**摘要：**在网络聊天盛行的今天，却没有合适的聊天室在Linux系统中供用户使用。本设计实现了Linux系统下简单的网络聊天室的设计。采用Client/Server模式，使用socket套接字和多线程的应用来进行编写。可以创建多个窗口分别作为服务端和多个客户端的程序。利用vim编辑器，通过gcc命令进行编译从而实现聊天室的运行，并使用gdb调试功能进行调试。本设计的主要功能包含群聊、私聊、查看用户列表等功能。

**关键词：**Linux；socket；C/S架构

目 录

第一章 引言 1

1.1 研究背景 1

1.2 开发环境 1

1.3 研究意义及内容 1

第二章 系统技术介绍 2

2.1 TCP和UDP通信的概念 2

2.11 UDP通信 2

2.1.2 TCP通信 2

2.2 客户端/服务器端架构 3

2.3 socket编程 4

2.4 多线程 5

第三章 系统总体设计 7

3.1 需求分析 7

3.2 系统功能模块设计 7

3.3 系统流程图设计 7

3.3.1 服务器端主要流程图设计 8

3.3.2 客户端主要流程图设计 9

第四章 系统关键模块实现 10

4.1 public.h 10

4.2 server.c 11

4.3 client.c 11

4.4 makefile 11

第五章 系统运行、调试及测试 12

第六章 结束语 17

参考文献 18

Abstract 19

# 第一章 引言

## 1.1 研究背景

在网络聊天盛行的今天，人与人之间的交流越来越方便，也更加注重时效性。QQ、MSN、微信等互联网通讯软件不一而足，使世界好像变得更小了，不管身处何地，只要连接上互联网就可以随时随地交流和沟通。但对于很多使用Linux操作系统的用户来说，现在使用的通讯软件都只适用于Windows操作系统，而并不支持Linux系统，这对Linux用户造成了极大的不便。

## 1.2 开发环境

**硬件：** 处理器：Intel(R)Core(TM)i5-7200U CPU @ 2.50GHz 2.71GHz

RAM：4.00GB

**软件：** VMware虚拟编辑器

Ubuntu操作系统

Vim 编辑器

GCC 编译器

GDB 调试器

## 1.3 研究意义及内容

目前IM(即时通讯)软件基本上都集中在MSwindows平台，很少考虑Linux用户的需求。因此如果能将网络聊天软件引入Linux操作系统，不仅弥补Linux系统下应用软件匮乏的现状，同时对Linux系统和网络聊天软件普及也有重要意义。

基于上述情况，将Linux与IM即时通信融合起来对Linux的发展和用户体验都有极大的意义。本次设计正是为了解决Linux下网络聊天问题，我们采用Ubuntu作为软件开发平台，以C语言为编程语言，采用C/S模式，利用vim编辑器以及gcc命令和gdb调试来设计一个简单的网络聊天系统。

# 第二章 系统技术介绍

## 2.1 TCP和UDP通信的概念

### 2.11 UDP通信

UDP：（用户数据报协议），英文全称“User Datagram Protocol”。UDP是一种无连接协议。UDP在传输前不需要先建立连接，远程主机的传输层在收到udp数据报后，不需要任何确认，因此不能保证数据传送的可靠性。

基本UDP客户端—服务器端程序设计基本框架：

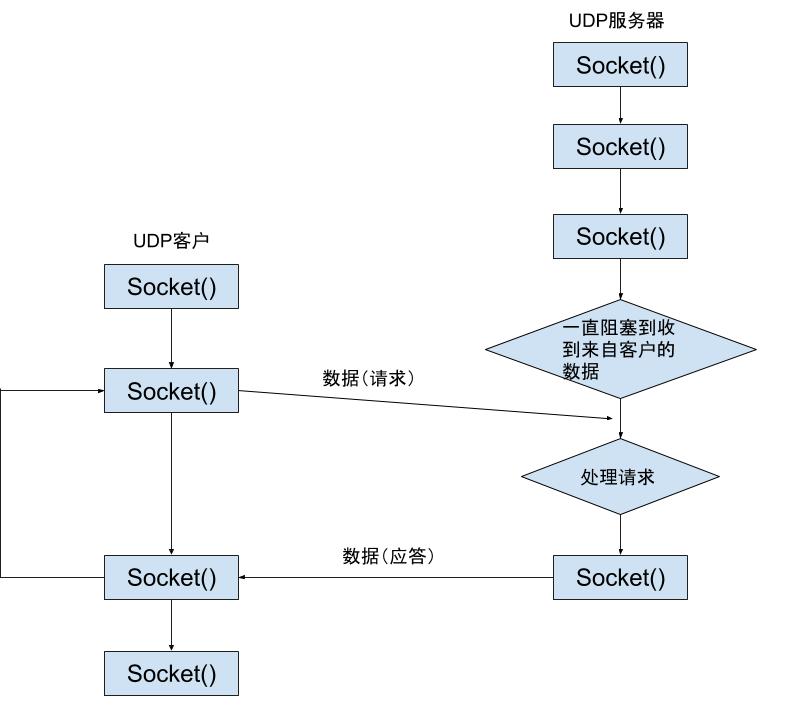


图2.1.1 UDP通信流程图

### 2.1.2 TCP通信

TCP：（传输控制协议），英文全称“Transmission Control Protocol”。TCP是一种面向连接的协议。TCP在传输前必须先建立连接，传输结束后要释放连接，其连接过程需要通过三次握手实现，因此TCP提供可靠的传输服务。

基本TCP客户—服务器程序设计基本框架：

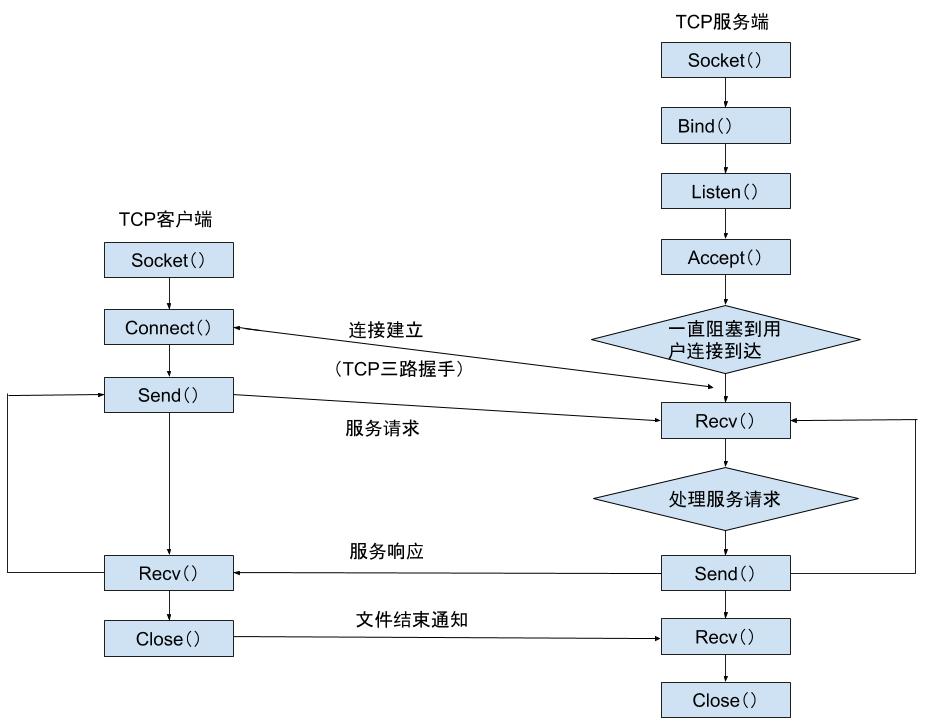


图2.1.2 TCP通信流程图

从图中可以明显的看出UDP相比于TCP而言没有三次握手的过程，这也是为什么UDP没有TCP可靠的原因。

## 2.2 客户端/服务器端架构

目前所熟知的软件系统体系结构有两种架构：C/S（客户/服务器）架构是一种比较早的软件架构，B/S（浏览器/服务器）架构是C/S架构的升级版，是一种全新的软件系统架构技术。

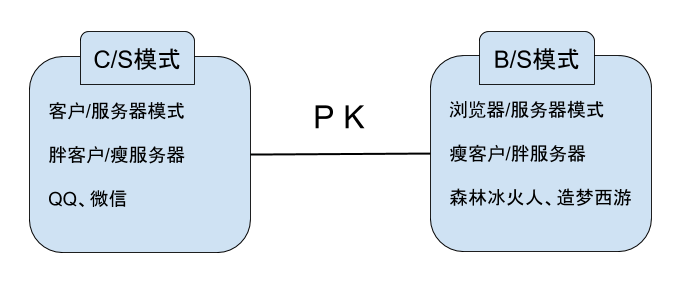


图2.2.1 C/S与B/S的对比

在客户/服务端的模型中，多个PC机都作为客户端，与服务器进行连接，并通过服务器进行信息传输。所以，采用这种架构来进行网络编程时，需要分别编写客户端和服务器端的程序。其中C/S对技术人员要求比较高，而B/S比较人性化，所以不需要掌握比较专业化的技能。

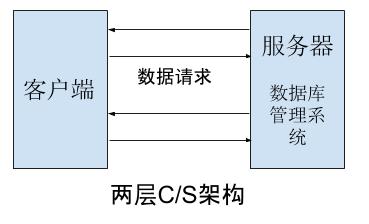


图2.2.2 C/S系统体系架构

**服务器的主要功能如下：**

1.在特定的端口上进行监听，等待客户端的连接。

2.用户可以配置服务器端的监听端口。

3.向连接的客户端发送登录成功信息。

4.向已经连接到服务器的客户端的用户发送系统消息。

5.使用TCP多线程并发服务器，向在线的所有客户端发送消息

**客户端的主要功能如下：**

1.客户端传参输入IP和端口号以及用户名进行注册

2.连接到已经开启的服务的服务端

3.用户可以向所有人发送信息

4.用户可以接受服务器发送的系统消息

## 2.3 socket编程

Socket是应用层与TCP/IP协议族通信的中间软件抽象层，它是一组接口。Socket接口上TCP/IP网络应用程序接口（API），它提供了许多函数和例程，程序员可以使用它们来开发TCP/IP网络应用程序。

Socket调用库函数主要有：

1.socket()--创建套接字

2.bind()--分配套接字地址

3.listen()--等待连接请求

4.accept()--允许连接请求

5.read()/write()--数据交换

6.close()--关闭连接

服务器的工作流程：

首先调用socket函数创建一个Socket，然后调用bind函数将其与本机地址以及一个本地端口号绑定，然后调用listen在相应的socket上监听，当accpet接收到一个连接服务请求时，将生成一个新的socket。服务器显示该客户机的IP地址，并通过新的socket向客户端发送字符串，最后关闭该socket。

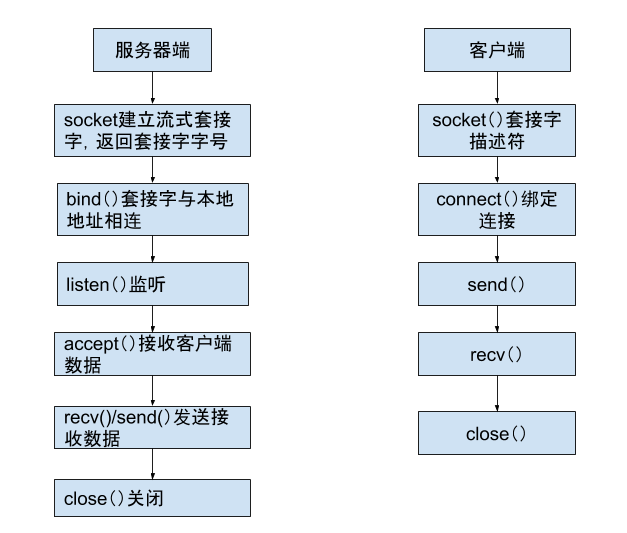


图2.3.1 socket通信流程图

## 2.4 多线程

上面介绍的是点对点通信完成了客户机与服务器之间的连接，建立连接的进程之间是一对一的联系，因为客户机进程可能会有多个，所以会存在多个任务通过一个端口地址接收和发送。对于这种多个并发的任务需要创建多个线程去实现，不会影响任务的执行。

1、什么是线程？

提到线程的概念，就会跟进程区分开，下面分享一下进程和线程简单的区别：

1）进程—资源分配的最小单位。

2）线程—程序执行的最小单位。

2、线程的创建

创建进程需要操作系统分配资源给它，Linux下C语言使用fork（）函数来创建。一个进程包含多个轻量级的线程，而线程用clone（）函数来创建。

3、线程的生命周期

线程的生命周期有四种状态：就绪、运行、阻塞和终止。见下表

表1.1 线程生命周期的状态

|  |  |
| --- | --- |
| 状态 | 含义 |
| 就绪 | 线程能够运行，但是在等待可用的处理器 |
| 运行 | 线程正在运行，在多核系统中，可能同时有多个线程在运行 |
| 阻塞 | 线程在等待处理器以外的其他条件 |
| 终止 | 线程从启动函数中返回，或者调用pthread\_exit函数，或者被取消 |

状态的转换发生在一定的条件下，下图显示了线程在一定条件下是如何转换状态的：

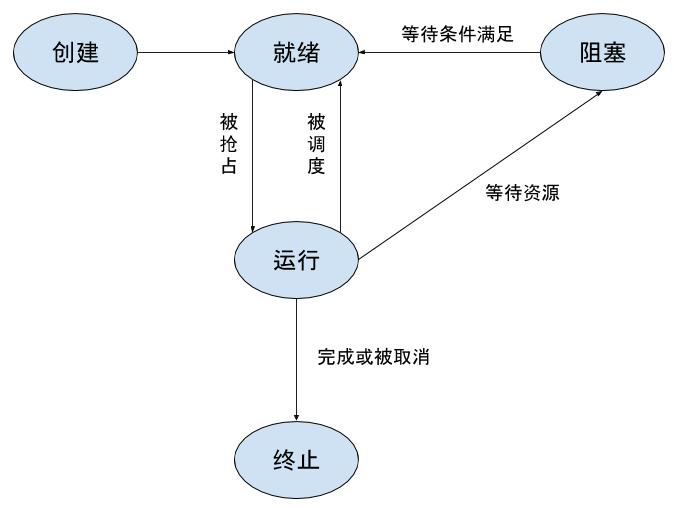


图2.4.1 线程的状态转换

4、多线程编程

在使用多线程函数编程时需要用以下命令进行编译：

$ gcc a.c -o a -lpthread

Linux在进行多线程编程时遵循POSIX线程接口规则，成为pthread 。编写程序用的头文件是pthread.h，因为pthread不是Linux系统的库，所以在编译时要加上 -lpthread。

# 第三章 系统总体设计

## 3.1 需求分析

Linux 下的网络聊天系统能够满足人们生活中日常交流的需要。对于用户来说，可以注册账号，登录系统，查看当前在线用户，以及与某一特定的用户进行私聊，或者与全部用户进行群聊，同时满足用户与用户之间传输的要求。

## 3.2 系统功能模块设计

用户首先需要注册来登录访问，登录后使用系统的聊天功能：例如查看在线好友列表，与其中一个用户私聊或者与大家在一起群聊。根据以上需求，基于Linux网络聊天的系统功能模块设计图如图3.2.1所示。

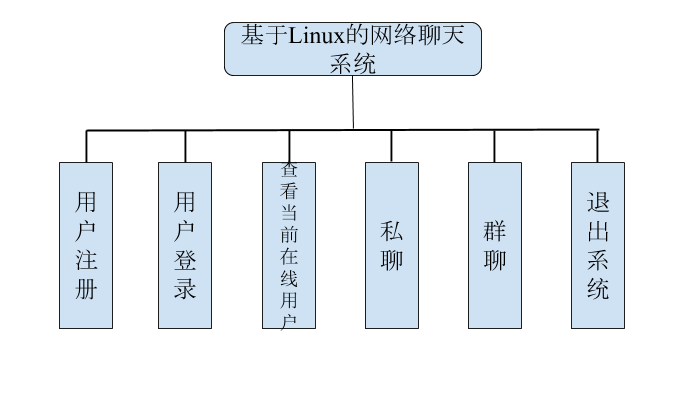


图3.2.1 功能模块设计

## 3.3 系统流程图设计

本系统是采用C/S架构设计的Linux网络聊天系统，也就是客户端/服务器端的结构。前面已经讲到，在C/S模式下多台PC机（client）可以跟服务器（server）相连，并通过服务器端传输信息。所以在进行系统设计时，需要分别为客户端和服务器端两部分编写程序。在实现系统功能的过程中，需要遵照一定的流程来完成系统功能的实现过程。

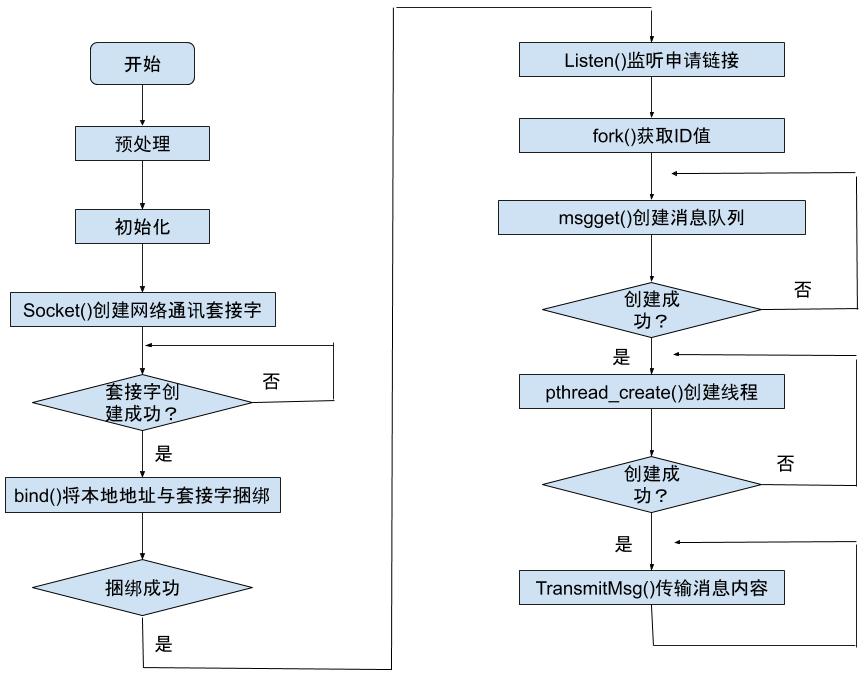


图3.3.1 服务器主程序流程图

### 3.3.1 服务器端主要流程图设计

服务器端是整个基于Linux 的网络聊天系统尤其重要的一部分，也是系统运行时最关键的部分。系统服务器程序部分主要包括服务端主程序、查看在线好友列表程序、用户聊天模式选择程序、用户发送消息处理程序，系统服务主程序最主要的功能是作为服务器程序主函数main()的入口，实现socket套接字的创建等操作。服务主程序流程图如图3.3.1 所示。

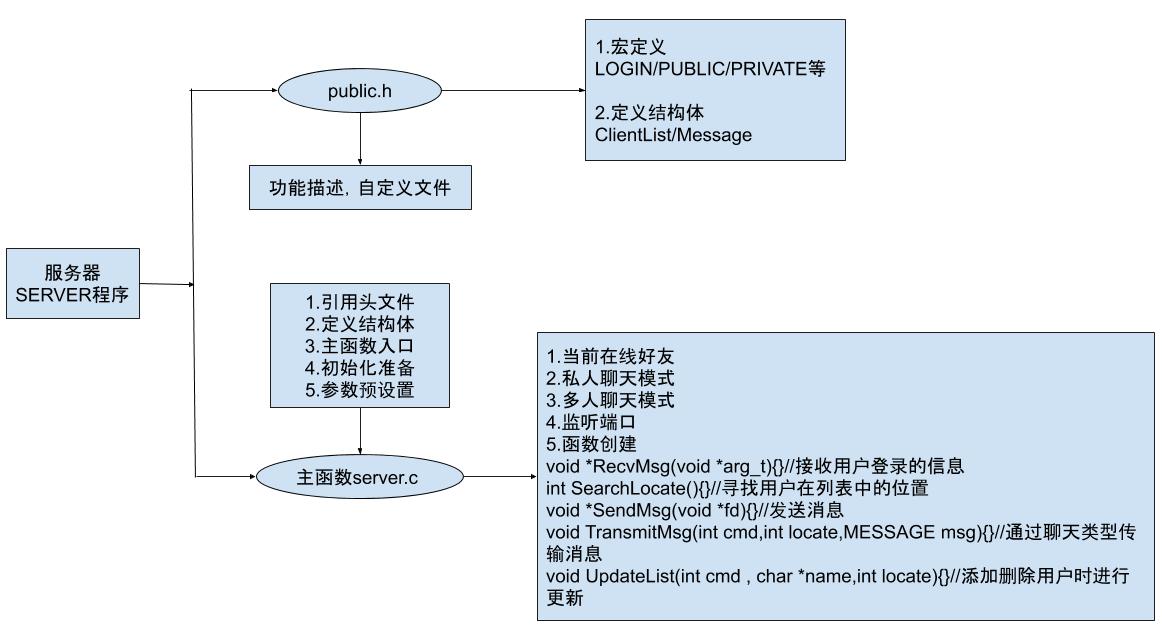


图3.3.1 服务器系统流程图

### 3.3.2 客户端主要流程图设计

由于服务器是一对多的工作模式，而客户端是一对一的工作模式。但是对于客户端来说只需完成客户端与服务器之间的一对一通讯即可。系统客户端程序部分主要包括客户端主程序、查看在线好友列表程序、接收消息程序、消息发送程序等部分。

1) 客户端主程序流程

客户端主程序作为主函数main ( ) 的入口，负责创建socket 网络通讯套接字、进行用户消息处理、连接服务器等操作。具体的客户端主程序流程是socket( ) 创建网络通讯套接字然后connect( ) 连接服务器连接成功后，进行客户端登录处理函数处理。

2) 客户端用户登录程序流程

在客户端开始运行时，系统进入模式选择界面。系统提供了注册并登录、选择公聊或私聊、查看用户列表、退出聊天室等四项功能供用户选择。如图3.3.2所示

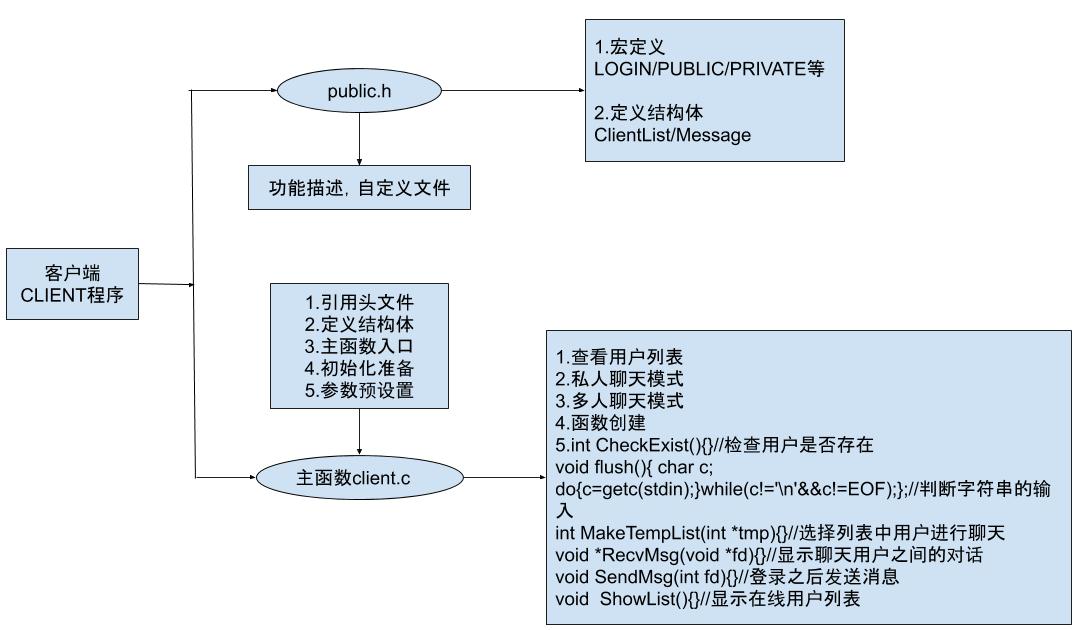


图3.3.2 客户端主要流程图

# 第四章 系统关键模块实现

本系统代码通过多文件系统，模块化实现，多文件系统有一定的优缺点，但对于大型系统，多文件系统可以提高代码的高内聚和低耦合。

模块化主要有以下几个优点：

1.便于复用代码。通用性强的重复的功能只要写一遍就可以了，下次要用在其它程序上时只要更改很小的部分或者可以不用更改。

2.便于多人协作。在设计软件之初就可以很清楚地分配各个开发部门的任务。模块的编写者本身只要关注他所写的东西，清楚这一部分的功能，留出接口就可以了。另外，对于整个工程的负责人而言，这样会方便浏览全局的工作进度，统筹人员安排。

3.便于修改和维护。如果能确定只是某个模块有问题，在模块内解决即可，不需要牵一发而动全身。要升级某一部分的功能，可以只针对具体的模块重新开发，节约成本。

其实不只是C语言，许多其它语言也经常使用这种方法。开发大型软件时这种方法非常有效（否则不明显，或者反而有副作用）。对于软件设计来说这不仅仅是一种风格，而是一种方法了。

include包含的声明函数的头文件，使用extern关键字可以直接调用，这样就实现了多文件和多模块了。

本系统总共包含两个部分，客户端和服务器端。下面通过两部分分开介绍。

## 4.1 public.h

在public.h中封装了两个结构体，分别如下：

//显示用户列表

typedef struct ClientList

{

char name[NAME\_LEN];

int socketFd;

}CLIENTLIST;

//定义用户名和消息的长度以及最大用户收发消息

typedef struct Message

{

char fromUser[NAME\_LEN];

int fromUserLocate;

int type;

int sendUserLocate;

char message[MSG\_LEN];

CLIENTLIST clientList[MAX\_CLIENT];

}MESSAGE;

## 4.2 server.c

在server.c中创建了五个函数分别是：

void \*RecvMsg(void \*arg\_t){}//接收用户登录的信息

int SearchLocate(){}//寻找用户在列表中的位置

void \*SendMsg(void \*fd){}//发送消息

void TransmitMsg(int cmd,int locate,MESSAGE msg){}//通过聊天类型传输消息

void UpdateList(int cmd , char \*name,int locate){}//添加删除用户时进行更新

还引用头文件public.h中封装的结构体和创建多线程来进行实现服务端的监听。

## 4.3 client.c

在client.c中同样引用了public.h中封装的结构体，并创建了六个函数分别是：

int CheckExist(){}//检查用户是否存在

void flush(){ char c; do{c=getc(stdin);}while(c!='\n'&&c!=EOF);};//判断字符串的输入

int MakeTempList(int \*tmp){}//选择列表中用户进行聊天

void \*RecvMsg(void \*fd){}//显示聊天用户之间的对话

void SendMsg(int fd){}//登录之后发送消息

void ShowList(){}//显示在线用户列表

另外在main函数中直接定义ip值为127.0.0.1，所以直接可以与本机互联，通过多线程服务来进行网络聊天。

## 4.4 makefile

Makefile文件的内容关系到了整个工程的编译规则。一个工程中的源文件不计其数，其按类型、功能、模块分别放在若干个目录中，makefile定义了一系列的规则来指定，哪些文件需要先编译，哪些文件需要后编译，哪些文件需要重新编译，甚至于进行更复杂的功能操作，所以makefile就像一个Shell脚本一样，其中也可以执行操作系统的命令。

下面是这个工程的makefile文件内容：

tar:server client

server:server.c

gcc -g -Wall -o $@ $< -lpthread

client:client.c

gcc -g -Wall -o $@ $< -lpthread

c:

rm -rf server client FIFO

因为在文件中使用了动态库pthread，因此在编译时要用包含指定的库，用“-l库名”来进行指定，动态库和静态库是有很大的区别的：静态库情况下，会把库直接加载到程序里，而在动态链接的时候，只是保留接口，将动态库与程序代码独立。这样就可以提高代码的可复用度和降低程序的耦合度。

