南開大學
Nankai University

南　开　大　学

网络空间安全学院

计算机网络课程报告

第一次实验报告

学号：2011897

姓名：任薏霖

年级：2020 级

专业：物联网工程

2022 年 10 月 21 日

一、实验要求

（一）使用流式 Socket，设计一个两人聊天协议，要求聊天信息带有时间标签。请完整地说明交互消息的类型、语法、语义、时序等具体的消息处理方式。

（二）对聊天程序进行设计。给出模块划分说明、模块的功能和模块的流程图。

（三）在 Windows 系统下，利用 C/C++对设计的程序进行实现。程序界面可以采用命令行方式，但需要给出使用方法。编写程序时，只能使用基本的 Socket 函数，不允许使用对 socket 封装后的类或架构。

（四）对实现的程序进行测试。

（五）撰写实验报告，并将实验报告和源码提交至本网站。

二、实验内容

（一）基本功能概述

1. 多人聊天通信，最多支持 10 人；
2. TCP 协议，流式套接字；
3. 支持包括但不限于中文、英文等发送；
4. 合理的用户上线下线机制；
5. 正常的用户退出方式。

（二）聊天协议

1. 消息的类型

消息分为普通消息、系统消息两种。

普通消息为用户之间聊天的内容消息；系统消息为聊天室标志、用户操作、服务器端和客户端的状态相互影响而返回的消息，其中包含动作消息、时间信息等，例如服务器开启监听状态时则返回类型为“SUCCESS”等信息、用户退出聊天室时显示的“程序将于 3 秒内退出”信息等。

2. 消息的语法

报文包括三个数据段，从高位到低位，第一个数据段表示发送/接收时间，第二个数据段表示发送者的昵称，第三段表示发送的信息。各数据段长度有限，若超出范围会导致信息丢失。并以键入换行符作为消息发送热键，在消息处理时通过“换行符”执行换行操作并打印信息。

报文结构如下所示：

```
发送时间:    char now_time[32];
发送人:      char username[16];//客户名
消息:        char buffer[256];
```

3. 消息的语义

消息的语义具体包含三部分，分别为时间、昵称、信息内容。其中时间包含日期和时刻两部分；昵称部分与时间部分通过“ ”（空格）相连；信息内容与昵称部分通过“:”（冒号）相连。

打印消息具体如下：

```
例:
2022-10-22 18:46:45  2011897:  任薏霖2011897
```

三、实验过程

（一）程序运行环境

本程序在 VisualStudio2019 下，通过 C++编写下完成。

使用的链接库如下所示：

```
#include<iostream>
#include<winsock2.h>
#include<string.h>
#pragma comment(lib, "ws2_32.lib")
#pragma warning(disable:4996)
```

（二）程序设计

1. 思路

多线程实现多人聊天

2. 设计细节

(1) 服务器端

为同时支持多个 client 在线，采用多线程模式，即使用 CreateThread 随机创建新的线程。在此线程的入口地址中处理对应的 client 请求。

在创建完新线程后，在线程中接收 client 发送的报文信息，若成功收到客户端信息，服务器端将打印信息从而形成聊天日志。

同时服务器端将进行转发操作，识别出具体发送消息的客户端并将此进程下从该客户端收到的消息转发到其他的客户端，使得其他客户端也显示出该客户端的消息。

(2) 客户端

客户端设立发送消息和接收消息的双线程。

在与服务器端完成连接后，在接收线程中接收服务器端转发的报文信息，同时在客户端显示已经收到该信息并将该信息的具体细节显示出来。

当一方客户端输入需要发送的消息时，ThreadSend 函数会将发送消息时的时间，发送人，和具体消息的信息进行打包，并将该消息整体发送到服务器端，由服务器端接收并转发给其余客户端,其余客户端通过 ThreadRecv 进行接收。

