****

**网络程序设计课程设计**

**第X组**

**网工180Y班**

**2020-2021-1学期**

|  |  |  |
| --- | --- | --- |
| **课程设计基本信息** | | |
| **题目** | 群聊天软件的设计与实现 | |
| **时间** | 2020年12月28日至2020年12月31日 | |
| **地点** | 2-16-504 | |
| **指导教师** | 梁荣 | |
| **组织方式** | 多人组队，合作完成 | |
| **小组成员信息及成绩** | | |
| **姓名** | **学号** | **成绩** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| **指导教师评语** | | |
|  | | |

**目 录**

**[1.](#_Toc59959669)****[选题分析及设计目标](#_Toc59959669)** [1](#_Toc59959669)

**[2.](#_Toc59959670)****[概要设计](#_Toc59959670)** [2](#_Toc59959670)

**[2.1 模块划分及功能描述](#_Toc59959671)** [2](#_Toc59959671)

**[2.2 模块间交互关系](#_Toc59959672)** [2](#_Toc59959672)

**[3.](#_Toc59959673)****[详细设计](#_Toc59959673)** [3](#_Toc59959673)

**[3.1 函数设计](#_Toc59959674)** [3](#_Toc59959674)

**[3.2 数据结构设计](#_Toc59959675)** [3](#_Toc59959675)

**[3.3 函数调用关系](#_Toc59959676)** [3](#_Toc59959676)

**[4.](#_Toc59959677)****[实现代码](#_Toc59959677)** [4](#_Toc59959677)

**[5.](#_Toc59959678)****[测试报告](#_Toc59959678)** [5](#_Toc59959678)

**[6.](#_Toc59959679)****[遇到的问题及解决方法小结](#_Toc59959679)** [6](#_Toc59959679)

**[7.](#_Toc59959680)****[参考文献](#_Toc59959680)** [7](#_Toc59959680)

**[8.](#_Toc59959681)****[个人小结](#_Toc59959681)** [8](#_Toc59959681)

1. **选题分析及设计目标**

本系统实现一个基于Linux的模拟即时通信系统，要求实现以下功能：  
1、模拟即时通信系统可以实现多人同时在线聊天功能；  
2、在线聊天用户登录本系统需输入用户名和密码；  
3、本系统需能够查询历史聊天记录；  
4、本系统运行后，需启动一个守护进程，该守护进程记录本系统启动和关闭的时间，每个用户登录和退出时间；

该聊天工具是在Linux平台下进行开发与实现的，可以实现多人通信和交流。系统采用了TCP/IP协议和socket接口。从总体来讲，该聊天系统主要包括服务端与客户端。  
服务端功能实现模块：  
 用户通过用户名和密码进行登录  
 接收和发送用户聊天信息功能（接受发信用户发来的信息，再将信息转发到收信用户）  
客户端功能实现模块：  
 验证登录用户的信息  
 客户端之间进行信息传递交流  
 保存聊天记录

