README

项目结构和运行方法

项目结构如下:

```
|--src
|    |--client.c
|    |--server.c
|    |--customsocket.c
|    |--customfile.c
|--inc
|    |--customsocket.h
|    |--customfile.h
|--obj
|--makefile
|--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
- REAME.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 clienti (i = 1, 2, 3) 发送到 server 的消息转发给除 clienti 之外的所有客户端

实现私聊, 就是将 clienti (i = 1, 2, 3) 发送到 server 的消息转发给 clientj (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:

```
pthread_create(&tid, NULL, chat, (void*)&cinfos[i]);
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:

```
struct cli_info {
  struct sockaddr_in addr;
  int cfd;
  char name[BUFSIZ];
}
```

```
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:

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

client 的实现

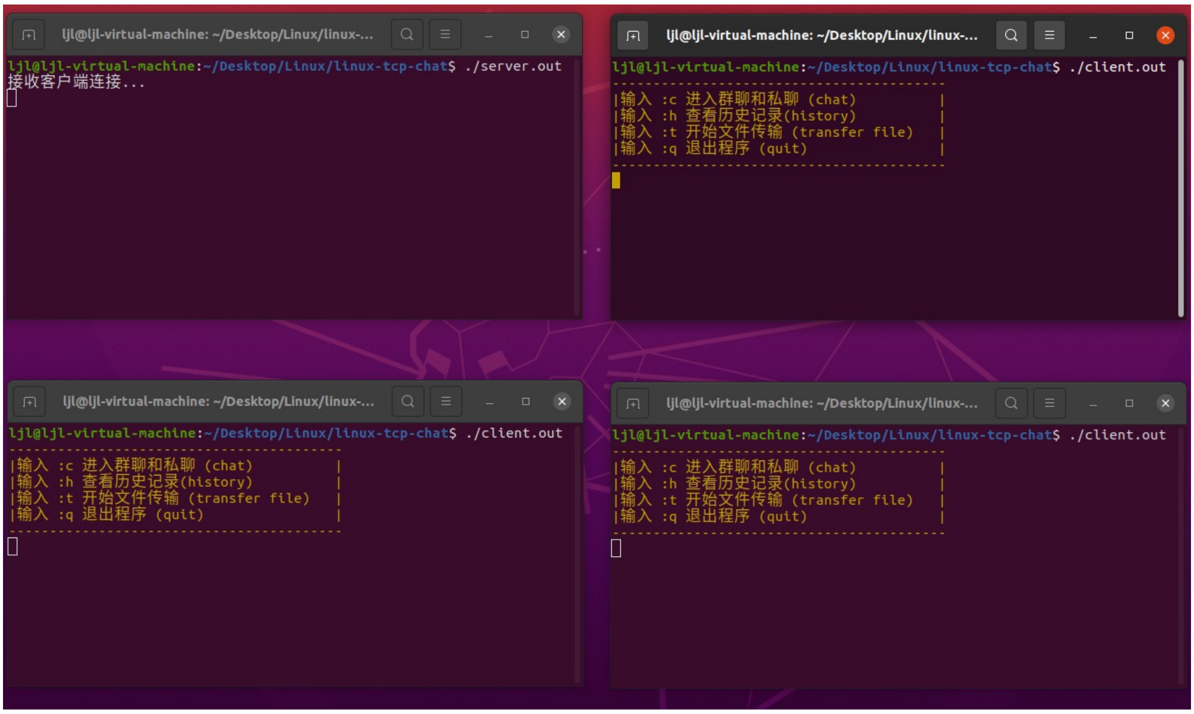
为了实现读入消息和接收消息并行, client 主线程读入消息, 子线程接收消息, 代码如下:

```
void chat() {
  ...
  pthread_create(&tid, NULL, chat_recv_msg, (void *)&sfd);
  pthread_detach(tid);
  ...
}
```

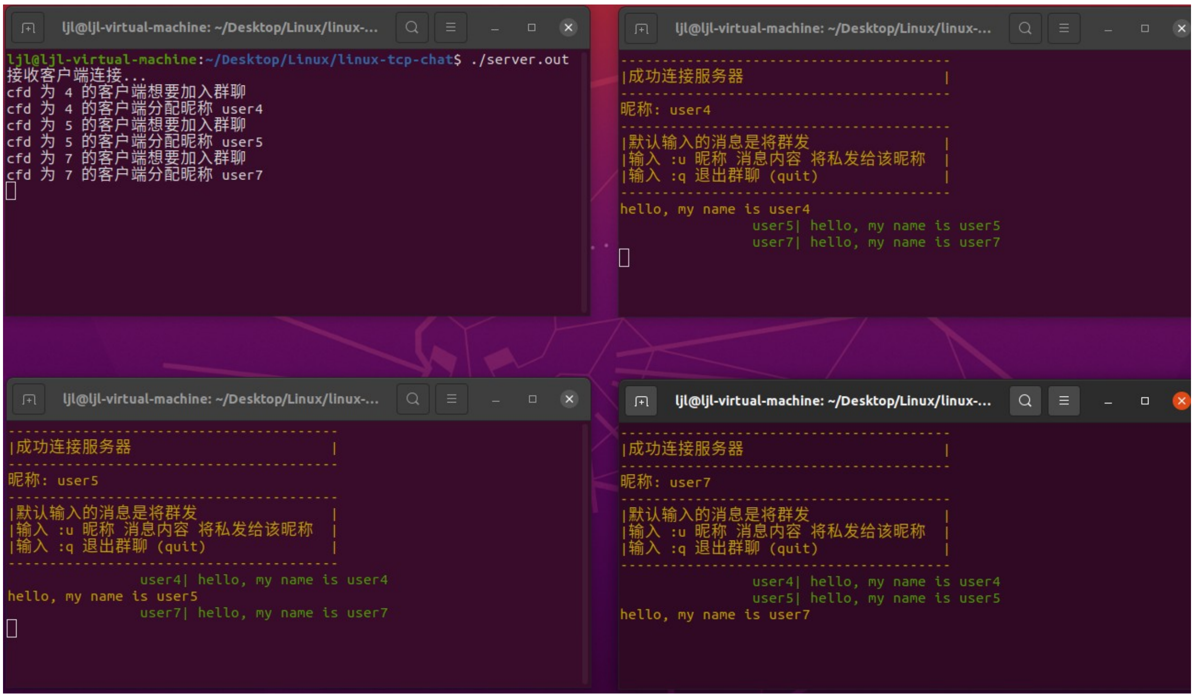
聊天效果图片

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



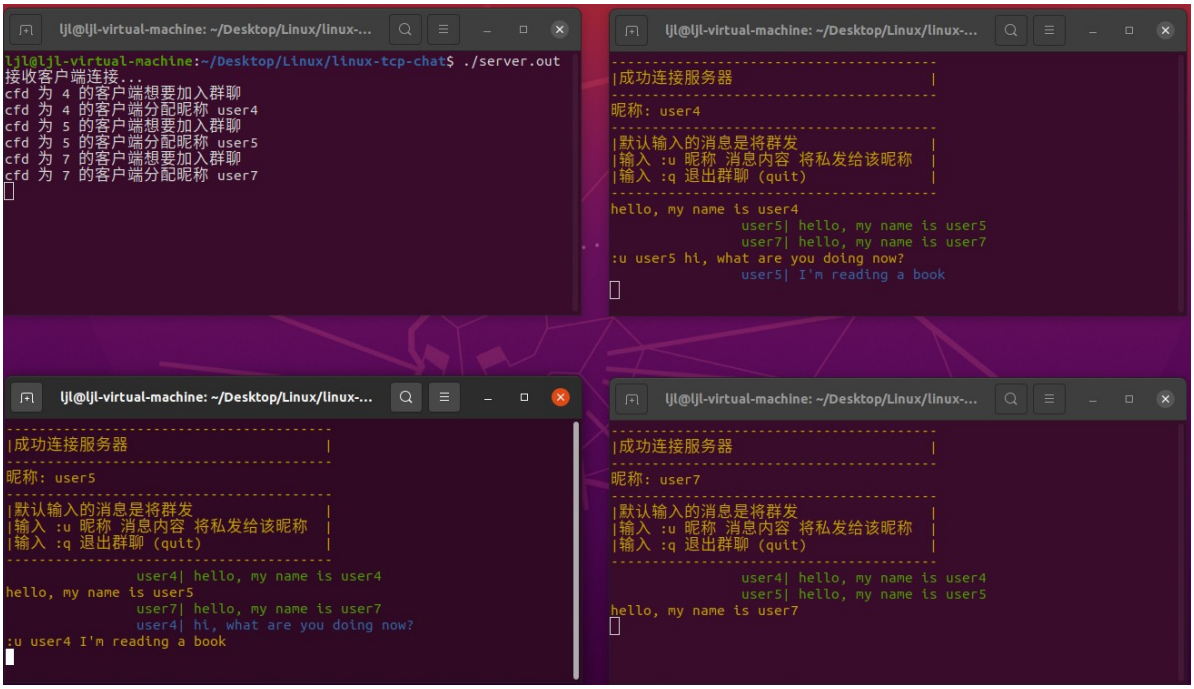
2. 发送群聊消息



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:

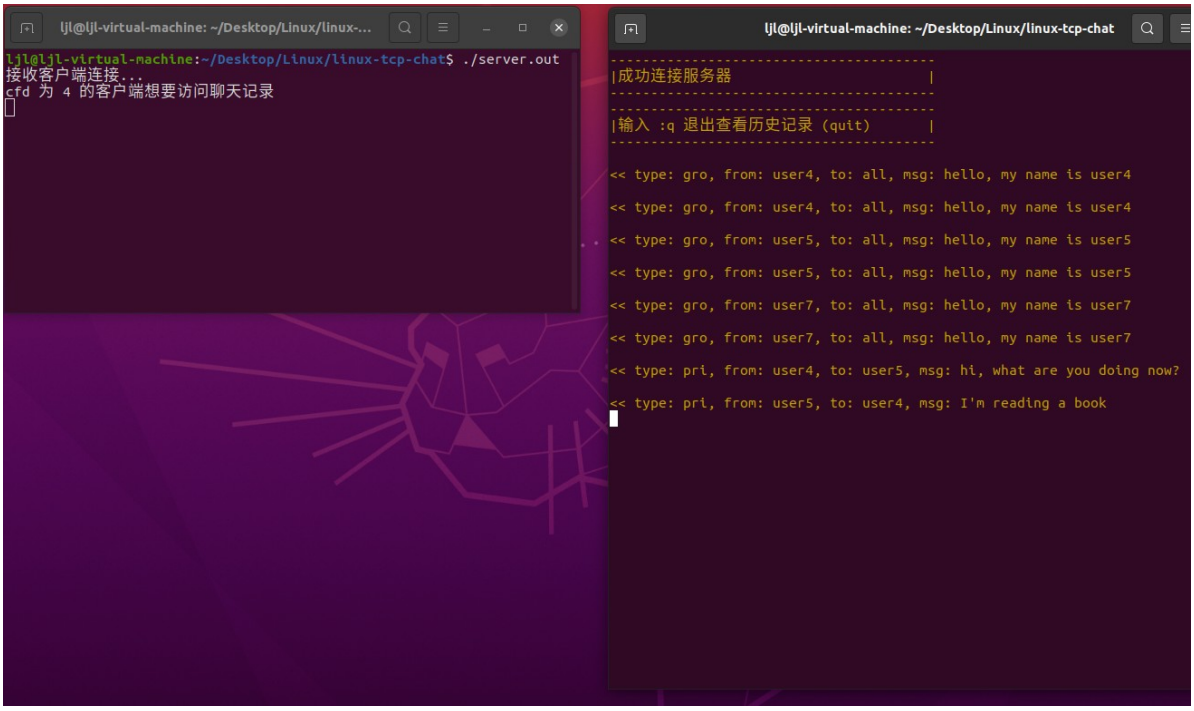
```
<< type: gro, from: user5, to: all, msg: hi
```

私聊的格式如下:

```
<< 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

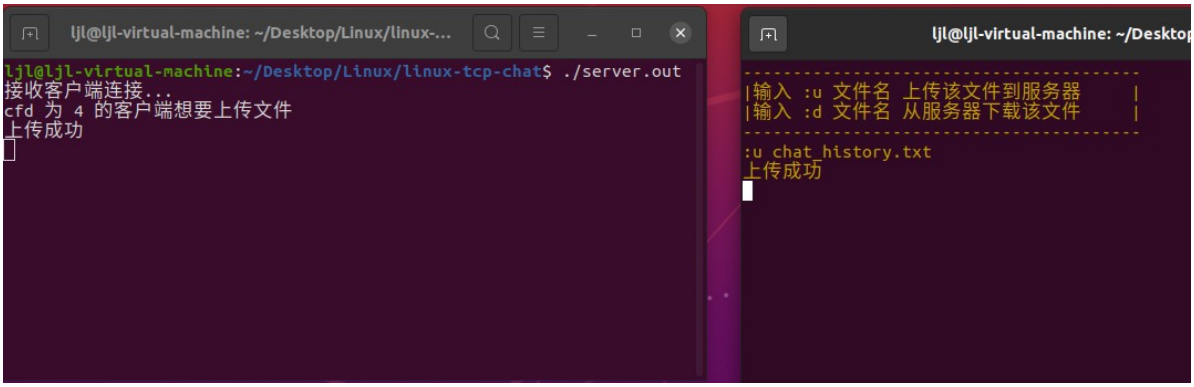
查看历史记录效果图片



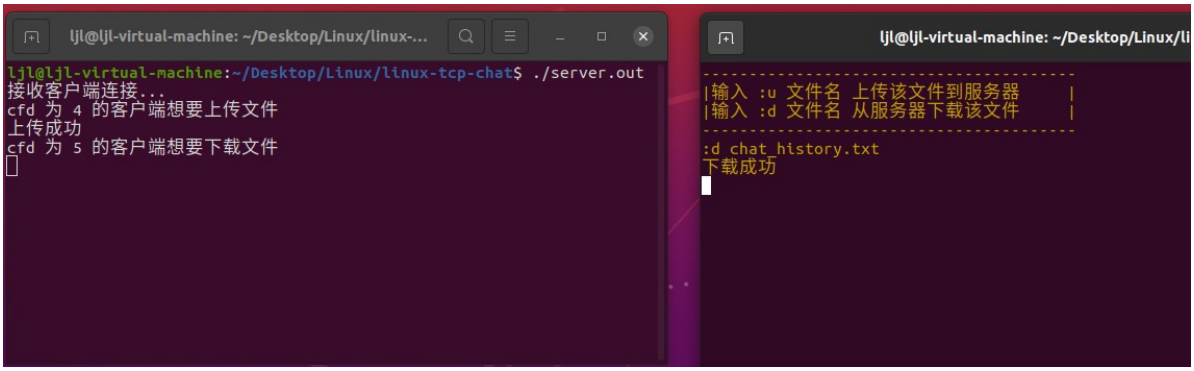
发送文件

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



2. 下载文件



现在我们可以查看上传和下载的文件: