**Linux操作系统与程序设计**

**课程设计报告书**

**目录**

1、实验项目目的 3

2、实验项目的功能及模块划分 14

3、实验项目的人员组成及指责划分 5

4、设计与实现 6

4．1 系统结构 6

4．2 客户端界面 6

4．2．1 界面设计 8

4．3服务器的实现 13

4．3．1 用户登录和注册功能的实现 13

4．3．2 聊天等消息传输功能 15

4．4 数据库访问或文件操作 17

4．5 通信模块的实现 17

5、测试与调试 18

6、总结 21

附录：程序代码 10

1、实验项目目的

（1）深入理解网络编程： 通过实现一个网上聊天程序，学生将深入理解TCP/IP协议栈、套接字编程等网络编程的基本概念。  
（2）掌握多线程编程： 为了支持多人同时在线聊天，需要掌握并发编程、多线程的使用，以提高服务器的处理能力。  
（3）实践文件传输： 实现图片和文件的传输将涉及到文件IO操作，为程序提供实际的应用场景。  
（4）用户认证与数据存储：将学会设计用户认证系统，了解用户信息的存储和管理，以及对数据库和文件的操作。  
（5）考虑安全性： 强调在设计中考虑安全性，尤其是在用户认证和数据传输阶段，以保障通信的安全性。  
（6）用户界面设计： 通过文本界面的设计，将体验到简单的用户交互设计，考虑如何提高用户体验。

（7）巩固课堂上学到的Linux平台上的编程规范、技术和技巧

（8）培养学生的团队协作能力

# 2、实验项目的功能及模块划分

本项目是一个文本界面下的多人网上聊天程序，实现的功能如下：  
（1）注册与登录

（2）查询用户信息

（3）创建聊天室

（4）加入聊天室

（5）查询所有的聊天室

（6）查询聊天室内用户

（7）私聊功能

（8）文件传输功能

（9）图片传输功能

（10）离线消息发送

模块划分：

1. 客户端模块：主要负责用户消息的接受和发送，主要处理逻辑放在服务端进行。
2. 服务端模块：主要服务汇集在服务端进行，支持并发连接多个用户同时进行交互，不断监听每个用户的输入，然后执行相应的程序进行处理。
3. 数据库模块：使用sqlite3数据库主要是保存用户的信息和离线消息，在进行验证登录和离线消息的收发时进行交互。
4. 文件交互模块：使用的是c语言中文件的相关读写操作，用于文件、图片等的发送和聊天室的交互之间。

# 实验项目的人员组成及指责划分

# 

# 4、**设计与实现**

## 4．1 系统结构

## 未命名文件 (7)

图4.1系统总功能展示

## 4．2 客户端界面

**4．2．1 界面设计**

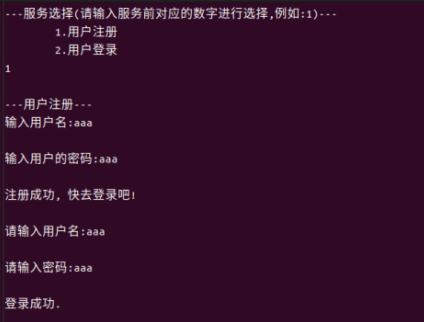


图4.2.1注册登录界面

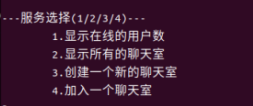


图4.2.2功能选择界面

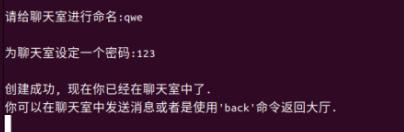


图4.2.3群聊界面

**4．3服务器的实现**

**4．3．1 用户登录和注册功能的实现**

在用户注册功能中，首先用户输入用户名和密码，在没有和数据库中信息重复的情况下，创建用户，并将信息存储到数据库中进行保存。在登录功能中用户首先输入用户名和密码，然后在服务端进行验证，如果正确则进入大厅执行后面的逻辑。

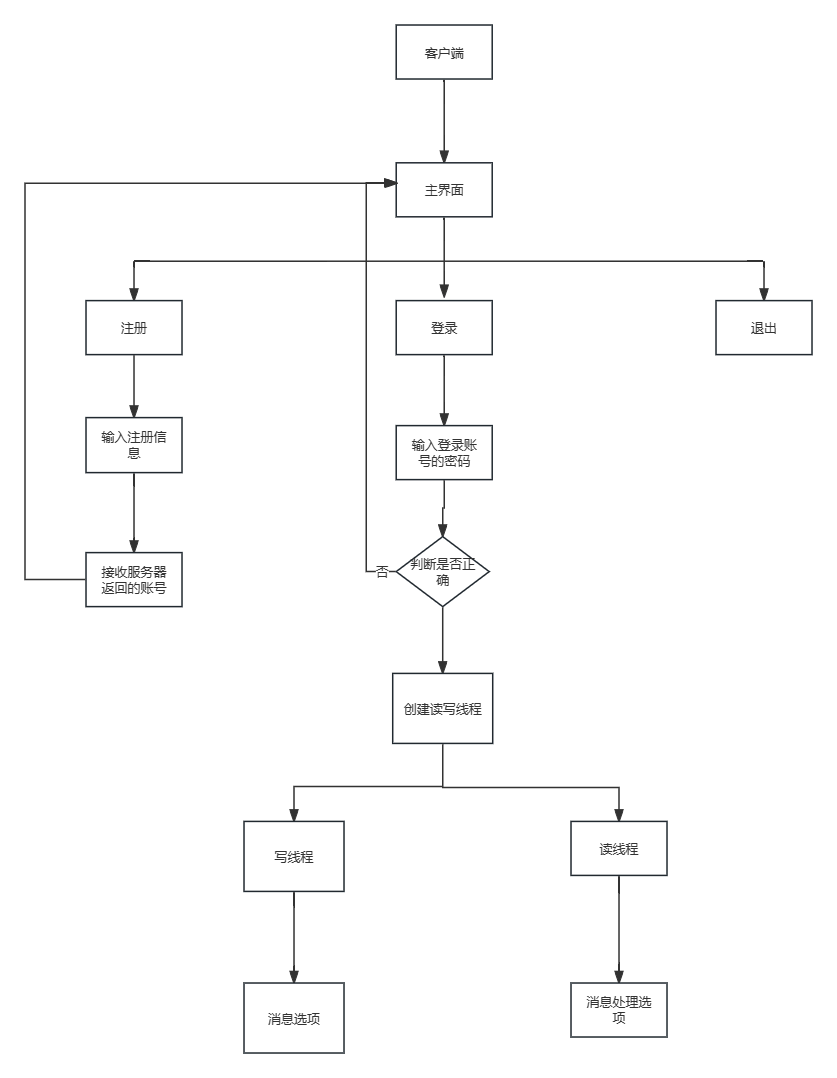
****

图4.3.1客户端流程图

**服务器端：**

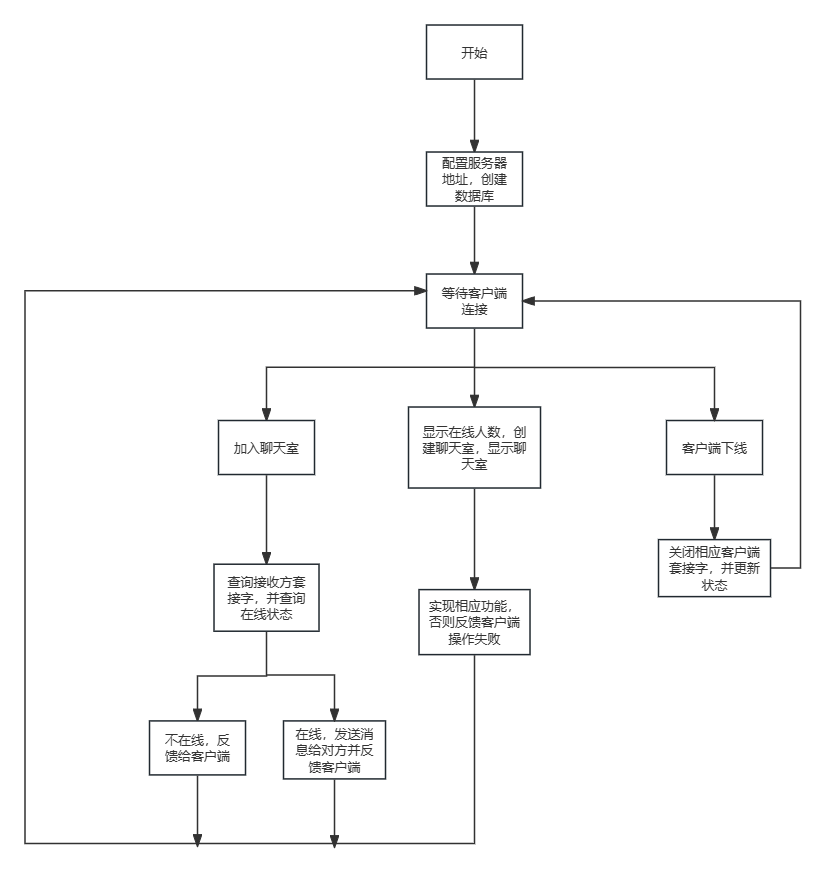
****

图4.3.1服务端端流程图

**4．3．2 聊天****等消息传输功能**

在这个聊天程序中，主要涉及两个方面的聊天消息：聊天室内的聊天和公屏聊天：

**1.聊天室内的聊天：**

## （1）群发消息： 在聊天室内，用户可以通过在公共频道发送消息来群发给聊天室的所有成员。这可以通过在公屏上直接输入消息实现。

## （2）私聊成员： 用户可以使用命令 chat to 来私聊特定成员。输入 chat to [用户名] [消息] 可以将消息直接发送给特定成员。

## （3）显示成员： 用户可以使用 show member 命令查看当前聊天室内的成员列表。

## 2. 公屏聊天：

## （1）私聊用户： 在公屏聊天中，用户可以通过 chat to 命令向特定用户发送私人消息。如果用户在线，消息将直接发送给用户；如果用户不在线，消息将被存储到数据库中，等到用户下次上线时进行发送。

## （2）发送文件和图片： 用户可以通过 send file to 命令发送文件，以及通过 send img to 命令发送图片。这可以通过在公屏上输入相应命令并提供文件或图片路径来实现。

## （3）公屏发消息： 用户可以直接在公屏上输入消息并发送，这些消息会被广播给所有在线用户。

## 这样的设计使得聊天程序更加灵活，支持不同形式的消息交互，并提供了私聊、群发、文件传输等功能。同时，采用数据库存储离线消息，确保用户在下次登录时可以接收到之前未读的消息。

## 4．4 数据库访问或文件操作

对于项目中的数据库使用的是Sqlite3，下面是数据库表创建的相关代码：  
void initDb()