由于 send 和 recv 在两个不同的线程之中，因此不会造成干扰。

3. 模块流程图

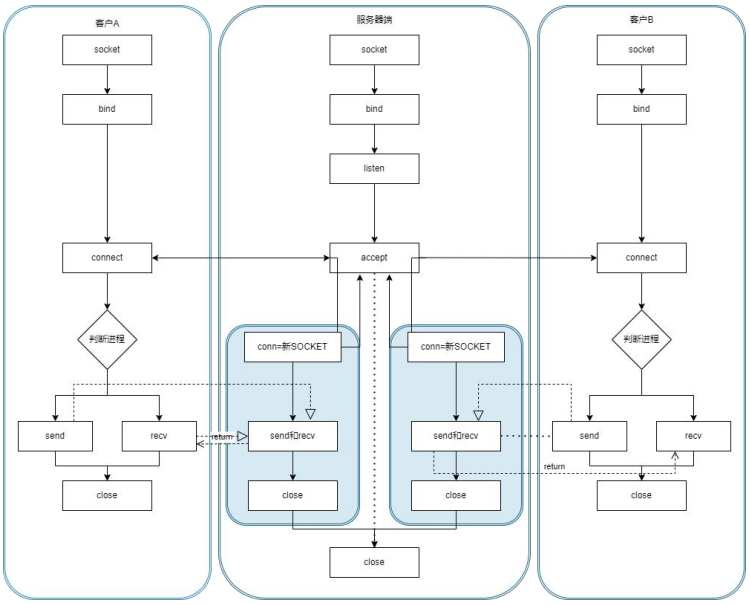

图 1 程序流程图

（三）代码模块详解

1. 服务器端

(1) 客户端结构体

定义客户端结构体分别由客户端套接字，客户名，消息拼接缓存，客户端标号和时间构成。

```
typedef struct CLIENT{
    SOCKET client;//客户端套接字
    char username[16];//客户名
    char buf[256];//消息拼接缓存
    int flag;//当前客户端标号
    char now_time[32];
}CLIENT;
```

(2) 时间输出函数

```
string getTime() {
    //获取系统时间
    time_t timep;
    time(&timep);
    char tmp[64];
    strftime(tmp, sizeof(tmp), "%Y-%m-%d %H:%M:%S", localtime(&timep));
    return tmp;
}
```

(3) 消息接收及转发模块

可通过创建死循环阻塞程序运行，若收到的信息合理，则将信息发送给除发送信息的客户端以外的其他客户端，达到信息转发的功能。

```
DWORD WINAPI AnswerThread(LPVOID lparam)
{while (1)
    {if (ret > 0)
        { for (i = 0; i < num; i++)
            {
                if (i != Client->flag)
                    send(chat[i].client, Client->buf, strlen(Client->buf), 0);
            }
        }
    }
}
```

(4) 主函数

首先需要初始化创建一个 socket 套接字 sockSrv

```
WORD sockVersion = MAKEWORD(2, 2);
    WSADATA wsaData;
    SOCKET Srv = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
```

接下来设置绑定的 IP 及端口

```
sockaddr_in sin;
sin.sin_family = AF_INET;
sin.sin_port = htons(49711);
sin.sin_addr.S_un.S_addr = INADDR_ANY;
```

绑定 IP 和端口，并进行监听

```
bind(Srv, (LPSOCKADDR)&sin, sizeof(sin)) == SOCKET_ERROR)
    //开始监听
listen(Srv, 2);
```

创建连接并收发数据

在此时，服务器端作为中转枢纽，首先等待来自客户端的连接请求，在成功接收客户端后，

创建线程以处理来自客户端消息的接收与转发，同时输出对应的系统消息以判断该流程是否成功进行。

```
//等待来自客户端的连接请求
    sockaddr_in ClientAddr;
    int len = sizeof(ClientAddr);
    while (1)
    {
        chat[num].client = accept(Srv, (SOCKADDR*)&ClientAddr, &len);//接受客户端
        chat[num].flag = num;
        recv(chat[num].client, chat[num].username, sizeof(chat[num].username), 0); //接收用户名
        HANDLE  hThread;
        //创建线程处理消息接收与转发
        hThread = CreateThread(NULL, NULL, (LPTHREAD_START_ROUTINE)AnswerThread, &chat[num], 0, &dwThreadId);
        if (hThread == NULL)
        {
            printf("线程号%d: 创建线程出错\n", GetCurrentThreadId());
        }
        else
        {
            printf("%s上线\n", chat[num].username);
            CloseHandle(hThread);
        }
        //标记加一
        num++;
    }
```

