [2023-10-20 00:25:47] Client disconnected: 127.0.0.1:24967</pre> <p>显示客户端退出</p>	<pre>PS D:\NKU\code\vscode\Computer_Network\Socket> ./client Please enter your name: 小A [2023-10-20 00:20:03] Welcome to the chat room. Enter your message and press Enter to send. Enter '*' and the content you wish to use for executing system commands. If you wanna chat to somebody, please input like '@00001:xxxxx'(length of id must be five) [2023-10-20 00:20:48] Client(2) 小B (127.0.0.1:24994) is online! [2023-10-20 00:20:58] Client(3) 小C (127.0.0.1:57717) is online! hello [2023-10-20 00:21:45] Sent 11 bytes to the server: T0000hello [2023-10-20 00:21:52] (2)小B(127.0.0.1:24994): 早上好 [2023-10-20 00:21:55] (3)小C(127.0.0.1:57717): 晚上好 下午好啊!!!! [2023-10-20 00:22:03] Sent 22 bytes to the server: T0000下午好啊!!!! @00002:我在单独和你聊天 [2023-10-20 00:23:50] Sent 29 bytes to the server: Q00002我在单独和你聊天 [2023-10-20 00:23:59] [private](2) 小B(127.0.0.1:24994): 哈哈哈哈哈我收到了 *exit Bye! PS D:\NKU\code\vscode\Computer_Network\Socket> </pre> <p>输入*进入系统模式，输入exit退出聊天客户端</p>
<pre>PS D:\NKU\code\vscode\Computer_Network\Socket> ./client Please enter your name: 小B [2023-10-20 00:20:48] Welcome to the chat room. Enter your message and press Enter to send. Enter '*' and the content you wish to use for executing system commands. If you wanna chat to somebody, please input like '@00001:xxxxx'(length of id must be five) [2023-10-20 00:20:58] Client(3) 小C (127.0.0.1:57717) is online! [2023-10-20 00:21:45] (1)小A(127.0.0.1:24967): hello 早上好 [2023-10-20 00:21:52] Sent 12 bytes to the server: T0000早上好 [2023-10-20 00:21:55] (3)小C(127.0.0.1:57717): 晚上好 [2023-10-20 00:22:03] (1)小A(127.0.0.1:24967): 下午好啊!!!! [2023-10-20 00:23:50] [private](1) 小A(127.0.0.1:24967): 我在单独和你聊天 @00001:哈哈哈哈哈我收到了 [2023-10-20 00:23:59] Sent 29 bytes to the server: Q00001哈哈哈哈哈我收到了 [2023-10-20 00:25:47] Client(1) 小A(127.0.0.1:24967) is offline!</pre>	<pre>PS D:\NKU\code\vscode\Computer_Network\Socket> ./client Please enter your name: 小C [2023-10-20 00:20:58] Welcome to the chat room. Enter your message and press Enter to send. Enter '*' and the content you wish to use for executing system commands. If you wanna chat to somebody, please input like '@00001:xxxxx'(length of id must be five) [2023-10-20 00:21:45] (1)小A(127.0.0.1:24967): hello 早上好 [2023-10-20 00:21:52] (2)小B(127.0.0.1:24994): 早上好 [2023-10-20 00:21:55] Sent 12 bytes to the server: T0000晚上好 [2023-10-20 00:22:03] (1)小A(127.0.0.1:24967): 下午好啊!!!! [2023-10-20 00:25:47] Client(1) 小A(127.0.0.1:24967) is offline!</pre> <p>其他用户收到下线提醒</p>

一些反馈

使用Ctrl+C强制下线服务器，客户端会得到提醒，并随之下线。

```
Windows PowerShell
PS D:\NKU\code\vscode\Computer_Network\Socket> ./server
[2023-10-20 12:27:10] Client connected from: 127.0.0.1:57660
[2023-10-20 12:27:11] Received from 小A(127.0.0.1:57660): T00000hi
[2023-10-20 12:27:17] Received from 小A(127.0.0.1:57660): T00000服务器不会崩吧?
PS D:\NKU\code\vscode\Computer_Network\Socket>

PS D:\NKU\code\vscode\Computer_Network\Socket> ./client
Please enter your name: 小A
[2023-10-20 12:27:10] Welcome to the chat room. Enter your message and press Enter to send. Enter '*' and the content you wish to use for executing system commands. If you wanna chat to somebody, please input like '@00001:xxxxx'(length of id must be five)
hi
[2023-10-20 12:27:11] Sent 8 bytes to the server: T00000hi
服务器不会崩吧?
[2023-10-20 12:27:17] Sent 22 bytes to the server: T00000服务器不会崩吧?
Receive failed with error 10054
PS D:\NKU\code\vscode\Computer_Network\Socket>
```

如果使用Ctrl+C强制下线服务器，客户端会相应得到提醒并下线

如果私聊输入不规范和目标ID不存在，客户端会收到相应提醒。

```
Windows PowerShell
PS D:\NKU\code\vscode\Computer_Network\Socket> ./server
[2023-10-20 12:21:55] Client connected from: 127.0.0.1:57513
[2023-10-20 12:21:59] Client connected from: 127.0.0.1:57519
[2023-10-20 12:22:03] Received from 小A(127.0.0.1:57513): T00000大家好
[2023-10-20 12:22:05] Received from 小B(127.0.0.1:57519): T00000你好
[2023-10-20 12:22:13] Received from 小A(127.0.0.1:57513): Q00002给你发的
[2023-10-20 12:22:28] Received from 小A(127.0.0.1:57513): Q10000回到

PS D:\NKU\code\vscode\Computer_Network\Socket> ./client
Please enter your name: 小A
[2023-10-20 12:21:55] Welcome to the chat room. Enter your message and press Enter to send. Enter '*' and the content you wish to use for executing system commands. If you wanna chat to somebody, please input like '@00001:xxxxx'(length of id must be five)
[2023-10-20 12:21:59] Client(2) 小B (127.0.0.1:57519) is online!
大家好
[2023-10-20 12:22:03] Sent 12 bytes to the server: T00000大家好
[2023-10-20 12:22:05] (2)小B(127.0.0.1:57519): 你好
@00002:给你发的
[2023-10-20 12:22:13] Sent 21 bytes to the server: Q00002给你发的
@01是吗
Input error
@10000:回到
[2023-10-20 12:22:28] Sent 17 bytes to the server: Q10000回到
[2023-10-20 12:22:28] Sent failed!Input the right ID
@19219: 稍等
Input error.

PS D:\NKU\code\vscode\Computer_Network\Socket> ./client
Please enter your name: 小B
[2023-10-20 12:21:59] Welcome to the chat room. Enter your message and press Enter to send. Enter '*' and the content you wish to use for executing system commands. If you wanna chat to somebody, please input like '@00001:xxxxx'(length of id must be five)
[2023-10-20 12:22:03] (1)小A(127.0.0.1:57513): 大家好
你好
[2023-10-20 12:22:05] Sent 10 bytes to the server: T00000你好
[2023-10-20 12:22:13] [private](1) 小A(127.0.0.1:57513): 给你发的
```

私聊机制的错误处理：输入规范和正确ID

无法发送空白信息。

```
Windows PowerShell
PS D:\NKU\code\vscode\Computer_Network\Socket> ./server
[2023-10-20 12:33:16] Client connected from: 127.0.0.1:57813
[2023-10-20 12:33:19] Received from 小a(127.0.0.1:57813): T00000aa
[2023-10-20 12:33:20] Received from 小a(127.0.0.1:57813): T00000aaaa

g system commands. If you wanna chat to somebody, please input like '@00001:xxxxx'(length of id must be five)
a
[2023-10-20 12:33:19] Sent 7 bytes to the server: T00000aa
aaa
[2023-10-20 12:33:20] Sent 9 bytes to the server: T00000aaaa

Please input the message.
Please input the message.
Please input the message.
Please input the message.
Please input the message.
Please input the message.
Please input the message.
```

无法发送空白信息

对于客户端容量的限制，将最大容量修改为2重新编译并运行程序。可以看到，在已有两个客户端连接的情况下，第三个客户端无法连接。下线第一个客户端，第三个客户端即可连接。

```
PS D:\NKU\code\vscode\Computer_Network\Socket> ./server
[2023-10-20 12:28:29] Client connected from: 127.0.0.1:57681
[2023-10-20 12:28:35] Received from 小A(127.0.0.1:57681): T00000我是小A, 我来了
[2023-10-20 12:28:40] Client connected from: 127.0.0.1:57684
[2023-10-20 12:28:46] Received from 小B(127.0.0.1:57684): T00000我是小B, 我也来啦
Maximum number of clients reached. Not accepting new connections.
[2023-10-20 12:29:03] Received from 小A(127.0.0.1:57681): T00000小C无法登录, 我先下线
[2023-10-20 12:29:06] Client disconnected: 127.0.0.1:57681
[2023-10-20 12:29:13] Client connected from: 127.0.0.1:57698
[2023-10-20 12:29:18] Received from 小C(127.0.0.1:57698): T00000我成功进来了

PS D:\NKU\code\vscode\Computer_Network\Socket> ./client
Please enter your name: 小A
[2023-10-20 12:28:29] Welcome to the chat room. Enter your message and press Enter to send. Enter '*' and the content you wish to use for executing system commands. If you wanna chat to somebody, please input like '@00001:xxxxx'(length of id must be five)
我是小A, 我来了
[2023-10-20 12:28:35] Sent 21 bytes to the server: T00000我是小A, 我来了
[2023-10-20 12:28:40] Client(2) 小B (127.0.0.1:57684) is online!
[2023-10-20 12:28:46] (2)小B(127.0.0.1:57684): 我是小B, 我也来啦
小C无法登录, 我先下线
[2023-10-20 12:29:03] Sent 27 bytes to the server: T00000小C无法登录, 我先下线
*exit
Bye!
PS D:\NKU\code\vscode\Computer_Network\Socket>

PS D:\NKU\code\vscode\Computer_Network\Socket> ./client
Please enter your name: 小C
[2023-10-20 12:28:52] Server is at maximum capacity. Try again later.
Receive failed with error 10054
PS D:\NKU\code\vscode\Computer_Network\Socket> ./client
Please enter your name: 小C
[2023-10-20 12:29:13] Welcome to the chat room. Enter your message and press Enter to send. Enter '*' and the content you wish to use for executing system commands. If you wanna chat to somebody, please input like '@00001:xxxxx'(length of id must be five)
我是小B, 我也来啦
[2023-10-20 12:29:03] (1)小A(127.0.0.1:57681): 小C无法登录, 我先下线
[2023-10-20 12:29:06] Client(1) 小A(127.0.0.1:57681) is offline!
[2023-10-20 12:29:13] Client(3) 小C (127.0.0.1:57698) is online!
[2023-10-20 12:29:18] (3)小C(127.0.0.1:57698): 我成功进来了
```

在这里, 将服务器容量限制为2
最多允许2个客户端连接

服务器容量限制

在已有2个客户端连接的情况下, 第3个客户端无法连接, 得到错误提醒

将第一个客户端下线, 现在能够再接纳一个客户端

wireshark捕获

发送一条消息, 在wireshark中捕获。

```
PS D:\NKU\code\vscode\Computer_Network\Socket> ./server
[2023-10-20 00:27:48] Client connected from: 127.0.0.1:19408
[2023-10-20 00:28:08] Received from 小A(127.0.0.1:19408): T00000Can you see it in WireShark?

PS D:\NKU\code\vscode\Computer_Network\Socket> ./client
Please enter your name: 小A
[2023-10-20 00:27:48] Welcome to the chat room. Enter your message and press Enter to send. Enter '*' and the content you wish to use for executing system commands. If you wanna chat to somebody, please input like '@00001:xxxxx'(length of id must be five)
Can you see it in WireShark?
[2023-10-20 00:28:08] Sent 34 bytes to the server: T00000Can you see it in WireShark?
```

可以捕获到这条分组:

Wireshark · 分组 292 · Adapter for loopback traffic capture

> Frame 292: 90 bytes on wire (720 bits), 90 bytes captured (720 bits) on interface \Device\NPF_{...} Null/Loopback

> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.127.127.127

> Transmission Control Protocol, Src Port: 19408, Dst Port: 8000, Seq: 1, Ack: 1, Len: 34

> Data (34 bytes)

Data: 54303030303043616e207966752073656520697420696e20576972265536861726b3f [Length: 34]

0000 02 00 00 00 45 00 00 56 ad 4f 40 00 40 06 00 00E..V..O@..@...

0010 7f 00 00 01 7f 7f 7f 7f 4b d0 1f 40 9c c1 08 a9K..@....

0020 16 58 86 f5 80 18 04 fe 4c 9d 00 00 01 01 08 0a ..X.....L.....

0030 03 87 98 6b 03 87 4a a7 54 30 30 30 30 30 43 61 ...k..J..T00000Ca

0040 6e 20 79 6f 75 20 73 65 65 20 69 74 20 69 6e 20 n you see it in

0050 57 69 72 65 53 68 61 72 6b 3f WireShark?

Show packet bytes

Close Help

关于数据丢失

在聊天程序的测试过程中和wireshark捕获中，都没有发现数据丢失的现象。在本次实验中，数据传输协议为TCP协议。TCP协议提供了一些机制来保证数据的完整性和可靠性，例如数据的序列化、确认机制、重传机制等，这能保证数据的可靠传输。

当然，由于数据缓冲区大小固定，聊天协议中具有传输的消息长度限制，如果超出了这个长度，就无法正确发送完整信息，导致数据丢失。