{

// 打开或创建数据库

int rc = sqlite3\_open("chatDb.db", &db);

if (rc != SQLITE\_OK)

{

fprintf(stderr, "无法打开数据库：%s\n", sqlite3\_errmsg(db));

return 1;

}

// 创建信息表

char \*create\_table\_query = "CREATE TABLE IF NOT EXISTS messages ("

"id INTEGER PRIMARY KEY AUTOINCREMENT,"

"send\_name TEXT,"

"receive\_name TEXT,"

"message TEXT);";

rc = sqlite3\_exec(db, create\_table\_query, 0, 0, &error\_message);

if (rc != SQLITE\_OK)

{

fprintf(stderr, "无法创建表: %s\n", error\_message);

sqlite3\_free(error\_message);

return 1;

}

// 创建用户表

char \*create\_user\_table\_query = "CREATE TABLE IF NOT EXISTS users ("

"id INTEGER PRIMARY KEY AUTOINCREMENT,"

"username TEXT,"

"password TEXT,"

"isLogin TEXT DEFAULT 'false');";

rc = sqlite3\_exec(db, create\_user\_table\_query, 0, 0, &error\_message);

if (rc != SQLITE\_OK)

{

fprintf(stderr, "无法创建用户信息表: %s\n", error\_message);

sqlite3\_free(error\_message);

return 1;

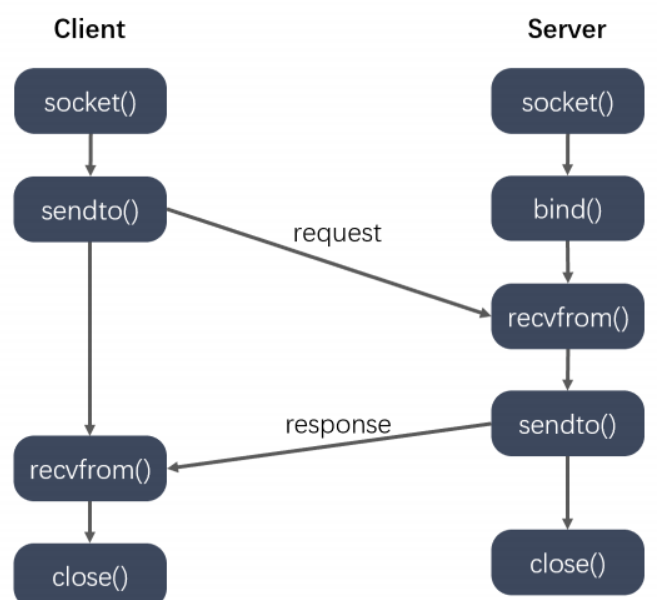
}

}

对于文件的操作使用的是c语言中的FILE指针及其相关的库实现的。

## 4．5 通信模块的实现

采用UDP传输



# **测试与调试**

# 5.1注册和登录功能

首先测试注册功能，假设我们输入的两组用户分别为：lyb和aaa，密码同用户名。

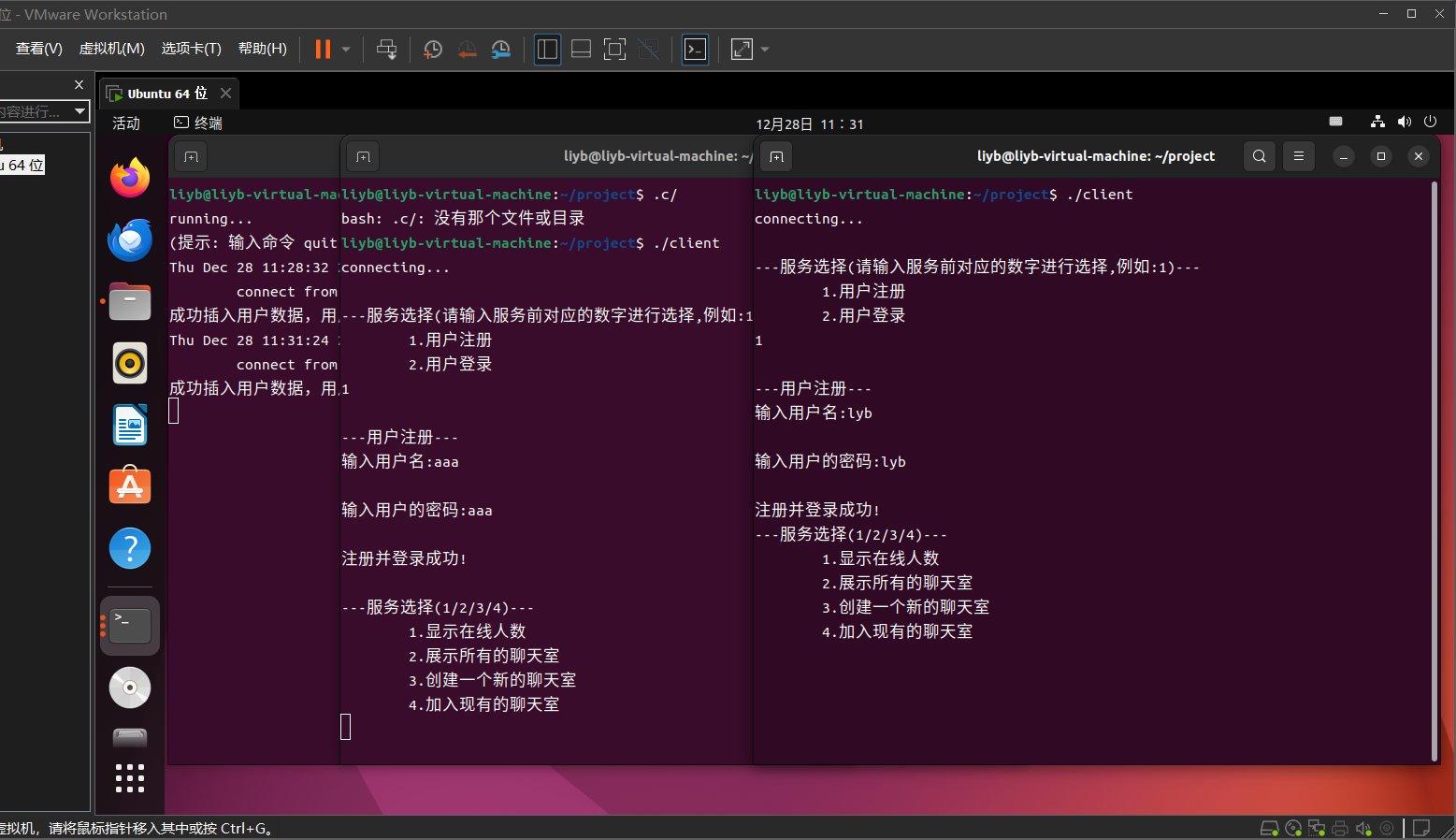


图5.1.1用户注册功能测试

然后测试用户登录功能，现在退出我们的用户aaa，然后重新进行登录

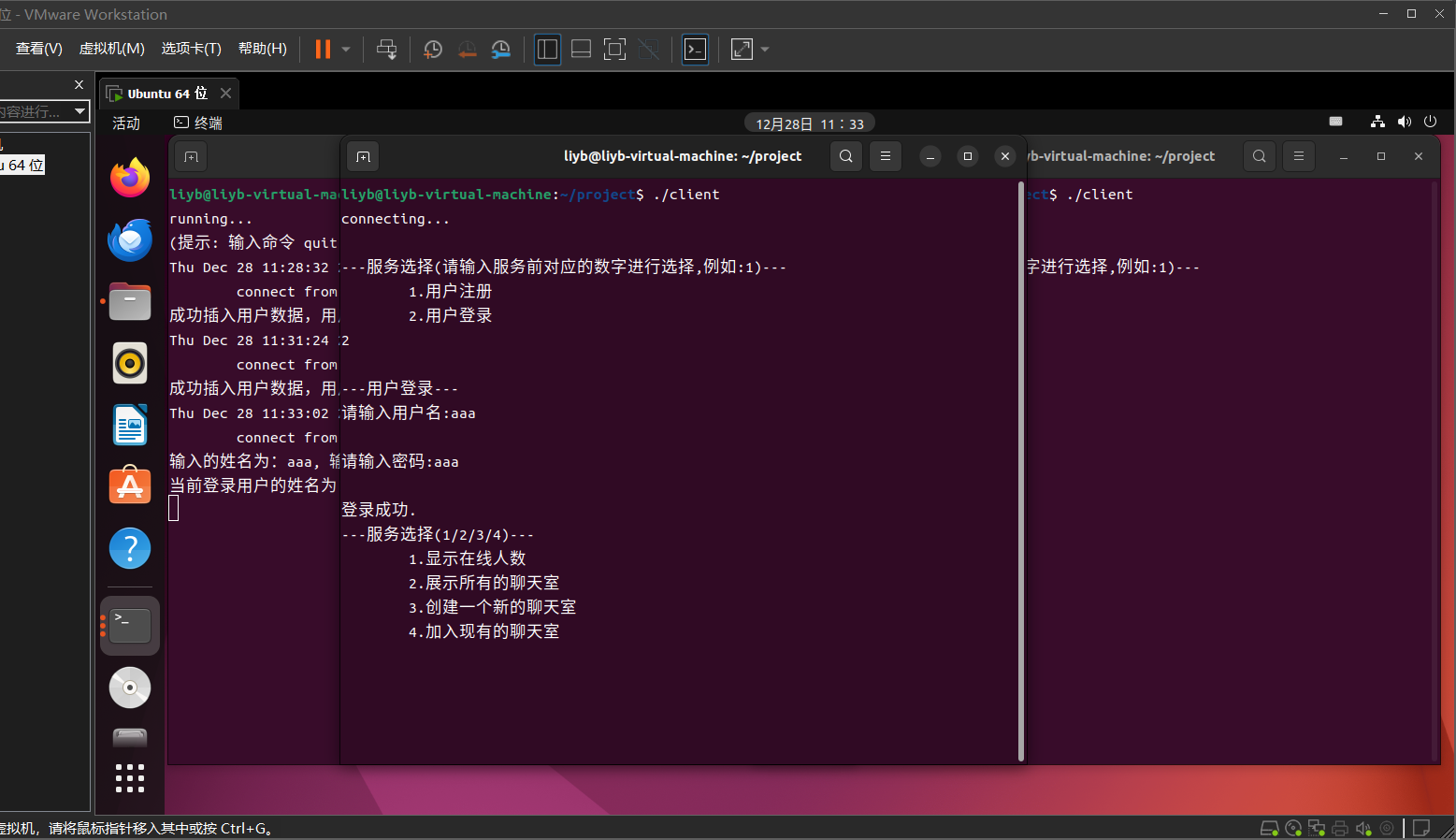
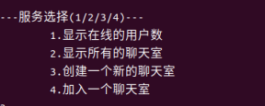


图5.1.2用户登录功能测试

**5.2聊天室功能**

聊天室有三种功能，可以显示所有聊天室，创建新的聊天室和加入一个聊天室



下面对创建聊天室功能来进行演示，输入3然后它会让你给聊天室起名和设定密码，创建成功会自动加入聊天室，此处聊天室的名称和密码都为ttt.

