

# README

---

## 项目结构和运行方法

---

项目结构如下:

```
1 | --src
2 |   |--client.c
3 |   |--server.c
4 |   |--customsocket.c
5 |   |--customfile.c
6 | --inc
7 |   |--customsocket.h
8 |   |--customfile.h
9 | --obj
10 | --makefile
11 | --README.pdf
```

- src 中保存着源代码
- inc 中保存着头文件
- obj 目录为空, 之后将保存生成的 .o 文件
- makefile 保存着编译的规则, 使用 make 命令 之后可以在它的同级目录下生成 client.out 和 server.out 文件
- README.pdf 则是对项目的描述

运行方法为, 输入 `make` 命令, 然后先运行 server.out, 再运行 client.out (可以同时运行多个 client)

## 大作业要求

---

在 Linux 平台上开发一个基于主机系统的多客户端多终端即时通讯/聊天室, 要求:

- 群聊
- 私聊
- 查看历史记录
- 发送文件
- 使用 Makefile

## 实现思路

---

由于群聊和私聊, 查看历史记录, 发送文件相对独立, 因此分为三部分介绍, 并搭配运行时的图片

### 群聊和私聊

群聊和私聊的实现是互相关联的, 举例而言, 有 client1, client2, client3 三个客户端, server 一个服务器  
实现群聊, 就是将 client<sub>i</sub> ( $i = 1, 2, 3$ ) 发送到 server 的消息转发给除 client<sub>i</sub> 之外的所有客户端

实现私聊, 就是将 client<sub>i</sub> ( $i = 1, 2, 3$ ) 发送到 server 的消息转发给 client<sub>j</sub> ( $j = 1, 2, 3$ )

### server 端的实现

为每一个连接上 server 的客户端分配了 Socket 描述符 cfd, 每一个 cfd 都对应一个聊天的线程, 代码如下:

```
1 pthread_create(&tid, NULL, chat, (void*)&cinfos[i]);
2 pthread_detach(tid); // 子线程分离,防止僵尸线程产生
```

将描述符 cfd, 客户端地址 addr 和给客户端分配的昵称封装为一个结构体 cli\_info, 并将 cli\_info 用一个数组统一管理, 代码如下:

```
1 struct cli_info {
2     struct sockaddr_in addr;
3     int cfd;
4     char name[BUFSIZ];
5 }
```

```
1 struct cli_info cinfos[CLI_CNT];
```

client 连接号 server 之后, 靠消息的格式判断是群聊还是私聊.

群聊没有要求, 私聊必须符合 `:u username msg` 的格式, 其中 username 是 server 分配给 clienti 的用户名, msg 是 clienti 发送过来的消息, 代码如下:

```
1 if (buf[0] == ':' && buf[2] == ' ') {
2     if (buf[1] == 'u') {
3         // 实现私聊的代码
4     }
5 } else {
6     // 实现群聊的代码
7 }
```

## client 的实现

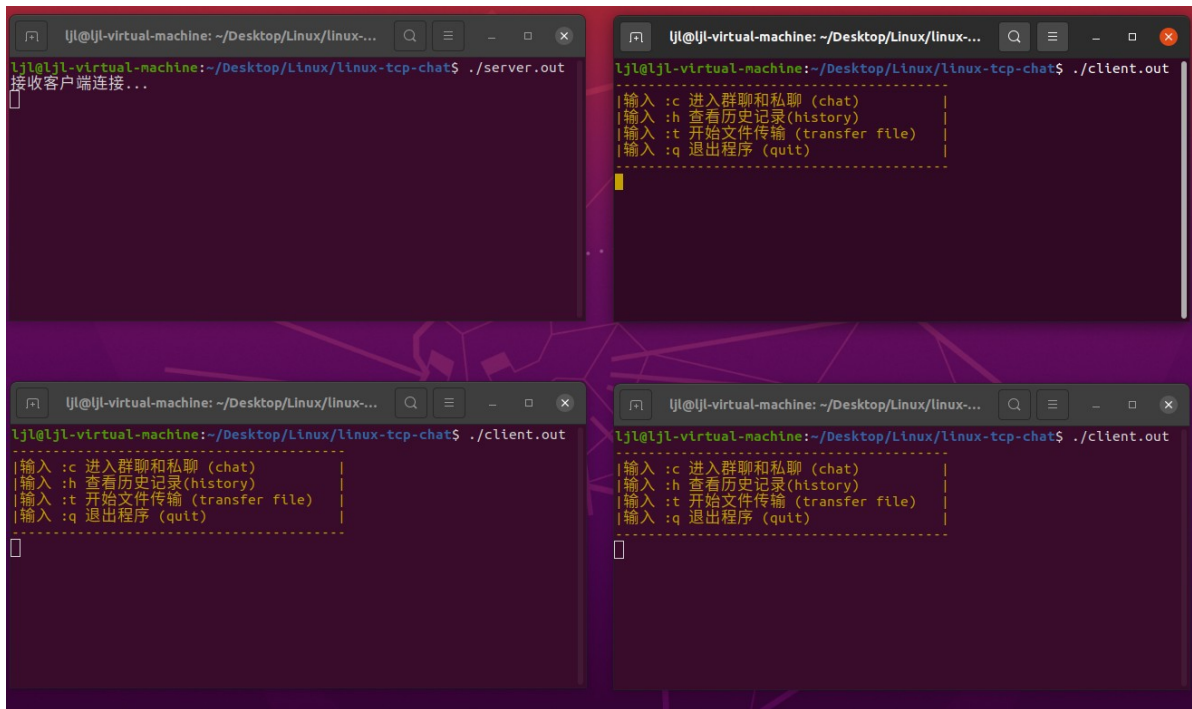
为了实现读入消息和接收消息并行, client 主线程读入消息, 子线程接收消息, 代码如下:

```
1 void chat() {
2     ...
3     pthread_create(&tid, NULL, chat_recv_msg, (void *)&sfd);
4     pthread_detach(tid);
5     ...
6 }
```

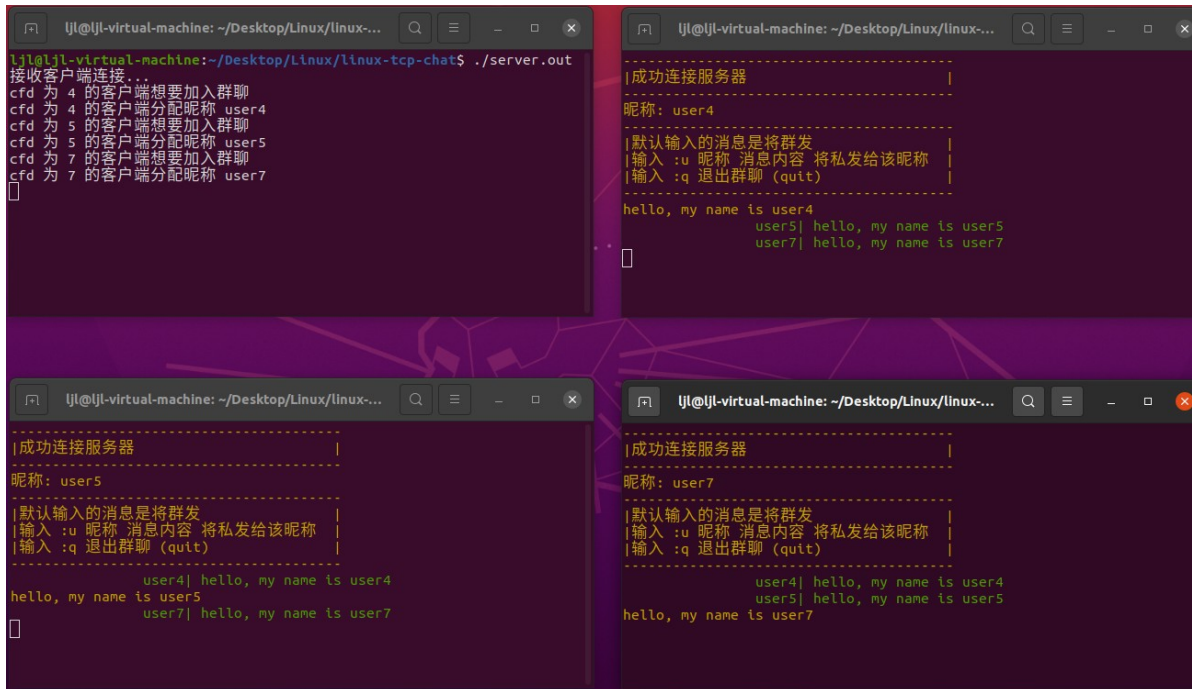
## 聊天效果图片

### 1. 选择进入聊天

运行 server.out 之后, 同时运行多个 client.out, 客户端输入 :c 进入聊天室



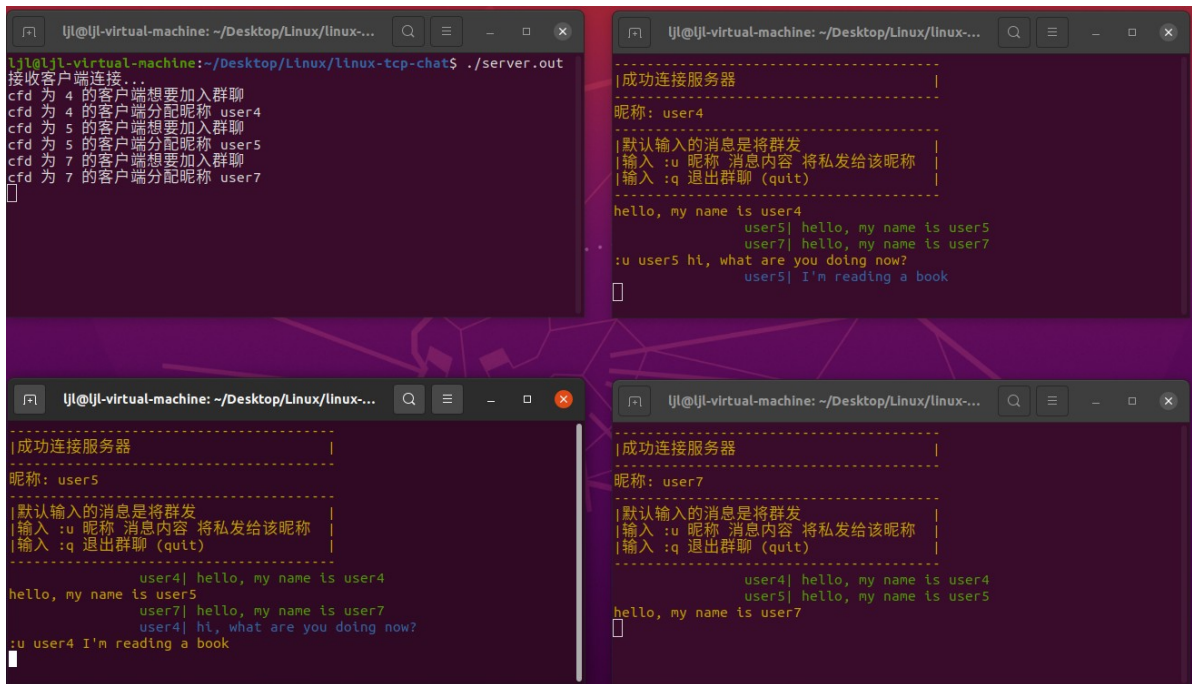
## 2. 发送群聊消息



如果是本人发送的信息, 颜色为黄色, 在客户端左侧显示.

如果是他人发送的信息, 颜色为绿色, 在客户端右侧显示

## 3. 发送私聊消息



与群聊消息不同, 他人发送的私聊消息, 颜色为蓝色

## 查看历史记录

将历史记录存储到文本 chat\_history.txt 中, 群聊的格式如下:

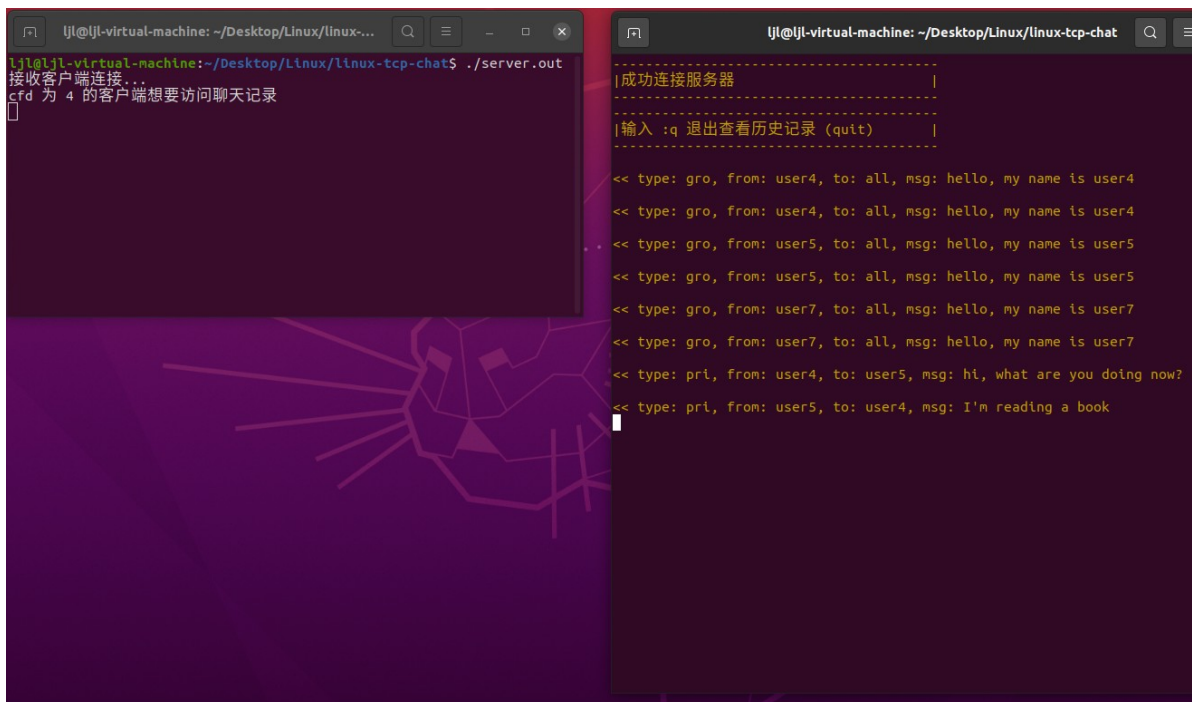
```
1 | << type: gro, from: user5, to: all, msg: hi
```

私聊的格式如下:

```
1 | << type: pri, from: user5, to: user4, msg: hello
```

聊天记录的存储是在群聊和私聊的时候写入到 chat\_history 中的, 至于查看聊天记录, 只是在 server 端读取文件后发送给 client

## 查看历史记录效果图片



# 发送文件

将 server 端作为一个中转站, client 可以选择上传文件还是下载文件, 其中上传文件的命令格式为 `:u filename`, 下载文件的命令格式为 `:d filename`, 下载到 client 的文件会多上 `down-` 的前缀, 而上传到 server 的文件会多上 `up-` 的前缀

## 1. 上传文件

```
ljl@ljl-virtual-machine: ~/Desktop/Linux/linux-...
ljl@ljl-virtual-machine:~/Desktop/Linux/linux-tcp-chat$ ./server.out
接收客户端连接...
cfd 为 4 的客户端想要上传文件
上传成功
```

```
ljl@ljl-virtual-machine: ~/Desktop/
|-----|
| 输入 :u 文件名 上传该文件到服务器 |
| 输入 :d 文件名 从服务器下载该文件 |
|-----|
:u chat_history.txt
上传成功
```

## 2. 下载文件

```
ljl@ljl-virtual-machine:~/Desktop/Linux/linux-...
ljl@ljl-virtual-machine:~/Desktop/Linux/linux-tcp-chat$ ./server.out
接收客户端连接...
cfd 为 4 的客户端想要上传文件
上传成功
cfd 为 5 的客户端想要下载文件
```

```
ljl@ljl-virtual-machine: ~/Desktop/Linux/li
|-----|
| 输入 :u 文件名 上传该文件到服务器 |
| 输入 :d 文件名 从服务器下载该文件 |
|-----|
:d chat_history.txt
下载成功
```

现在我们可以查看上传和下载的文件:

```
ljl@ljl-virtual-machine:~/Desktop/Linux/linux-...
ljl@ljl-virtual-machine:~/Desktop/Linux/linux-tcp-chat$ ./server.out
接收客户端连接...
cfd 为 4 的客户端想要上传文件
上传成功
cfd 为 5 的客户端想要下载文件
```

```
ljl@ljl-virtual-machine:~/Desktop/Linux/linux-tcp-chat$ cat up-chat_history.txt
<< type: gro, from: user4, to: all, msg: hello, my name is user4
<< type: gro, from: user4, to: all, msg: hello, my name is user4
<< type: gro, from: user5, to: all, msg: hello, my name is user5
<< type: gro, from: user5, to: all, msg: hello, my name is user5
<< type: gro, from: user7, to: all, msg: hello, my name is user7
<< type: gro, from: user7, to: all, msg: hello, my name is user7
<< type: pri, from: user4, to: user5, msg: hi, what are you doing now?
<< type: pri, from: user5, to: user4, msg: I'm reading a book
ljl@ljl-virtual-machine:~/Desktop/Linux/linux-tcp-chat$ cat down-chat_history.txt
<< type: gro, from: user4, to: all, msg: hello, my name is user4
<< type: gro, from: user4, to: all, msg: hello, my name is user4
<< type: gro, from: user5, to: all, msg: hello, my name is user5
<< type: gro, from: user5, to: all, msg: hello, my name is user5
<< type: gro, from: user7, to: all, msg: hello, my name is user7
<< type: gro, from: user7, to: all, msg: hello, my name is user7
<< type: pri, from: user4, to: user5, msg: hi, what are you doing now?
<< type: pri, from: user5, to: user4, msg: I'm reading a book
ljl@ljl-virtual-machine:~/Desktop/Linux/linux-tcp-chat$
```