在本系统中主要分为两个模块分别是客户端程序和服务器端程序，如下图1所示描述该系统的功能模块图。图2描述整个系统的通信过程。

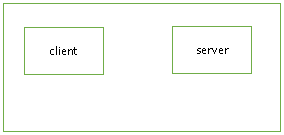


图1 功能图

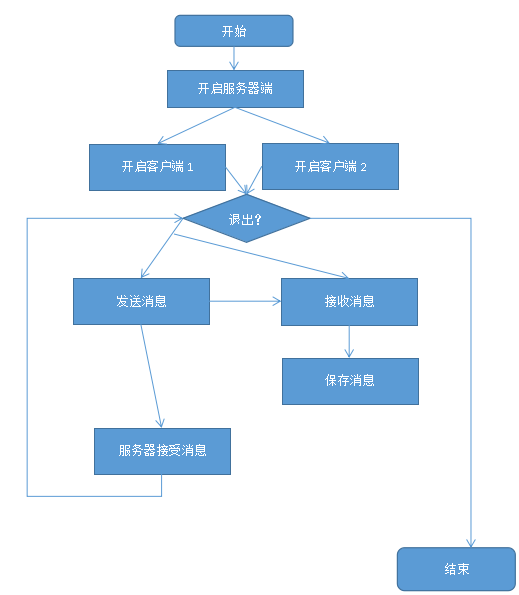


图2 流程图

1. **概要设计**

**2.1 模块划分及功能描述**

（1） 服务器端：  
服务器端登录时与用户名和密码相关联，成功登陆后，可以开启多个客户端，服务器端通过send（）函数来向客户端发送服务器端开启的消息，通过accept（）接收了客户端连接的请求，连接成功，实现了多人在线聊天的功能。发送消息时，将存储的消息以char \*的形式通过函数给socket通信。接收消息时，当有消息传来，通过recv（）函数先检查套接字sock的接收缓冲区，如果sock的接收缓冲区中没有数据或者协议正在接收数据，那么recv就一起等待，直到把数据接收完毕。当协议把数据接收完毕，recv函数就把s的接收缓冲区中的数据copy到rbuf中，可以显示在聊天界面的文本框内。

（2） 客户端：  
多个客户端在和服务器端连接成功后，开始进行多人在线聊天，通过checklogin（）函数实现用户名和密码匹配，从而实现用户登录，验证完之后开始进行多人聊天，通过write（）函数将聊天记录保存log.txt.同样用户发送消息时通过send（）函数向服务器端发送消息，通过recv（）函数来接收消息内容。

**2.2 模块间交互关系**

首先是开启服务器端，然后再开启多个客户端，客户端之间相互发送消息，需要通过服务器端接收再次发送给另一方客户端。当客户端想结束消息的发送时，需要输入字符串bye来结束客户端的对话，同时客户之间的对话会保存到相应文件，当客户需要查询时可以输入字符串find由服务器端打印出消息记录。在结束客户端对话后，可以直接关闭服务器端。

在整个通话中，如果不开启服务器端是不能进行正常对话，服务器端在整个过程中起着核心的作用。

1. **详细设计**

**3.1 函数设计**

client.c:

void save(char name[],char s[])实现聊天记录的信息保存操作；

main(int argc,char \*argv[])：核心功能函数，实现客户与服务器之间的通信。

server.c:

int main(int argc,char \*argv[])：函数实现服务器与客户通信，并能显示查看消息记录。

**3.2 数据结构设计**

服务器端：

struct client{

int s;

char name[50];

int online;

};

static struct client cli[MAX\_CLIENT];

该结构体用于保存客户的上线信息，cli为结构体数组保存客户信息。

static struct sockaddr\_in server,client;：表示服务器端需要通信的客户机地址。

客户端：

static struct sockaddr\_in server;实现服务器端通信地址；

static char rbuf[1024];表示接收或发送的消息。

**3.3 函数调用关系**

客户端：

main函数调用其内实现的save函数，保存消息记录。并且在该函数内调用用于socket通信的接口实现通信编程，如：inet\_addr，用于接收服务器端口地址信息，建立连接；htons接收指定的服务端口。

服务器端：

调用socket接口实现通信，其中bind函数用于信息地址的绑定，建立信息的连接；listen函数用于监听客户端的状态，便于信息的连接。

1. **实现代码**

#include <stdio.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <string.h>

#include <unistd.h>

#include <fcntl.h>

#include <stdlib.h>

#include<time.h>

#define PORT 54188

#define MAX\_CLIENT 10

// 服务器端代码：

struct client{

int s;

char name[50];

int online;

};

static int sock;

static char rbuf[1024];

static char wbuf[1024];

static struct sockaddr\_in server,client;

static struct client cli[MAX\_CLIENT];

int main(int argc,char \*argv[]){

int max\_sock;

int len;

sock=socket(AF\_INET,SOCK\_STREAM,0);

if(sock==-1){

perror("socket");

return 1;

}

max\_sock=sock;

bzero(&server,sizeof(server));

server.sin\_port=htons(PORT);

server.sin\_addr.s\_addr=INADDR\_ANY;

server.sin\_family=AF\_INET;

if(bind(sock,(struct sockaddr \*)&server,sizeof(struct sockaddr))==-1){

perror("bind");

return 2;