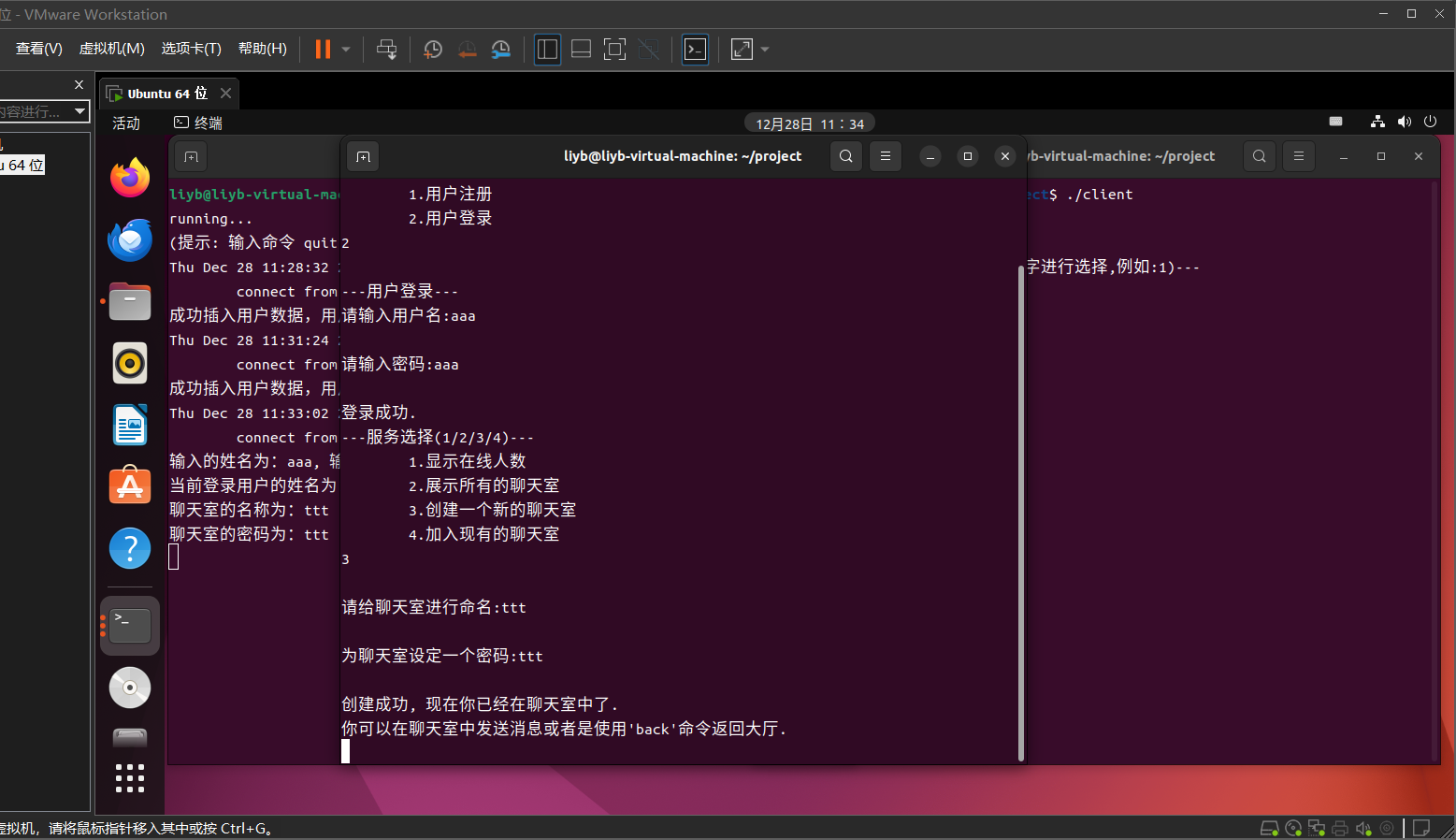


图5.2.1创建聊天室

然后显示所有的聊天室：

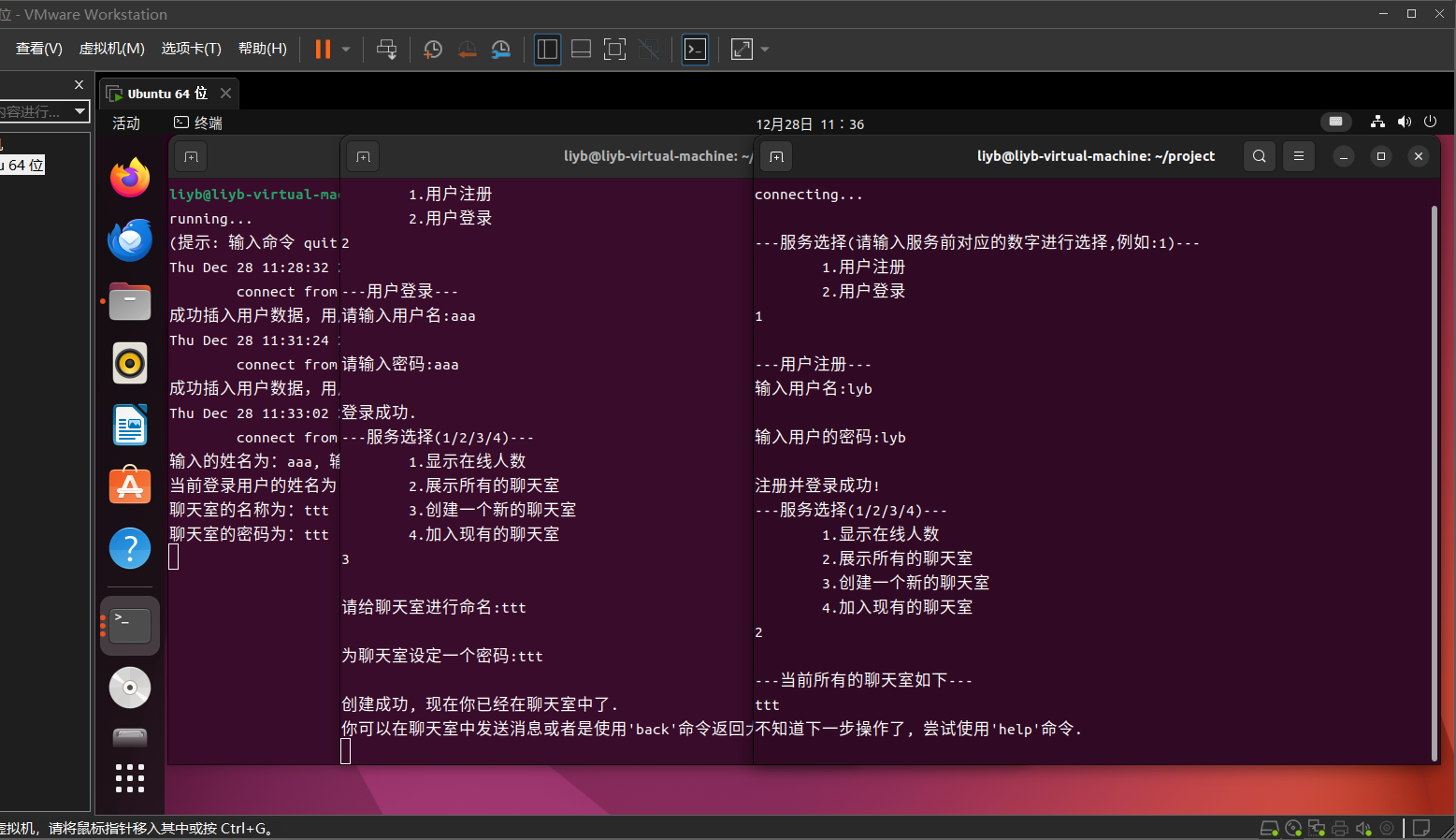


图5.2.2显示所有的聊天室

之后我们让用户lyb加入聊天室并进行聊天：

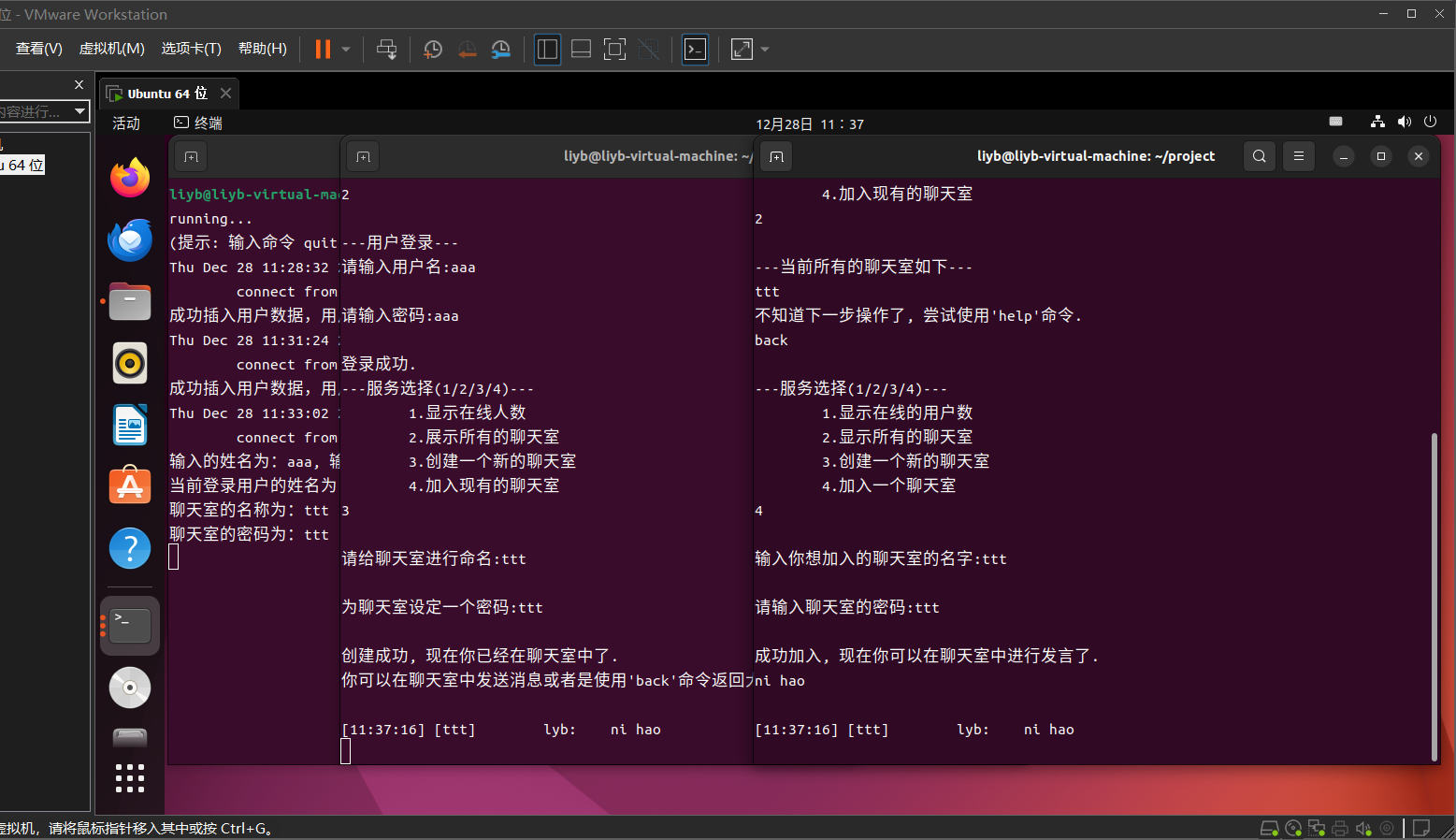


图5.2.3聊天室内聊天

显示聊天室内所有的用户：

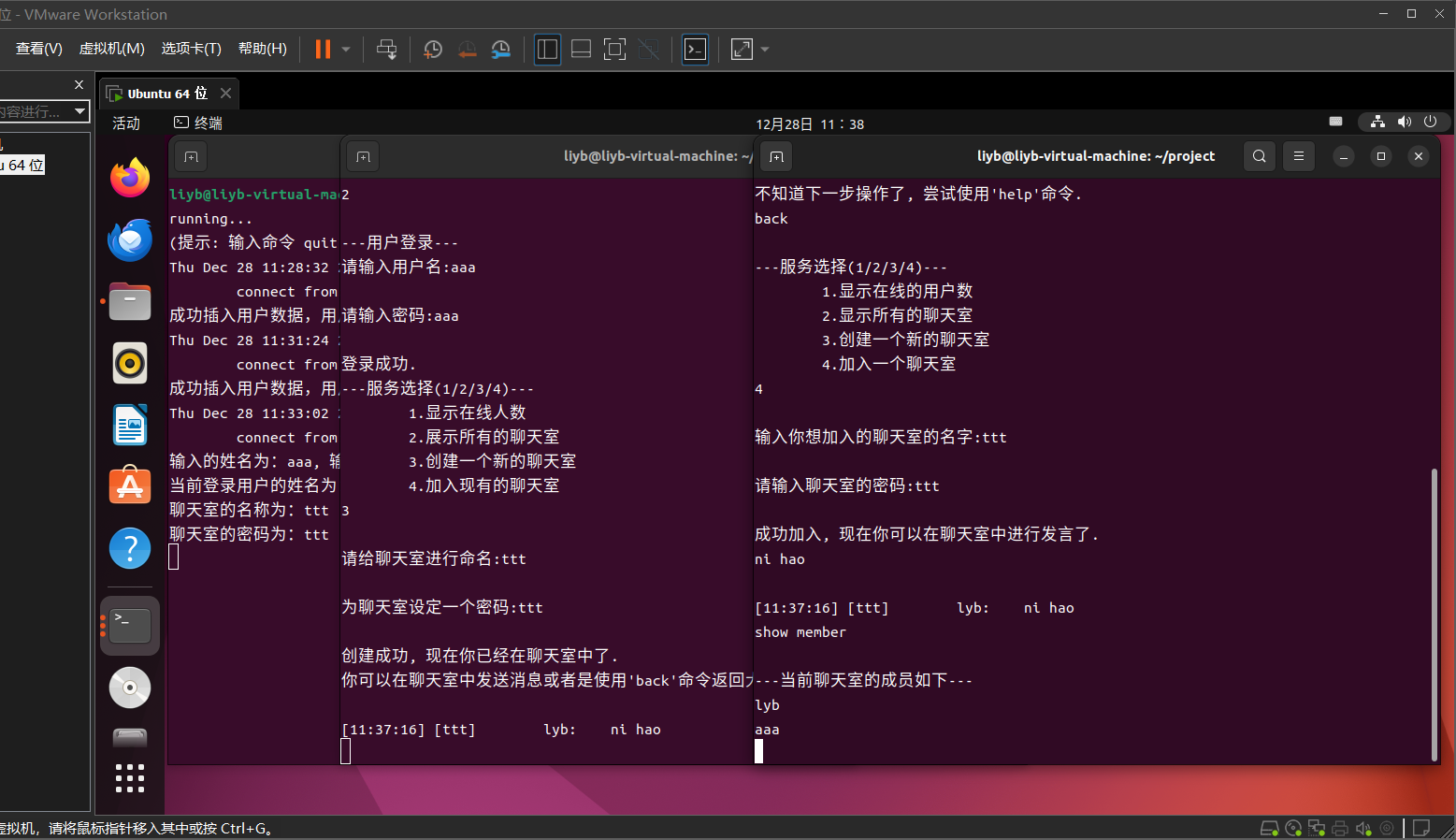


图5.2.4显示聊天室内所有成员

**5.3私聊功能**

为了体现私聊功能，我们再创建一个新的用户qwe，然后让用户aaa私聊用户qwe

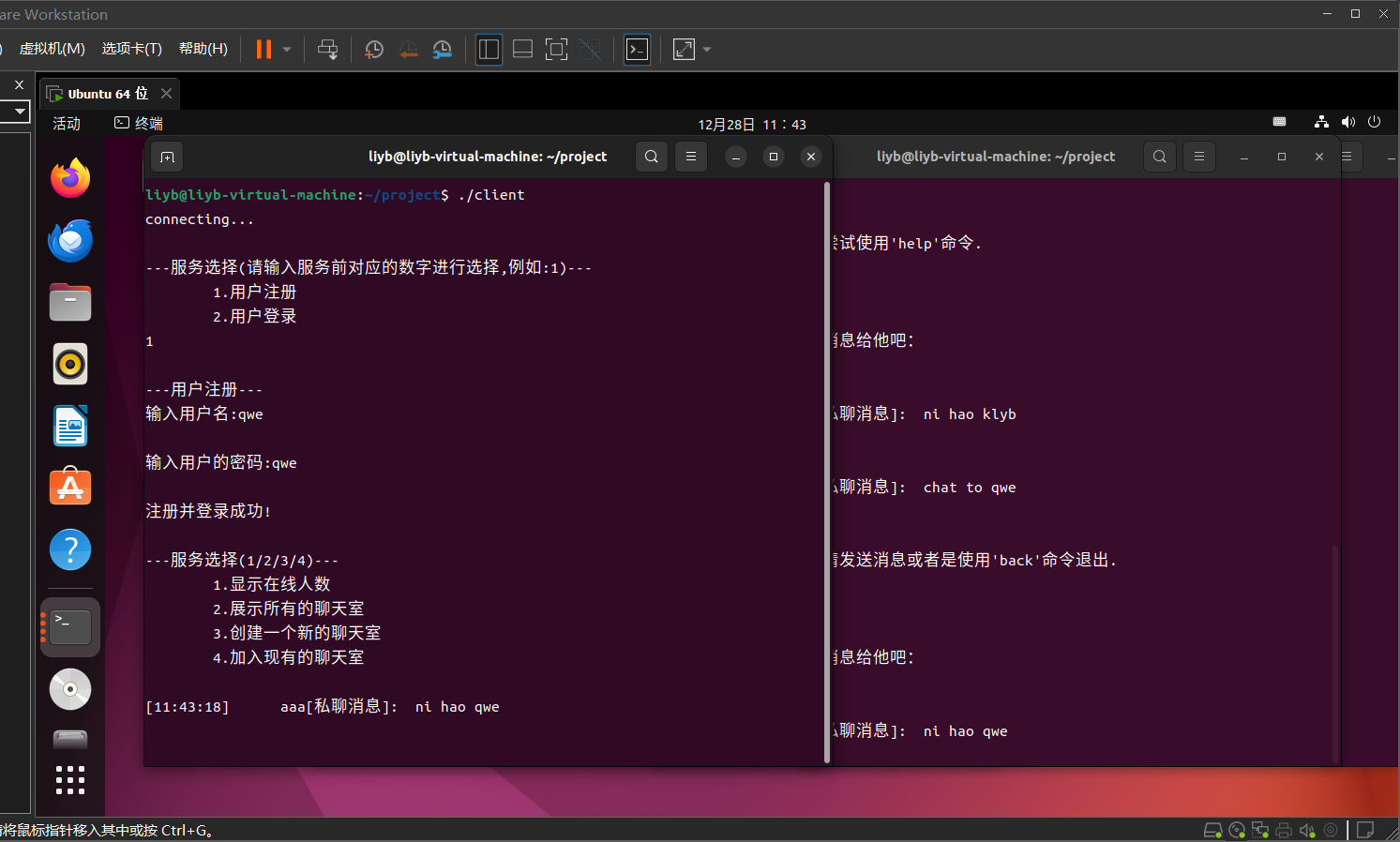


图5.3.1私聊功能

**5.4离线消息发送功能**

让用户qwe下线，然后用lyb去私聊他，当下次qwe上线的时候会自动将数据库中的消息显示出来并删除数据库中的消息。

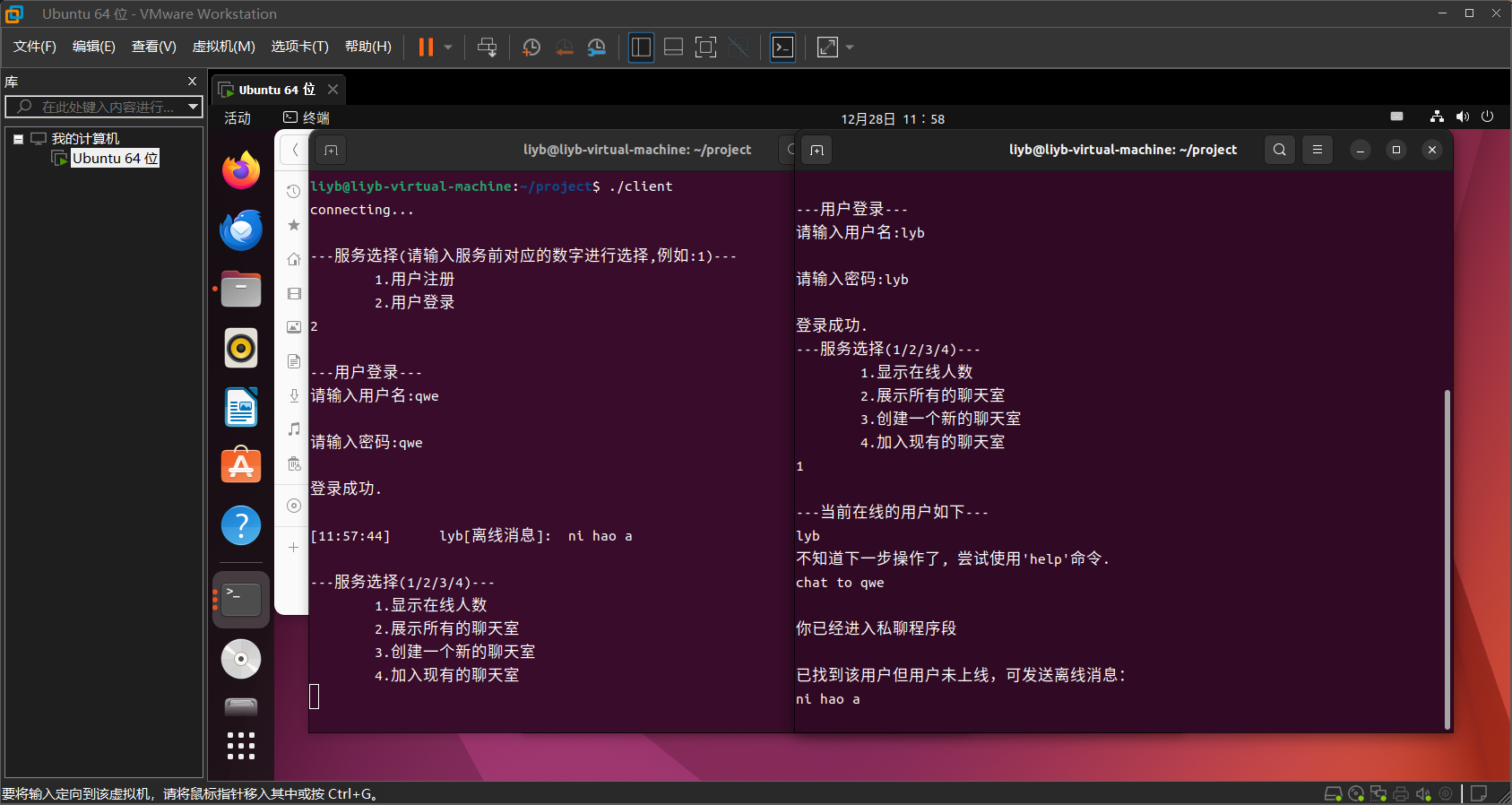


图5.4.1离线消息发送

**5.5文件发送功能**

用户lyb给qwe用户发送lyb用户目录下的文件

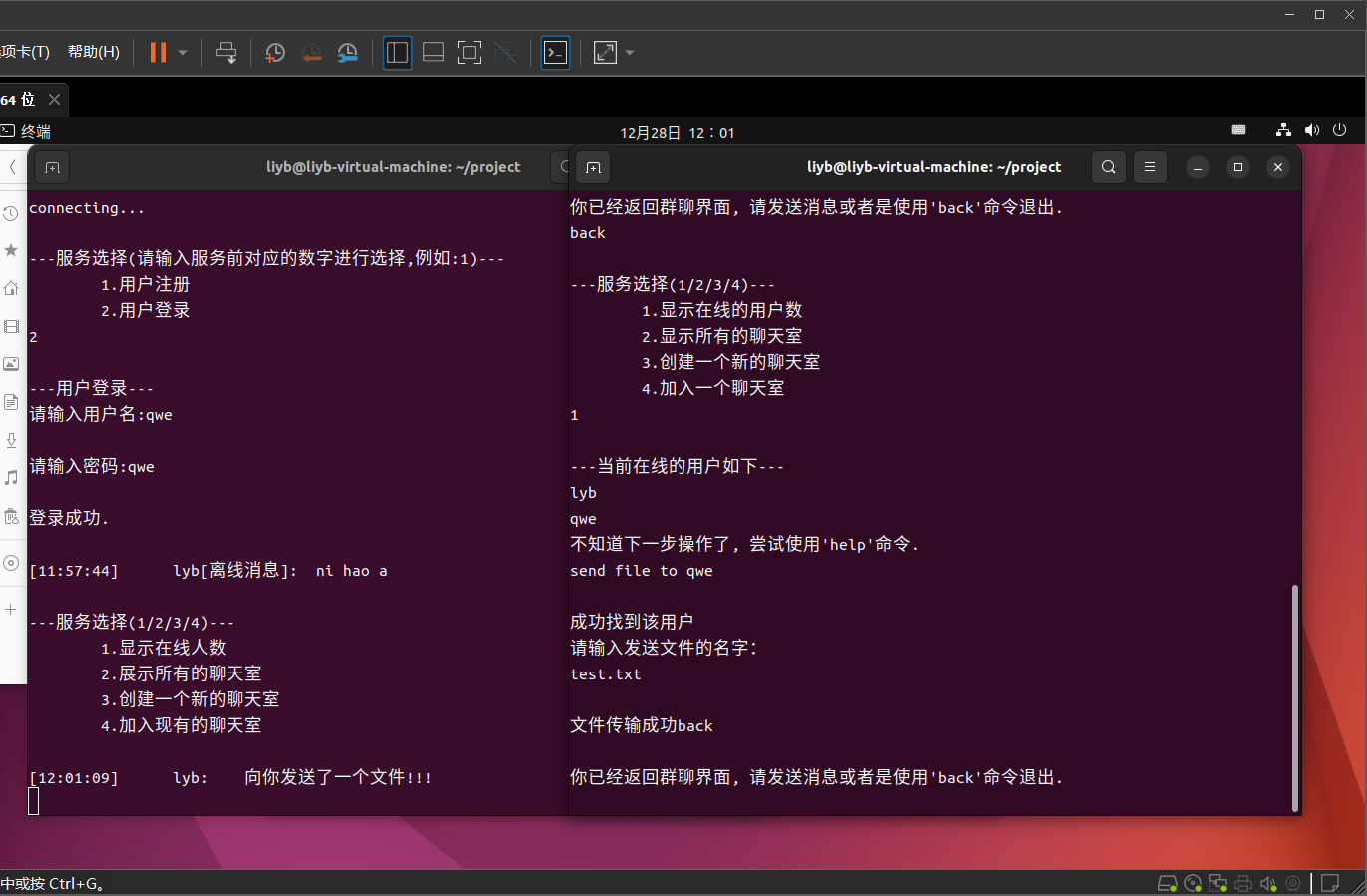


图5.5.1文件传输命令展示

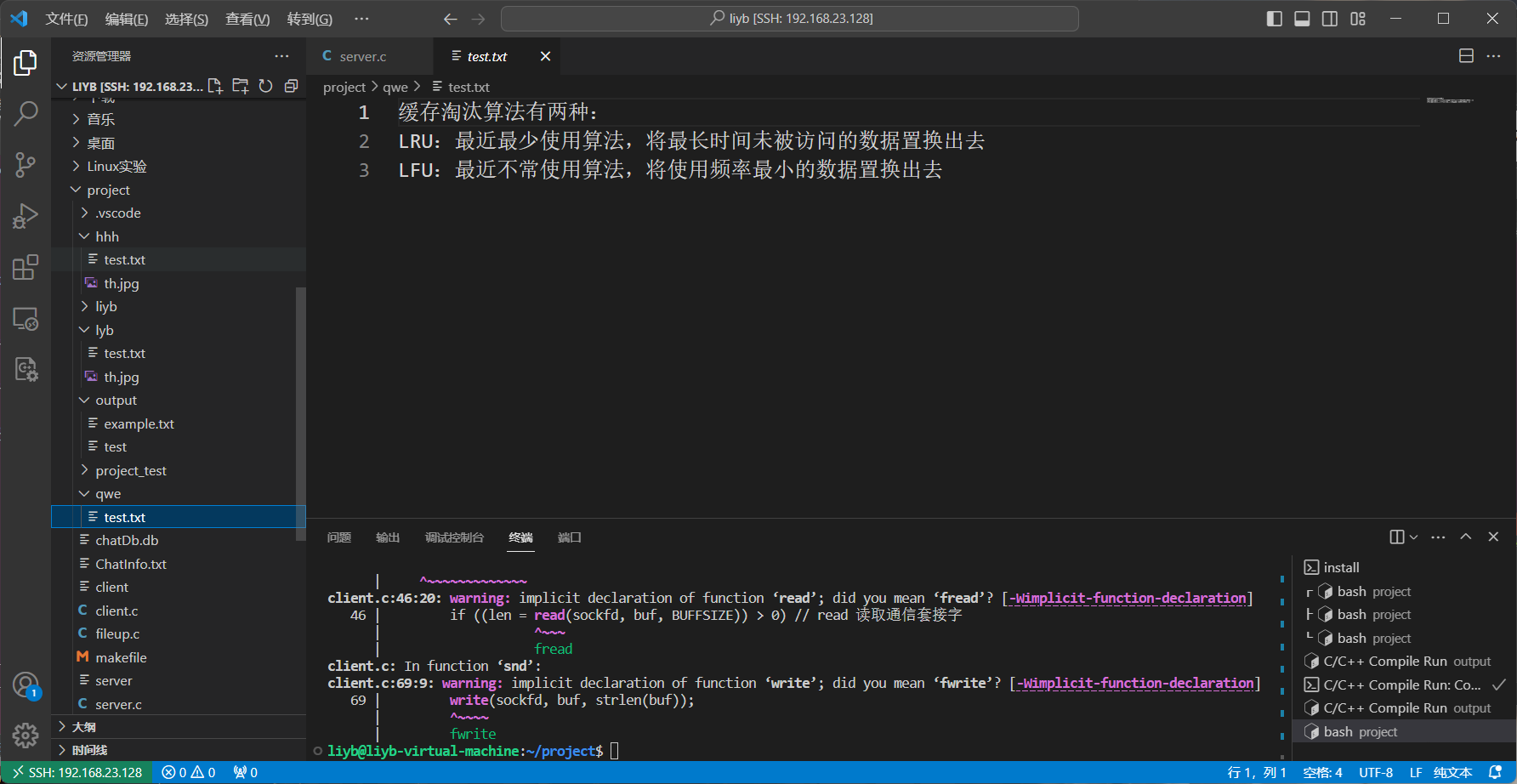


图5.5.2文件存放位置展示

**5.6图片发送功能**

用户lyb给qwe用户发送lyb目录下的文件