2. 客户端

(1) 时间输出函数

```
string getTime() {
    //获取系统时间
    time_t timep;
    time(&timep);
    char tmp[64];
    strftime(tmp, sizeof(tmp), "%Y-%m-%d %H:%M:%S", localtime(&timep));
    return tmp;
}
```

(2) 接收消息模块

```
DWORD WINAPI  ThreadRecv(LPVOID lparam){
    SOCKET Cli = (SOCKET)(LPVOID)lparam;
    char buf[128] = { 0 };
    int relen = 0;
    while (1){
        memset(buf, 0, sizeof(buf));
        relen = recv(Cli, buf, 128, 0);
        if (relen < 0){
            cout << "消息接收失败!" << endl;
            break;
        }
        else{
            cout << buf <<"【已接收】" << endl;
        }
    }
    return 0;
}
```

(3) 发送消息模块

搭建发送消息模块。当客户端发送消息时，消息首先将发给服务器端，并通过服务器 d 端的消息转发功能使得消息打印至其余客户端页面。另外，该模块会判断信息内容是否是 exit，若为 exit 则同时对该客户端执行退出操作。

```
DWORD WINAPI  ThreadSend(LPVOID lparam){
    SOCKET Cli = (SOCKET)(LPVOID)lparam;
    char buffer[128] = { 0 };
    char SendTime[32] = { NULL };
    int selen = 0;
    int lent = 0;
    while (1){
        //获取系统时间
        //获取系统时间
        time_t t = time(0);
        memset(buffer, 0, sizeof(buffer));
        strftime(SendTime, sizeof(SendTime), "%Y-%m-%d %H:%M:%S", localtime(&t));
        //添加一个无回显的读取字符函数读取回车
        int c = getch();
        printf("%s  %s:", SendTime, userName);
        gets_s(buffer);
        if (strcmp(buffer, "exit") == 0){
            lent = send(Cli, SendTime, 32, 0);
            selen = send(Cli, buffer, 128, 0);
```

```
            //##########对消息分类exit,over############
            cout << "程序将在3秒后退出" << endl;
            Sleep(3000);
            closesocket(Cli);
            WSACleanup();
            return 0;
        }
        lent = send(Cli, SendTime, 32, 0);
        selen = send(Cli, buffer, 128, 0);
        if (selen < 0 && lent < 0){
            cout << "消息发送失败！" << endl;
            break;
        }
    }
    return 0;
}
```

(4) 主函数

主函数部分与服务器端流程相近，首先向服务器端发送连接请求。

```
//初始化套接字
    WORD wVersionRequested = MAKEWORD(2, 2);     //版本号
    WSADATA wsaData;                             //socket详细信息
    WSAStartup(wVersionRequested, &wsaData);     //初始化Socket DLL,协商使用的Socket版本
    WSAStartup(wVersionRequested, &wsaData)
    //socket()
    SOCKET Cli = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    //设置服务器地址
    //设置要绑定的IP及端口
    sockaddr_in SrvAddr;
    SrvAddr.sin_family = AF_INET;
    SrvAddr.sin_port = htons(49711);
    SrvAddr.sin_addr.S_un.S_addr = inet_addr("127.0.0.1");
    int len = sizeof(SrvAddr);
    connect(Cli, (SOCKADDR*)&SrvAddr, len);
```

创建接收线程和发送线程

开启接收消息线程和发送消息线程。在线程进行时，会出现堵塞，因此应该在成功建立线程之后及时关闭线程，使得聊天内容可以继续进行。

```
//开启接受线程和发送线程
    HANDLE hThread1 = CreateThread(NULL, 0, ThreadRecv, (LPVOID)Cli, 0, NULL);
    HANDLE hThread2 = CreateThread(NULL, 0, ThreadSend, (LPVOID)Cli, 0, NULL);

    //进程关闭，主函数返回
    if (WaitForSingleObject(hThread1, INFINITE) == WAIT_OBJECT_0 ||
        WaitForSingleObject(hThread2, INFINITE) == WAIT_OBJECT_0) {
        CloseHandle(hThread1);
        CloseHandle(hThread2);
        return 0;
    }
```