}

listen(sock,MAX\_CLIENT);

int i=0,j=0;

fd\_set rfd;

while(1){

FD\_ZERO(&rfd);

FD\_SET(sock,&rfd);

for(i=0;i<MAX\_CLIENT;i++){

if(cli[i].s==0)

continue;

FD\_SET(cli[i].s,&rfd);

}

switch(select(max\_sock+1,&rfd,NULL,NULL,NULL)){

case 0:

continue;

case -1:

continue;

default:

if(FD\_ISSET(sock,&rfd)){

for(i=0;i<MAX\_CLIENT;i++){

if(cli[i].s==0){

len=sizeof(struct sockaddr);

cli[i].s=accept(sock,(struct sockaddr \*)&client,&len);

if(cli[i].s==-1){

perror("accept");

cli[i].s=0;

break;

}

cli[i].online=0;

if(cli[i].s>max\_sock)

max\_sock=cli[i].s;

printf("accept!\n");

break;

}

}

}

for(i=0;i<MAX\_CLIENT;i++){

if(cli[i].s==0)

continue;

if(FD\_ISSET(cli[i].s,&rfd)){

int rsize;

bzero(rbuf,sizeof(rbuf));

rsize=recv(cli[i].s,rbuf,sizeof(rbuf),0);

if(rsize>0){

printf("%s\n",rbuf);

if(cli[i].online){

for(j=0;j<MAX\_CLIENT;j++){

if(cli[j].s==0 || j==i || cli[j].online==0)

continue;

bzero(wbuf,sizeof(wbuf));

strcpy(wbuf,cli[i].name);

strcat(wbuf,rbuf);

send(cli[j].s,wbuf,strlen(wbuf)+rsize,0);

}

if(strcmp(rbuf,"find\n")==0){

char buff[1024];

FILE \*fp=open("log.txt","r");

while(feof(fp)==0){

fscanf(fp,"%s",buff);

printf("%s\n",buff);

}

if(!strcmp(rbuf,"bye\n")){

send(cli[i].s,"quit",4,0);

close(cli[i].s);

cli[i].s=0;

cli[i].online=0;

memset(cli[i].name,0,sizeof(cli[i].name));

}

}

else{

if(!strncmp(rbuf,"#name",5)){

strcpy(cli[i].name,rbuf+5);

cli[i].online=1;

}

}

}

}

}

}

}

return 0;

}

//client.c：

#include <unistd.h>

#include <fcntl.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include<time.h>

#define PORT 54188

static struct sockaddr\_in server;

static char rbuf[1024];

static int sock;

void save(char name[],char s[])

{

time\_t t;

char buf[1024];

FILE \*fp;

int i;

int n;

fp=fopen("log.txt","a+");

if(!fp)

{

printf("file open fail!");

exit(0);

}

time(&t);

ctime\_r(&t,buf);

fprintf(fp,"%s",buf);

fprintf(fp,"%s",name);

fprintf(fp,"%s",s);

fclose(fp);

printf("data save successfull\n");

}

int main(int argc,char \*argv[]){

FILE \*fp;

char logname[100];

char pwd[100];

int temp;

printf("please input username:");

scanf("%s",logname);

printf("please input password:");

scanf("%s",pwd);

// if(checklogin(logname, pwd, temp) == 0)

// return 0;

int i,j,err;

char name[100];

int rsize,wsize;

sock=socket(AF\_INET,SOCK\_STREAM,0);

if(sock==-1){

perror("socket");

return 1;

}

bzero(&server,sizeof(server));

server.sin\_family=AF\_INET;

server.sin\_addr.s\_addr=inet\_addr("127.0.0.1");

server.sin\_port=htons(54188);

err=connect(sock,(struct sockaddr \*)&server,sizeof(struct sockaddr));

if(err==-1){

perror("connect");

return 2;