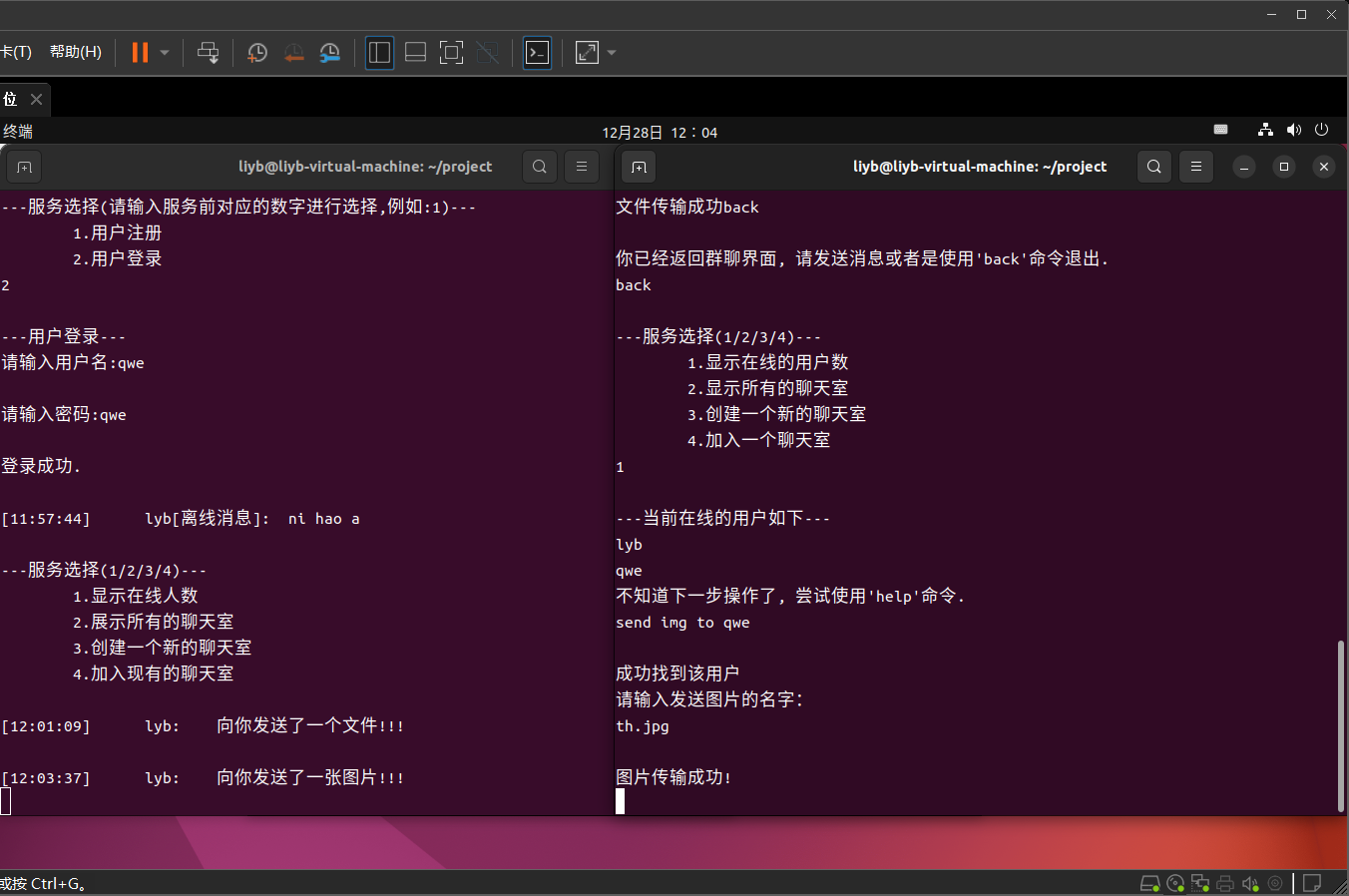


图5.6.1图片传输命令展示

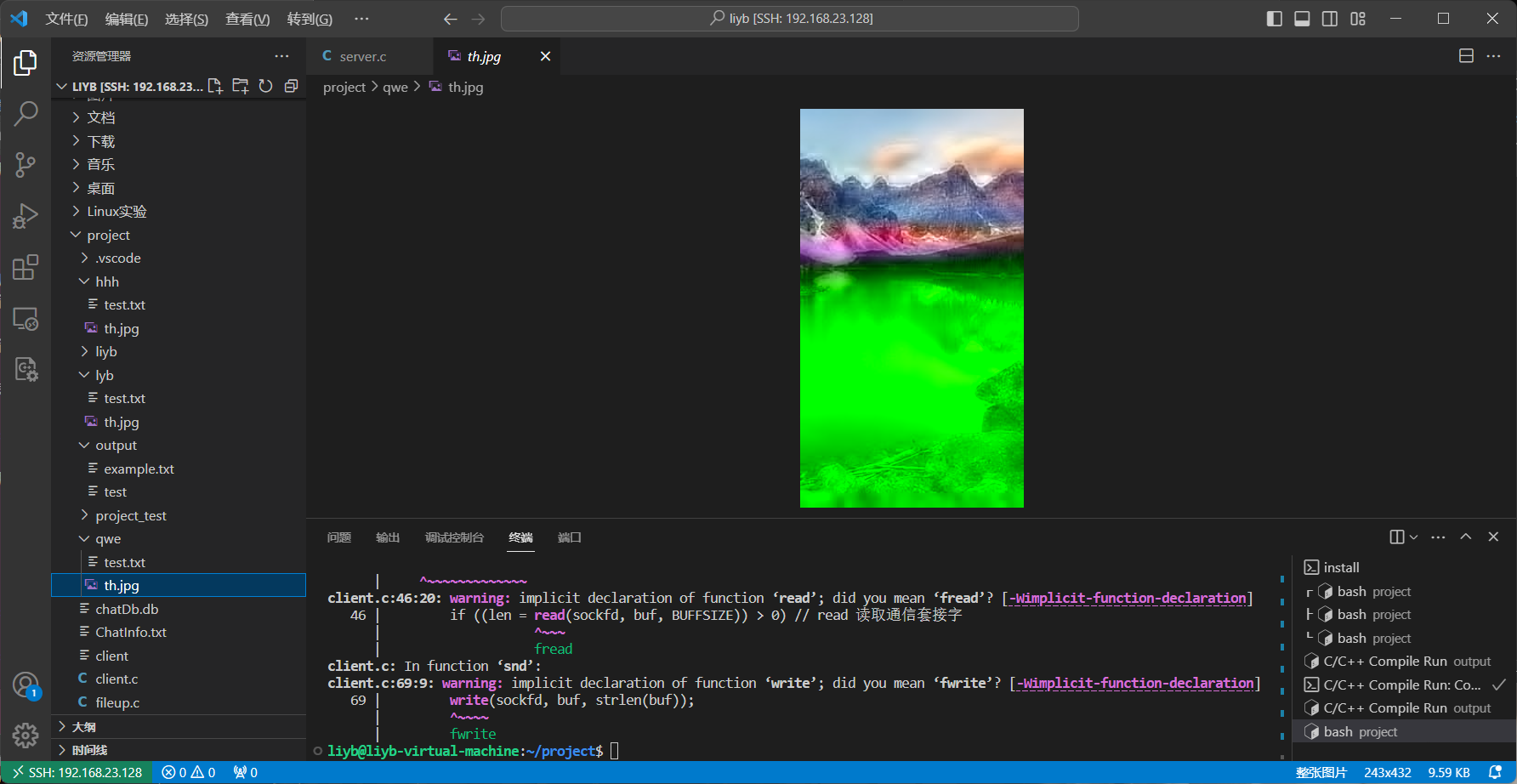


图5.6.2图片存放位置展示

### 6、总结

聊天程序设计最初的时候，主要是对相关资料的收集和理论知识的学习。在这个阶段，最好是同时结合资料和源码一起来看，效果会比较好,学习效率较高。看资料和教程是从细节和基础上去学习知识，而看相程序的源码则是从整体和实现上去了解一个系统。在具体实现的时候，可以以模块或分层次的思想来分析系统。重点掌握核心的模块，其他模块可以采用现有的类库实现，这样可以提高开发的效率。软件开发对于代码量的积累是很重要的。当积累了一定的代码量后，看问题就会比较有程序的思想，能够从层次，模块的角度来分析问题，这样思路就比较清晰了。

实验过程中，也对团队精神的进行了考察，让我们在合作起来更加默契，在成功后一起体会喜悦的心情。果然是团结就是力量，只有互相之间默契融洽的配合才能换来最终完美的结果。

此次设计也让我明白了思路即出路，有什么不懂不明白的地方要及时请教或上网查询，只要认真钻研，动脑思考，动手实践，就没有弄不懂的知识，收获颇丰。

### 附录：程序代码

Makefile:

main: server.c client.c

gcc -o server server.c -lpthread -lsqlite3

gcc -o client client.c -lpthread -lsqlite3

client.c:

int main()

{

pthread\_t thread; /\*pthread\_t 线程，gcc编译时需加上-lpthread\*/

struct sockaddr\_in serv\_addr; // struct sockaddr\_in

char buf[BUFFSIZE];

/\*初始化服务端地址结构\*/

memset(&serv\_addr, 0, sizeof(struct sockaddr\_in));

serv\_addr.sin\_family = AF\_INET; // sin\_family AF\_INET

serv\_addr.sin\_port = htons(PORT); // sin\_port htons(PORT)

inet\_pton(HOST\_IP, &serv\_addr.sin\_addr); // inet\_pton

// 创建客户端套接字

sockfd = socket(AF\_INET, SOCK\_STREAM, 0); // socket 创建套接字

if (sockfd < 0)

{

perror("fail to socket");

exit(-1);

}

// 与服务器建立连接

printf("connecting... \n");

if (connect(sockfd, (struct sockaddr \*)&serv\_addr, sizeof(serv\_addr)) < 0) // connect

{

perror("fail to connect");

exit(-2);

}

/\* === 主进程接收数据，从此处开始 程序分做两个线程 === \*/

// 创建发送消息的线程，调用发送消息的函数snd

pthread\_create(&thread, NULL, (void \*)(&snd), NULL); // pthread\_create

// 接收消息的线程,将服务器上发送的消息打印到客户端的屏幕上

while (1)

{

int len;

if ((len = read(sockfd, buf, BUFFSIZE)) > 0) // read 读取通信套接字

{

buf[len] = '\0'; // 添加结束符，避免显示缓冲区中残留的内容

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

fflush(stdout); // fflush 冲洗标准输出，确保内容及时显示

}

}

return 0;

}

/\*发送消息的函数\*/

void snd()

{

char temp[32], buf[BUFFSIZE];

while (1)

{

fgets(buf, BUFFSIZE, stdin);

buf[strcspn(buf, "\n")] = '\0'; // 清除读入的回车

// printf("客户端输入的信息为：%s", buf);

write(sockfd, buf, strlen(buf));

memset(buf, 0, sizeof(buf));

// printf("传输完成后的信息为：%s", buf);

/\* 客户端输入bye则退出 \*/

if (strcmp(buf, "bye\n") == 0) // 注意此处的\n

exit(0);