（四）程序演示

首先开启服务器端和两个客户端并输入其用户名，创立聊天室；

此时服务器界面会展示聊天的全部信息，形成记录日志；

而客户端界面会展示彼此的信息交互，对于其余客户端发送的消息，会添加【已收到】标识。

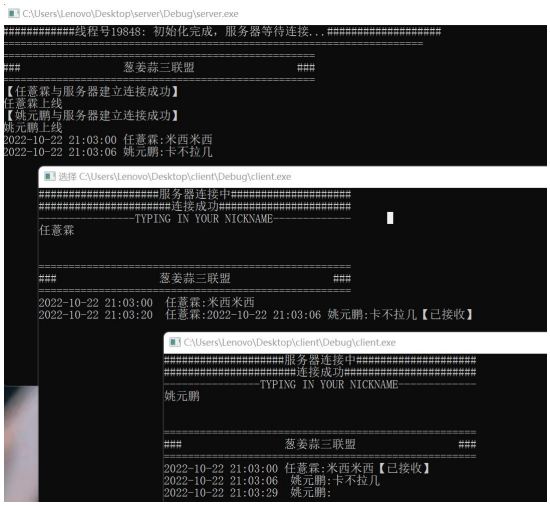

图 2 搭建聊天室

开启第三个客户端并加入到聊天室，实现群聊；

图 3 用户加入开启群聊

用户可以通过输入 exit 指令，退出聊天室，其余用户可以继续进行聊天；

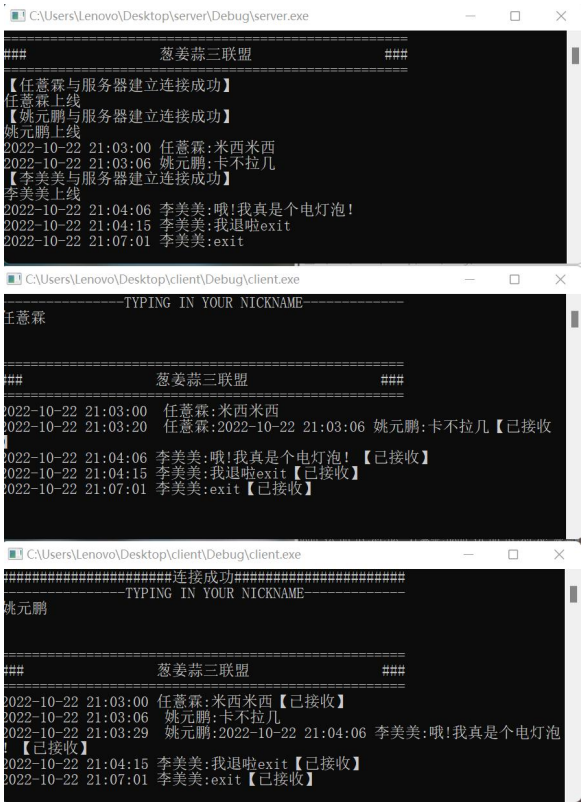

图 4 用户退出聊天继续

最终通过关闭服务器端结束完整进程。

四、实验分析

本次实验，初步了解了 SOCKET 编程，并初步实现了多人聊天功能。但距离真实聊天程序仍有一定差距。本程序的遗留问题如下：

1. 如何分辨客户端发送的信息是 exit 字符还是退出指令“exit”；
2. 如何实现多人聊天程序中的私聊功能；
3. 如何实现多人聊天程序中文件、图片等内容的发送。

因此，在接下来的学习中，应当以上述问题为起点继续学习有关 SOCKET 编程的相关知识，拟解决以上问题，实现一个更为全面的聊天程序。