}

printf("connect success!\n");

bzero(rbuf,sizeof(rbuf));

bzero(name,sizeof(name));

rsize=read(STDIN\_FILENO,rbuf,sizeof(rbuf));

rbuf[strlen(rbuf)-1]='\0';

sprintf(name,"#name%s:",rbuf);

send(sock,name,sizeof(name),0);

j=0;i=0;

fd\_set rfd;

FD\_ZERO(&rfd);

FD\_SET(STDIN\_FILENO,&rfd);

FD\_SET(sock,&rfd);

while(1){

FD\_ZERO(&rfd);

FD\_SET(STDIN\_FILENO,&rfd);

FD\_SET(sock,&rfd);

switch(select(sock+1,&rfd,NULL,NULL,NULL)){

case -1:

case 0:

continue;

default:

if(FD\_ISSET(STDIN\_FILENO,&rfd)){

bzero(rbuf,sizeof(rbuf));

rsize=read(STDIN\_FILENO,rbuf,sizeof(rbuf));

if(rsize>0){

send(sock,rbuf,rsize,0);

save(name,rbuf);

}

}

if(FD\_ISSET(sock,&rfd)){

bzero(rbuf,sizeof(rbuf));

rsize=recv(sock,rbuf,sizeof(rbuf),0);

if(rsize>0){

printf("\*\*\*\*%s\n",rbuf);

if(strcmp(rbuf,"quit")==0){

close(sock);

exit(0);

}

}

}

}

}

return 0;