}

}

Server.c:

// 当前函数是用于服务器和客户端进行交互操作的

void rcv\_snd(int n)

{

int len, i;

char buf[BUFFSIZE], mytime[32], v[5]; /\*v用接受服务命令字符\*/

time\_t ticks;

int ret; // 存储当线程的退出状态信息

write(connfd[n], "---服务选择(请输入服务前对应的数字进行选择,例如:1)---\n\t1.用户注册\n\t2.用户登录\n",

strlen("---服务选择(请输入服务前对应的数字进行选择,例如:1)---\n\t1.用户注册\n\t2.用户登录\n"));

len = read(connfd[n], v, 1); /\* 读取客户端输入的１或者２命令 \*/

v[len] = '\0';

if (strcmp(v, "1") == 0)

{ /\*如果选择注册服务\*/

strcpy(buf, "---用户注册---\n");

/\*在客户端显示当前状态\*/

write(connfd[n], buf, strlen(buf));

regist(n);

}

else if (strcmp(v, "2") == 0)

{ /\*如果选择登录服务\*/

strcpy(buf, "---用户登录---\n");

/\*在客户端显示当前状态\*/

write(connfd[n], buf, strlen(buf));

login(n);

}

else

{

write(connfd[n], "无效的命令，请重新进行输入.\n",

strlen("无效的命令，请重新进行输入.\n"));

rcv\_snd(n); /\* 输入其他值时重新进入服务选项 \*/

}

// TODO: 对发送过来的消息进行显示

selectDataFromDbMessage(db, info\_class[n].name, n); // 查询是否有人发送消息给自己

/\* 登陆成功后服务提示，这里就是大厅 \*/

write(connfd[n], "---服务选择(1/2/3/4)---\n\t1.显示在线人数\n\t2.展示所有的聊天室\n\t3.创建一个新的聊天室\n\t4.加入现有的聊天室\n",

strlen("---服务选择(1/2/3/4)---\n\t1.显示在线人数\n\t2.展示所有的聊天室\n\t3.创建一个新的聊天室\n\t4.加入现有的聊天室\n\t"));

show(n); // 对上面用户进行的指令分别进行处理

// 在大厅处理完命令后开始对用户的输入进行监听，默认情况下进入群聊系统

while (1)

{

char temp[BUFFSIZE];

char \*str;

if ((len = read(connfd[n], temp, BUFFSIZE)) > 0)

{

// printf("客户端得到的输入为：%s", temp);

temp[len] = '\0';

/\* 当用户输入bye时，当前用户退出 \*/

if (strcmp(temp, "bye") == 0)

{

close(connfd[n]);

connfd[n] = -1;

pthread\_exit(&ret);

printf("执行bye指令，当前的connfd的值为：%d\n", connfd[n]);

}

else if (strcmp(temp, "help") == 0)

{

memset(buf, 0, sizeof(buf));

strcpy(buf, "\t你可以尝试输入下面的命令.\n");

write(connfd[n], buf, strlen(buf));

strcpy(buf, "[chat to user1]: 和某一个用户进行私聊.\n");

write(connfd[n], buf, strlen(buf));

strcpy(buf, "[back]: 返回大厅.\n");

write(connfd[n], buf, strlen(buf));

strcpy(buf, "[bye]: 断开和服务器的连接.\n");

write(connfd[n], buf, strlen(buf));

}

else if (strcmp(temp, "back") == 0)

{

write(connfd[n], "---服务选择(1/2/3/4)---\n\t1.显示在线的用户数\n\t2.显示所有的聊天室\n\t3.创建一个新的聊天室\n\t4.加入一个聊天室\n",

strlen("---服务选择(1/2/3/4)---\n\t1.显示在线的用户数\n\t2.显示所有的聊天室\n\t3.创建一个新的聊天室\n\t4.加入一个聊天室\n"));

memset(info\_class[n].chatroom, 0, sizeof(info\_class[n].chatroom)); /\* 返回大厅后即退出聊天室 \*/

show(n);

}

/\* 如果当前客户端在聊天室内 \*/

else if (strlen(info\_class[n].chatroom) != 0)

{

/\* 如果发现关键字符串"chat to"，从聊天室转入私聊程序段 \*/

if ((str = strstr(temp, "chat to ")) != 0)

{

char des[10]; /\* 用来存储目标用户的名称，从chat to user中获取 \*/

strncpy(des, temp + 8, sizeof(des) - 1);

memset(temp, 0, sizeof(temp));

int flag = 0; // 默认是不存在的

// 首先在所有上线的用户中查找是否存在当前用户

for (i = 0; i < MAXMEM; i++)

{

if (strcmp(des, info\_class[i].name) == 0) // 找到要私聊的对象

{

flag = 1;

// 循环处理当前请求

while (1)

{

memset(temp, 0, sizeof(temp));

if ((len = read(connfd[n], temp, BUFFSIZE)) > 0) // 从发送方的客户端不断进行读取消息

{

temp[len] = '\0';

if (strcmp(temp, "back") == 0)

{ /\* 如果收到back字段，则返回群聊 \*/

write(connfd[n], "你已经返回群聊界面, 请发送消息或者是使用'back'命令退出.\n",

strlen("你已经返回群聊界面, 请发送消息或者是使用'back'命令退出.\n"));

break;

}

else

{

// strcat用来实现字符串追加

ticks = time(NULL);

strftime(mytime, sizeof(mytime), "[%H:%M:%S]", localtime(&ticks));

strcpy(buf, mytime); // 显示时间

strcat(buf, "\t");

strcat(buf, info\_class[n].name); // 将发送方的名字追加过去，即显示姓名

strcat(buf, ":\t");

strcat(buf, temp); // 客户端消息内容

temp[len] = '\0';

strcat(buf, "\n");

write(connfd[i], buf, strlen(buf));

write(connfd[n], buf, strlen(buf));

}

}

}

}

}

}

else if (strcmp(temp, "show member") == 0)

{ /\* show member显示聊天室内成员 \*/

char member[50];

strcpy(member, "---当前聊天室的成员如下---\n");

for (int i = 0; i < MAXMEM; i++)

{

if (strcmp(info\_class[i].chatroom, info\_class[n].chatroom) == 0)

{

strcat(member, info\_class[i].name); /\* 将在线用户名添加到info \*/

strcat(member, "\n");

}

}

write(connfd[n], member, strlen(member));

}

else

{ /\* 不是私聊那就是在聊天室内群发 \*/

for (i = 0; i < MAXMEM; i++)

{

/\* 寻找结构体中聊天室名相同的发送信息 \*/

if (strcmp(info\_class[i].chatroom, info\_class[n].chatroom) == 0)

{

char chatroom[20];

strcpy(chatroom, info\_class[n].chatroom);

chatroom[strlen(chatroom) - 1] = '\0'; // 聊天室名称多加了一个回车

ticks = time(NULL);

strftime(mytime, sizeof(mytime), "[%H:%M:%S] [", localtime(&ticks));

strcpy(buf, mytime); // 显示时间

strcat(buf, chatroom); // 显示群聊名称

strcat(buf, "]\t");

strcat(buf, info\_class[n].name); // 显示用户名

strcat(buf, ":\t");

strcat(buf, temp); // 客户端消息内容

temp[len] = '\0';

strcat(buf, "\n");

write(connfd[i], buf, strlen(buf)); /\* 将消息发送给聊天室内的每一个人 \*/

}

}

}

}

/\* 如果发现关键字符串"chat to"，转入私聊程序段 \*/

else if ((str = strstr(temp, "chat to ")) != 0)

{

int x; /\* 查找chat to后跟用户名的开始下标值 \*/

char des[10]; /\* 用来存储目标用户的名称，从chat to user中获取 \*/

x = strspn(temp, "chat to "); /\* chat to 后面用户名字符串开始的索引 \*/

strncpy(des, temp + x, strlen(temp) - x + 1); // 将用户名复制到des中

int flag = 0;

write(connfd[n], "你已经进入私聊程序段\n", strlen("你已经进入私聊程序段\n"));

memset(temp, 0, sizeof(temp));

for (i = 0; i < MAXMEM; i++)

{

if (strcmp(des, info\_class[i].name) == 0)

{ /\* 遍历匹配user1的姓名 \*/

flag = 1;

write(connfd[n], "找到该在线用户，发送消息给他吧：\n", strlen("找到该在线用户，发送消息给他吧：\n"));

while (1)

{

if ((len = read(connfd[n], temp, BUFFSIZE)) > 0) // 等待发送方输入下一步的指令

{

temp[len] = '\0';

printf("用户输入的消息为：%s", temp);

if (strcmp(temp, "back") == 0)

{ /\* 如果收到back字段，则返回大厅 \*/

write(connfd[n], "你已经返回群聊界面, 请发送消息或者是使用'back'命令退出.\n",

strlen("你已经返回群聊界面, 请发送消息或者是使用'back'命令退出.\n"));

break;

}

else

{

ticks = time(NULL);

strftime(mytime, sizeof(mytime), "[%H:%M:%S]", localtime(&ticks));

strcpy(buf, mytime); // 显示时间

strcat(buf, "\t");

strcat(buf, info\_class[n].name); // 显示用户名

strcat(buf, "[私聊消息]:\t");

strcat(buf, temp); // 客户端消息内容

temp[len] = '\0';

strcat(buf, "\n");

printf("我被执行了！");

write(connfd[i], buf, strlen(buf));

write(connfd[n], buf, strlen(buf));

}

}

}

}

if (flag)

{

break;

}

}

// TODO: 现在已经从数据库中查询出来了用户，只要不断从发送方接受数据然后存储到数据库中就行了

if (!flag)

{

struct user us;

int result = selectUserFromDbUser(db, des, &us);

printf("这里的result值为：%d\n", result);

printf("从数据库中查询出来用户名为：%s", us.username);

if (result == 1)

{

printf("用户未上线，但在数据库中查询到当前用户存在。\n");

write(connfd[n], "已找到该用户但用户未上线，可发送离线消息：\n",

strlen("已找到该用户但用户未上线，可发送离线消息：\n"));

flag = 1;

// 接收发送方的消息

while (1)

{

if ((len = read(connfd[n], temp, BUFFSIZE)) > 0)

{

temp[len] = '\0';

if (strcmp(temp, "back") == 0)

{ /\* 如果收到back字段，则返回大厅 \*/

write(connfd[n], "你已经返回群聊界面, 请发送消息或者是使用'back'命令退出.\n",

strlen("你已经返回群聊界面, 请发送消息或者是使用'back'命令退出.\n"));

break;

}

else

{

ticks = time(NULL);

strftime(mytime, sizeof(mytime), "[%H:%M:%S]", localtime(&ticks));

strcpy(buf, mytime); // 显示时间

strcat(buf, "\t");

strcat(buf, info\_class[n].name); // 显示用户名

strcat(buf, "[离线消息]:\t");

strcat(buf, temp); // 客户端消息内容

temp[len] = '\0';

strcat(buf, "\n");

printf("消息的内容如下：\n%s", buf);