# 第五章 系统运行、调试及测试

对本系统进行测试，需要运行一个服务器端，多个客户端进行测试，服务器端的运行结果如下图5.1所示

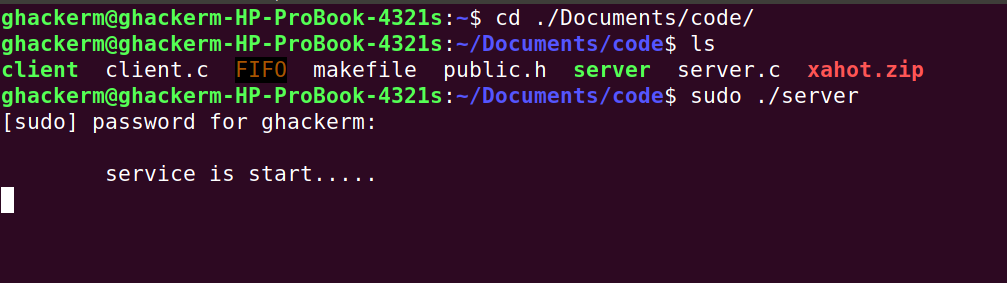


图5.1 服务器端运行结果

**步骤1：**首先编译server.c 和client.c文件，gcc命令是用来编译C/C++语言等，-o后面加的是输出的文件名，-lpthread前面有提到，见上文。命令如下：

$gcc server.c -o server -lpthread

$gcc client.c -o client -lpthread

其中编译的中间过程（预处理、编译、汇编和链接）可自行查看官方文档了解，这里不作说明。

**问题1**：一开始直接用普遍的编译命令$gcc file.c -o filename来编译，但是一直报错，拿着报错的内容上网搜索之后发现多线程网络编程需要加-lpthread。

**问题2：**生成的server可执行文件可以直接用./server运行，但是报错显示没有执行权限，于是加了sudo ./server执行服务器端开始监听。

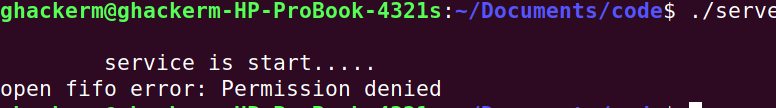


图5.2 服务器端报错显示

**步骤2：**然后服务器开启之后建立多个窗口来依次注册“宫本武藏”“娜可露露”“不知火舞”三个用户，选择1 public选项分别发送一条群聊消息。如图5.3所示

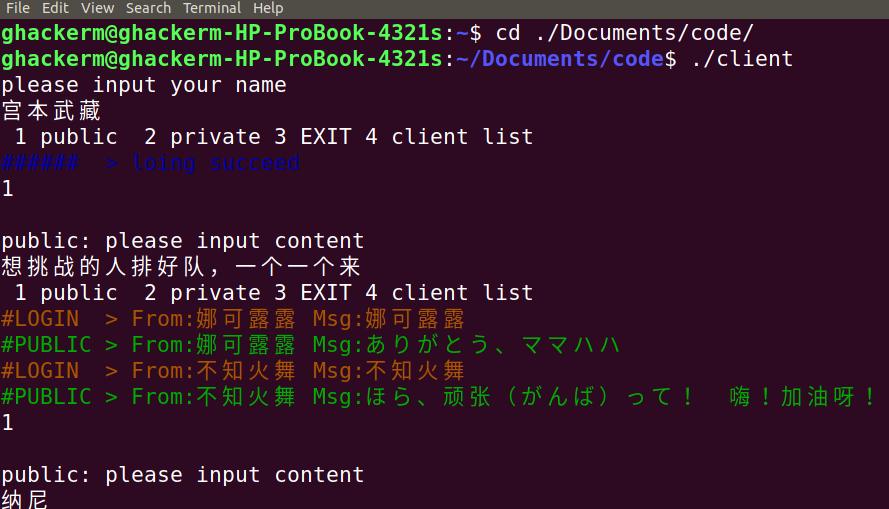


图5.3 创建用户

**步骤3：**接着选择2 private测试私聊发送消息的功能，显示在线用户列表后选择用户进行聊天。

图5.4显示的是宫本武藏与不知火舞的聊天界面

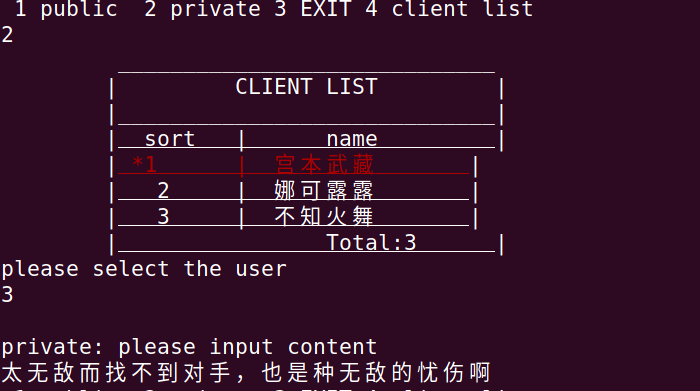


图5.4 宫本武藏与不知火舞的聊天界面

图5.5显示的是宫本武藏与娜可露露的聊天界面



图5.5 宫本武藏与娜可露露的聊天界面

**步骤5：**进入不知火舞和娜可露露的聊天界面，测试用户接收消息的功能

图5.6显示的是不知火舞的聊天界面，其中收到了宫本武藏的私聊消息及所有用户的群聊消息。



图5.6 不知火舞的聊天界面

图5.7显示的是娜可露露的聊天界面，其中收到了宫本武藏的私聊消息及所有用户的群聊消息。

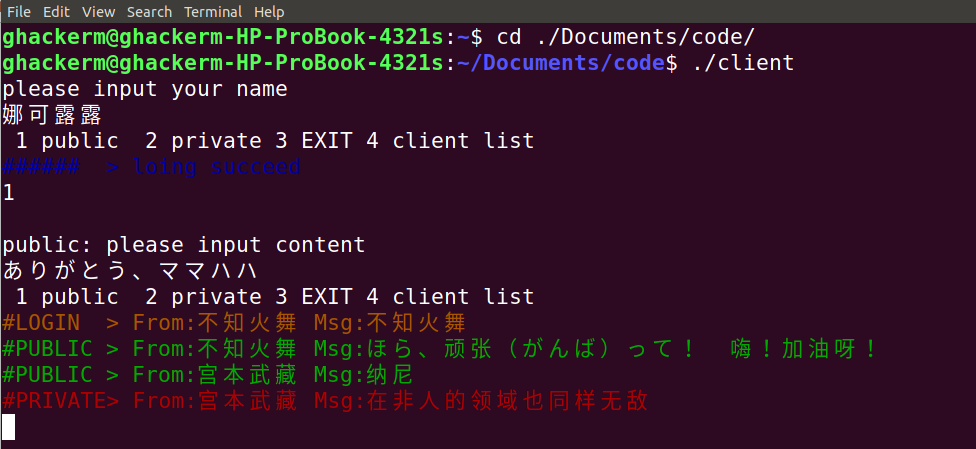


图5.7 娜可露露的聊天界面

**问题3：**在图5.5中发的消息在输入时会出现未删掉的情况，但在发送出去后消息中并不会显示未删掉的内容，这是一个bug，可能会造成用户体验感不佳。目前看来是Linux终端的问题，并不影响消息的完整性。

**步骤6：**以上的聊天室操作都在server中进行监听，下面显示的是所有客户端操作的内容



图5.8 服务器的监听界面

**步骤7：**最后选择3 exit退出系统，服务器监听到用户退出之后删除该用户，CTRL+C结束程序。

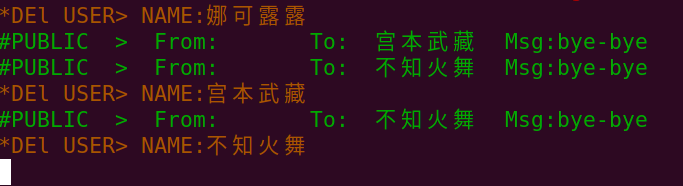


图5.9 用户退出并删除用户

# 第六章 结束语

本设计是针对Linux网络聊天室的设计，用户的注册和登录；相比于其他的聊天室完善了查询在线用户列表的功能；使用socket套接字,TCP/IP协议和多线程技术实现聊天室的基本功能；所有的显示通过终端的字符界面呈现出来。

在编写的过程中，客户端和服务器端的交互是关键，通过相互收发消息进行确认，相当于自定义一个协议来判断客户端和服务器端的连接是否正常；代码的错误处理程序是灵魂，这对于代码的调试是不可或缺的。

这款软件目前所具备的功能已经能够满足日常的聊天需求。服务器经过测试后，能够运行。客户端的字符界面效果良好，使用起来也简单易用。欠缺之处在于：自己写程序交互性比较差，很多修改需要重新修改代码，重新编译才能进行运行。这主要是自己做的项目还是比较少，需要在以后多多提高。