}

1. **测试报告**

程序测试通过执行程序的结果进行显示，如下图3所示，描述程序编译操作，打开两个终端分别编译两个程序文件。



图3 程序编译

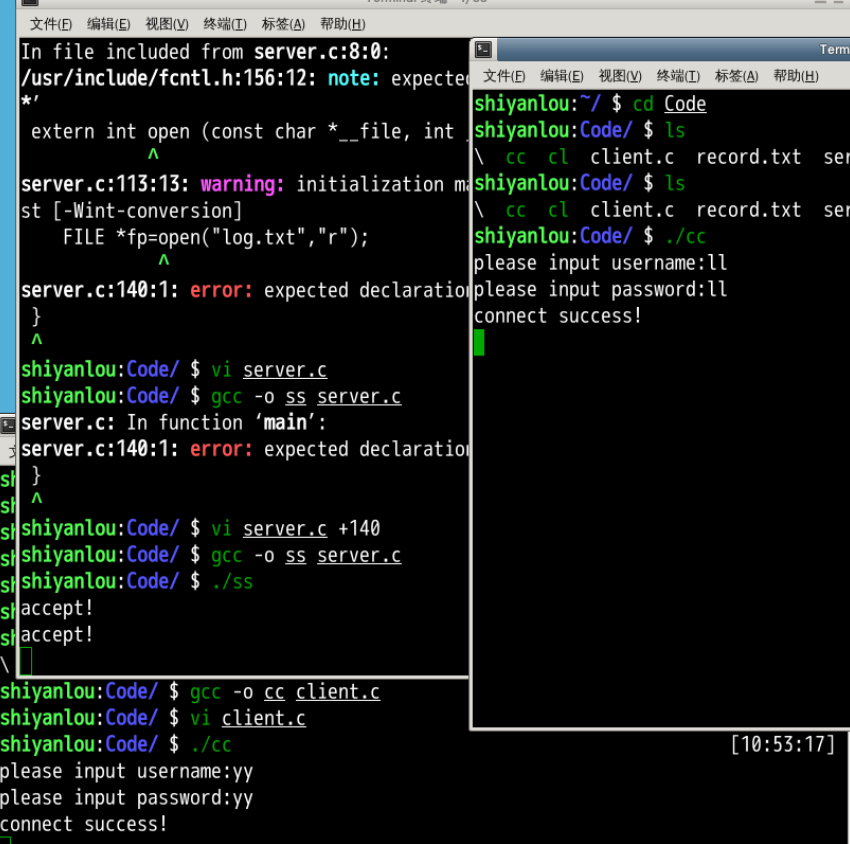
打开第三个终端，表示客户端，建立通信连接，如下图4所示。

图4 通信建立

消息传递，当输入bye结束操作，当输入find在会显示log.txt保存的历史记录，如下图5所示。

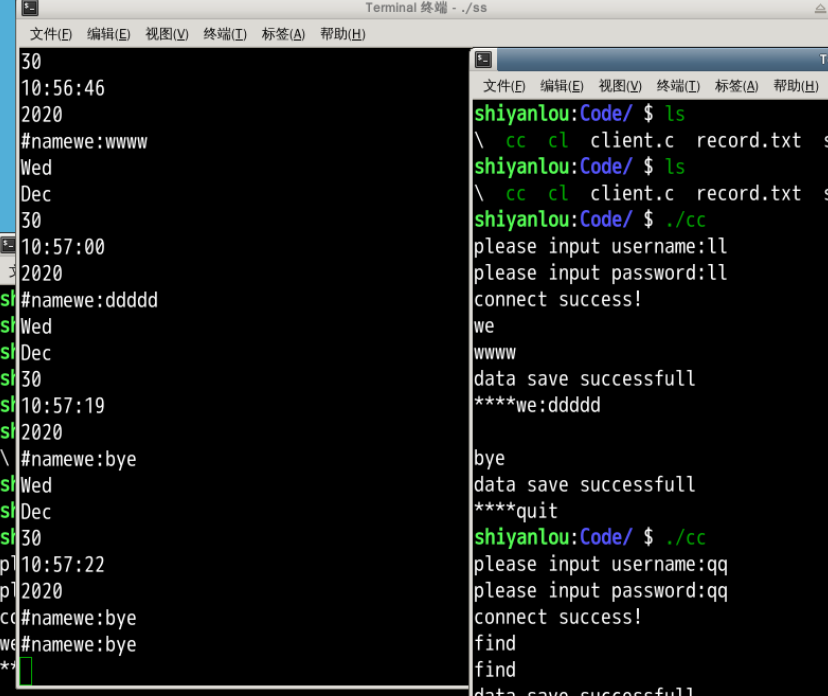


图5 通信操作

1. **遇到的问题及解决方法**

遇到问题：

1. 没有成功建立通信；
2. 遍历读取文件时失败，不能获取信息。

解决方法：

1）在客户端输入与服务器端一致的端口号，修改客户机的ip地址为127.0.0.1；

2）使用while循环，在循环内检测文件指针是否指向末尾，即：while(feof(fp)==0)。

1. **参考文献**
2. 翟高粤.基于Java语言Socket C/S通信的实现方法[J].甘肃科技纵横,2005,34(6):26-27,200. DOI:10.3969/j.issn.1672-6375.2005.06.020.
3. 王魁祎,陈东方.C语言编程技巧在C语言学习中的应用[J].信息与电脑,2020,32(12):68-70.
4. 庄淼.计算机C++语言编辑程序中常见问题及编程技巧[J].信息技术与信息化,2019,(10):87-89. DOI:10.3969/j.issn.1672-9528.2019.10.024.
5. 张雪坤.基于Socket的网络聊天程序设计与实现[J].电脑编程技巧与维护,2018,(4):16-17,24. DOI:10.3969/j.issn.1006-4052.2018.04.006.
6. 韩利凯.利用SOCKET进行网络通信程序设计[J].西安联合大学学报,2002,5(4):80-83. DOI:10.3969/j.issn.1008-777X.2002.04.021.
7. **个人小结**

无论你编程学得多好，上机实现才是目的，所以，不怕要辛苦，把你的每一段代码都敲进计算机，让计算机来实现，这样有助于你对程序的理解，并试着修改你的代码，让你的代码更精简，效率更高。平时没事的时候，在计算机上多敲代码，一个编程厉害的高手，他的计算机上会有多代码。

想成为编程高手的军规： 大学生活丰富多彩，会令你一生都难忘，但难忘有很多种，你可以学了很多东西而难忘，也会因为什么都没学到而难忘；编程不是技术活，而是体力活；C语言是基础，很重要，如果你不学好C语言，那么什么高级语言你都学不好。