// 将消息写入数据库中进行保存，当用户上线的时候进行显示

addDataToDbMessage(db, info\_class[n].name, us.username, buf);

}

}

}

}

else if (result == 0)

{

printf("用户不存在。\n");

}

else

{

fprintf(stderr, "检查用户是否存在时出错。\n");

}

}

if (!flag)

write(connfd[n], "未找到该用户，请重新键入命令输入\n",

strlen("未找到该用户，请重新键入命令输入\n"));

}

// TODO : recent 实现文件的发送功能

else if ((str = strstr(temp, "send file to ")) != 0)

{

printf("文件发送模块中接受者的姓名为：%s\n", temp);

char des[50]; /\* 用来存储目标用户的名称，从send file to user中获取 \*/

strncpy(des, temp + 13, sizeof(des));

printf("得到的des中的值为：%s\n", des);

int flag = 0;

// 文件存储的目录位置默认为该用户的用户名

char directory\_name[100];

memcpy(directory\_name, des, sizeof(des));

// 当前目录下文件的名字

char file\_name[100];

// 文件的数据

char data[MAXCOUNT];

printf("收件人的姓名为：%s\n", directory\_name);

for (int i = 0; i < MAXMEM; i++)

{

printf("用户%d的姓名为：%s,长度为：%d\n", i, info\_class[i].name, strlen(info\_class[i].name));

printf("%d\n", strcmp(des, info\_class[i].name));

if (connfd[i] != -1 && strcmp(des, info\_class[i].name) == 0)

{ /\* 遍历匹配user1的姓名 \*/

write(connfd[n], "成功找到该用户\n", strlen("成功找到该用户\n"));

write(connfd[n], "请输入发送文件的名字：\n", strlen("请输入发送文件的名字：\n"));

flag = 1;

while (1)

{

if ((len = read(connfd[n], temp, BUFFSIZE)) > 0)

{

temp[len] = '\0';

if (strcmp(temp, "back") == 0)

{ /\* 如果收到back字段，则返回大厅 \*/

write(connfd[n], "你已经返回群聊界面, 请发送消息或者是使用'back'命令退出.\n",

strlen("你已经返回群聊界面, 请发送消息或者是使用'back'命令退出.\n"));

break;

}

else

{

strcpy(file\_name, temp);

ticks = time(NULL);

strftime(mytime, sizeof(mytime), "[%H:%M:%S]", localtime(&ticks));

strcpy(buf, mytime); // 显示时间

strcat(buf, "\t");

strcat(buf, info\_class[n].name); // 显示用户名

strcat(buf, ":\t");

strcat(buf, "向你发送了一个文件!!!\n");

write(connfd[i], buf, strlen(buf));

// 将发送方中文件的内容复制到数组中

char send\_path[1000] = "/home/liyb/project/";

strcat(send\_path, info\_class[n].name);

strcat(send\_path, "/");

strcat(send\_path, file\_name);

FILE \*send\_file = fopen(send\_path, "r");

if (send\_file == NULL)

{

write(connfd[n], "文件传输失败,请重试\n",

strlen("文件传输失败,请重试\n"));

}

fseek(send\_file, 0, SEEK\_END);

int file\_size = ftell(send\_file);

fseek(send\_file, 0, SEEK\_SET);

size\_t bytes\_read = fread(data, 1, file\_size, send\_file);

data[bytes\_read] = '\0';

fclose(send\_file);

printf("文件中的数据为：\n%s\n", data);

// 将内容传送到接收方的目录中

// TODO：这部分内容应该在客户端实现，因为应该是在客户端存储对应的文件

char dir\_path[1000] = "/home/liyb/project/";

strcat(dir\_path, directory\_name);

// 检查目录是否存在

if (access(dir\_path, F\_OK) == 0)

{

// 目录存在，下载文件

strcat(dir\_path, "/");

strcat(dir\_path, file\_name);

printf("存储的目录为；%s", dir\_path);

FILE \*file = fopen(dir\_path, "w");

if (file != NULL)

{

printf("文件创建成功！\n");

fprintf(file, "%s", data);

fclose(file);

write(connfd[n], "文件传输成功!\n",

strlen("文件传输成功!\n"));

}

else

{

printf("无法创建文件");

}

}

else

{

printf("%s\n", dir\_path);

// 目录不存在，可以进行创建

int status = mkdir(dir\_path, S\_IRWXU | S\_IRWXG | S\_IROTH | S\_IXOTH | S\_IRWXU);

if (status == 0)

{

printf("目录被成功创建！.\n");

// 在当前目录下下载该文件

strcat(dir\_path, "/");

strcat(dir\_path, file\_name);

FILE \*file = fopen(dir\_path, "w");

if (file != NULL)

{

printf("文件创建成功！");

fprintf(file, "%s", data);

fclose(file);

write(connfd[n], "文件传输成功！",

strlen("文件传输成功"));

}

else

{

printf("无法创建文件");

}

}

else

{

write(connfd[i], "文件目录创建失败，请重试",

strlen("文件目录创建失败，请重试"));

}

}

}

}

}

}

}

if (!flag)

{

write(connfd[n], "未找到该用户，请重新键入命令\n",

strlen("未找到该用户，请重新键入命令\n"));

}

}

// TODO: 实现图片的发送

else if ((str = strstr(temp, "send img to ")) != 0)

{

printf("图片发送模块中接受者的姓名为：%s\n", temp);

char des[50]; /\* 用来存储目标用户的名称，从send image to user中获取 \*/

strncpy(des, temp + 12, sizeof(des) - 1);

printf("得到的des中的值为：%s\n", des);

int flag = 0;

// 文件存储的目录位置默认为该用户的用户名

char directory\_name[100];

memcpy(directory\_name, des, sizeof(des));

// 当前目录下文件的名字

char file\_name[100];

// 文件的数据

char data[MAXCOUNT];

printf("收件人的姓名为：%s\n", directory\_name);

for (int i = 0; i < MAXMEM; i++)

{

printf("用户%d的姓名为：%s,长度为：%d\n", i, info\_class[i].name, strlen(info\_class[i].name));

printf("%d\n", strcmp(des, info\_class[i].name));

if (connfd[i] != -1 && strcmp(des, info\_class[i].name) == 0)

{ /\* 遍历匹配user1的姓名 \*/

write(connfd[n], "成功找到该用户\n", strlen("成功找到该用户\n"));

write(connfd[n], "请输入发送图片的名字：\n", strlen("请输入发送图片的名字：\n"));

flag = 1;

while (1)

{

if ((len = read(connfd[n], temp, BUFFSIZE)) > 0)

{

temp[len] = '\0';

if (strcmp(temp, "back") == 0)

{ /\* 如果收到back字段，则返回大厅 \*/

write(connfd[n], "你已经返回群聊界面, 请发送消息或者是使用'back'命令退出.\n",

strlen("你已经返回群聊界面, 请发送消息或者是使用'back'命令退出.\n"));

break;

}

else

{

strcpy(file\_name, temp);

ticks = time(NULL);

strftime(mytime, sizeof(mytime), "[%H:%M:%S]", localtime(&ticks));

strcpy(buf, mytime); // 显示时间

strcat(buf, "\t");

strcat(buf, info\_class[n].name); // 显示用户名

strcat(buf, ":\t");

strcat(buf, "向你发送了一张图片!!!\n");

write(connfd[i], buf, strlen(buf));

// 将发送方中图片的内容复制到数组中

char send\_path[1000] = "/home/liyb/project/";

printf("发送方的姓名为：");

strcat(send\_path, info\_class[n].name);

strcat(send\_path, "/");

strcat(send\_path, file\_name);

printf("发送方的路径为：%s\n", send\_path);

FILE \*send\_file = fopen(send\_path, "rb");

if (send\_file == NULL)

{

write(connfd[n], "图片传输失败\n",

strlen("图片传输失败\n"));

return 1;

}

fseek(send\_file, 0, SEEK\_END);

int file\_size = ftell(send\_file);

fseek(send\_file, 0, SEEK\_SET);

size\_t bytes\_read = fread(data, 1, file\_size, send\_file);

fclose(send\_file);

// 将内容传送到接收方的目录中

// TODO：这部分内容应该在客户端实现，因为应该是在客户端存储对应的图片

char dir\_path[1000] = "/home/liyb/project/";

strcat(dir\_path, directory\_name);

printf("%s\n", dir\_path);

// 检查目录是否存在

if (access(dir\_path, F\_OK) == 0)

{

// 目录存在，下载图片

strcat(dir\_path, "/");

strcat(dir\_path, file\_name);

printf("存储的目录为：%s", dir\_path);

FILE \*file = fopen(dir\_path, "wb");

if (file != NULL)

{

printf("图片创建成功！\n");

fwrite(data, 1, bytes\_read, file);

fclose(file);

write(connfd[n], "图片传输成功!\n",

strlen("图片传输成功!\n"));

}

else

{

printf("无法创建图片文件\n");

}

}

else

{

printf("%s\n", dir\_path);

// 目录不存在，可以进行创建

int status = mkdir(dir\_path, S\_IRWXU | S\_IRWXG | S\_IROTH | S\_IXOTH | S\_IRWXU);

if (status == 0)

{

printf("目录被成功创建！.\n");

// 在当前目录下下载该图片

strcat(dir\_path, "/");

strcat(dir\_path, file\_name);

FILE \*file = fopen(dir\_path, "wb");

if (file != NULL)

{

printf("图片创建成功！");

fwrite(data, 1, bytes\_read, file);

fclose(file);

write(connfd[n], "图片传输成功！\n",

strlen("图片传输成功！\n"));

}

else

{

printf("无法创建图片文件\n");

}

}

else

{

write(connfd[i], "图片目录创建失败，请重试\n",

strlen("图片目录创建失败，请重试\n"));

}

}

}

}

}

}

}

if (!flag)

{

write(connfd[n], "未找到该用户，请重新键入命令\n",

strlen("未找到该用户，请重新键入命令\n"));

}

}

/\* 默认状态下在大厅消息将发送给所有在线用户 \*/

else

{

ticks = time(NULL);

strftime(mytime, sizeof(mytime), "[%H:%M:%S]", localtime(&ticks));

strcpy(buf, mytime); // 显示时间

strcat(buf, "\t");

strcat(buf, info\_class[n].name); // 显示用户名

strcat(buf, ":[群聊消息]\t");

strcat(buf, temp); // 客户端消息内容

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

strcat(buf, "\n");

/\* 发给在线所有用户 \*/

for (i = 0; i < MAXMEM; i++)

{

if (connfd[i] != -1)

write(connfd[i], buf, strlen(buf));

}

}

memset(temp, 0, sizeof(temp));

}

}

}