通过本次课程设计，我的软件开发能力在一定程度上提高了，对LINUX网络编程有了比较多的了解。实验过程中遇到了很多问题，刚开始对于shell一些简单的编程都不是很熟悉，通过去图书馆查阅资料，询问老师和同学，上网查阅资料，才得以解决各个问题。但是由于自己能力的有限，没能做出一个窗体，让系统更完美化。一开始想的是用QT实现的真正带窗口的聊天室，但是项目有点大，越研究内容越多，最终没能实现，这还需要以后的继续努力。

# 参考文献

[1]. 赵敏哲. 64位Linux操作系统与应用实例[M]. 北京：机械工业出版社，2001

[2]. 庞丽萍. 操作系统原理(第三版)[M]. 武汉：华中科技大学出版社，2000

[3]. 张饶学. Linux网络编程[M]. 北京:清华大学出版社，2000

[4]. 李善萍, 陈文智. 边学边干----Linux内核指导. 杭州:浙江大学出版社，2002

[5]. 高传善. 数据通信与计算机网络[M ]. 高等教育出版社，2000.

[6]. 胡明庆, 高巍, 钟梅. 操作系统教程与实验. 北京:清华大学出版社，2007

[7]. 白静慧. Linux下基于 Socket的网络通信. 有线电视技术[J].，2008(1).

[8]. Warren W. 詹俊鹄译.实战Linux Socket编程[M] .西安电子科技大学出版社，2001

[9]. 陈学峰,柴俊霞. Linux下基于Socket的网络编程. 淮北煤炭师范学院出版社，2004

[10]. 周炎涛,李立明. TCP/ IP 协议下网络编程技术的实现[J].航空计算技术，2002

[11]. 李翔，赵东升. Linux环境下的网络编程[J].电子与电脑. ，1999

[12]. 杜金祥, 杜宇轩. 基于Linux的局域网聊天系统设计[J]. 山西电子技术, 2014(1):67-69.

[13]. Warren W. Gay.Linux Socket Programming by Example[M].United States of America.2000.4：159-160.

**Abstract**

**Design of Linux-based Network Chat Room**

**GAO Ming**

**Abstract:** Today, when web chat is prevalent, there is no suitable chat room for users to use in the Linux system. This design implements the design of a simple online chat room under the Linux system. In client/server mode, it is written using socket sockets and multi-threaded applications. You can create multiple windows as programs for the server and multiple clients. Use the VIM editor to compile with the GCC command to run the chat room and debug using the GDB debugging function. The main features of this design include group chat, private chat, view user list and other functions.

**Key words:** Linux; socket; C/S architecture