

操作系统与 Linux 程序设计大作业

2020 春季

基于 TCP 协议的超级聊天室程序

第六小组

瞿昊 (组长)	181180099
李禄轶	181180058
宁展	181180090
潘禹龙	181180092
吴泳锟	181180141
周丽	181180208

Credited to Group 6 (Linux 小霸王)

目录

基于 TCP 协议的超级聊天室程序	1
0 目录.....	2
1 功能简介.....	3
2 人员分工.....	4
2.1 李禄轶.....	4
2.2 吴泳锬.....	4
2.3 宁展	4
2.4 潘禹龙.....	5
2.5 瞿昊	5
2.6 周丽	5
3 功能测试.....	6
3.1 登录与退出.....	6
3.2 群发与私发.....	8
3.3 查看聊天记录	11
3.4 表情包	12
3.5 发送与下载文件.....	13
3.6 图片的目标识别.....	17
3.7 读写博客功能	18
3.8 登录注册功能(Beta)	25
4 实验总结.....	27

CHAPTER 1

功能简介

本多功能聊天室软件基于 **TCP/IP** 协议，使用多线程机制控制不同客户端与服务器之间的联系，具备基本的聊天功能和部分现代服务器的高端功能。

基础功能部分，每一个用户在登录时都会拥有唯一的昵称，昵称重复时系统会要求更换昵称；用户可以在客户端群发消息且消息不会回显至自己的用户；用户随时可以通过 `\exit` 指令退出聊天室。

增强功能部分，每一个用户可以查看当前所有在线用户的列表，并单独指定另一用户进行私聊；可以通过 `\emoji` 指令发送特定表情；可以通过 `\send_file` 和 `\download` 指令进行文件的上传与下载；可以通过 `\message` 指令查看消息记录；还可以用一系列 `blog` 指令撰写属于自己的博客，并获得其他用户的点赞。另外，为了模仿当今服务器的人工智能服务，我们特别推出图片文件的目标识别，客户端可以从服务器端下载已上传照片的识别结果。

具体功能列表如下：

退出聊天室：输入命令 `\exit`

表情包：输入命令 `\emoji`

查看所有成员：输入命令 `\member`

私发消息：输入命令 `\send_to`

发送文件：输入命令 `\send_file`

下载文件：输入命令 `\download`

聊天记录：输入命令 `\message`

写博客：输入命令 `\write_blog`

读博客：输入命令 `\read_blog`

CHAPTER 2

人员分工

2.1 李禄轶

- 解决了多用户登录时的重名问题
- 编写\member, \send_to, \write_blog, \read_blog 模块
- 合作编写\emoji 模块
- 软件的鲁棒性测试

2.2 吴泳锴

- 编写\send_file（及其对应的 object detection），\download, \message 模块
- 合作编写\emoji 模块
- 软件的鲁棒性测试

2.3 宁展

- 协助解决多用户登录时的重名问题
- 软件的前期测试

2.4 潘禹龙

- 协助解决聊天室的登录问题
- 软件的后期测试

2.5 瞿昊

- 软件的后期测试

2.6 周丽

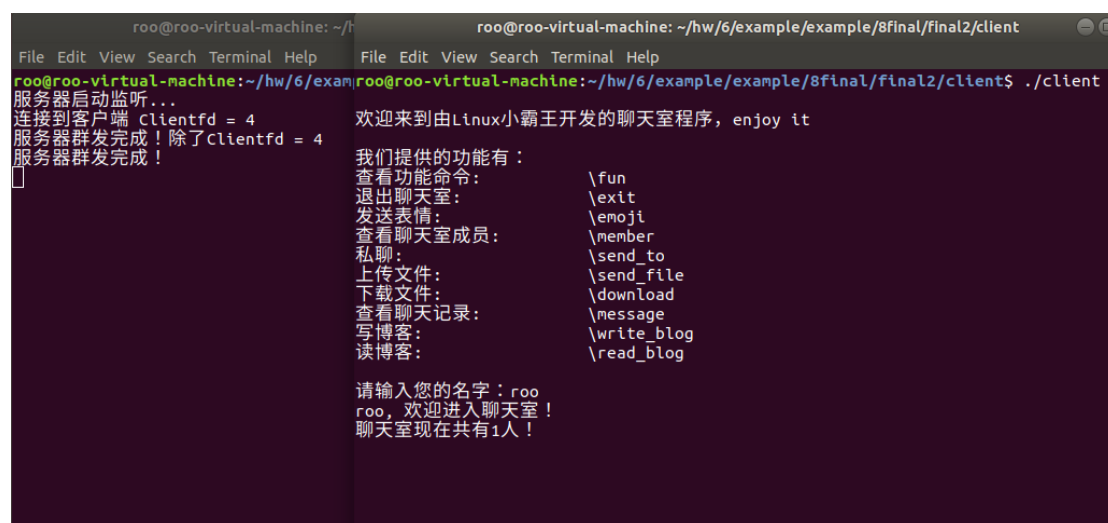
- 软件的后期测试

CHAPTER 3

功能测试

3.1 登录与退出

用户可以选择服务器的 IP 地址与端口号进行登录，但考虑到测试难度的问题，我们默认登录 IP 地址即为本机 IP，并使用同一台虚拟机进行相关测试。初始登录界面如图 1。



```
roo@roo-virtual-machine: ~/h
File Edit View Search Terminal Help
roo@roo-virtual-machine: ~/h/w/6/example/example/8final/final2/client$ ./client
服务器启动监听...
连接到客户端 Clientfd = 4
服务器群发完成！除了Clientfd = 4
服务器群发完成！

欢迎来到由Linux小霸王开发的聊天室程序，enjoy it

我们提供的功能有：
查看功能命令：      \fun
退出聊天室：        \exit
发送表情：          \emoji
查看聊天室成员：    \member
私聊：              \send_to
上传文件：          \send_file
下载文件：          \download
查看聊天记录：      \message
写博客：            \write_blog
读博客：            \read_blog

请输入您的名字：roo
roo，欢迎进入聊天室！
聊天室现在共有1人！
```

图 1 登录界面

若有另一用户使用同一昵称进行登录，系统会请求更换昵称，如图 2。

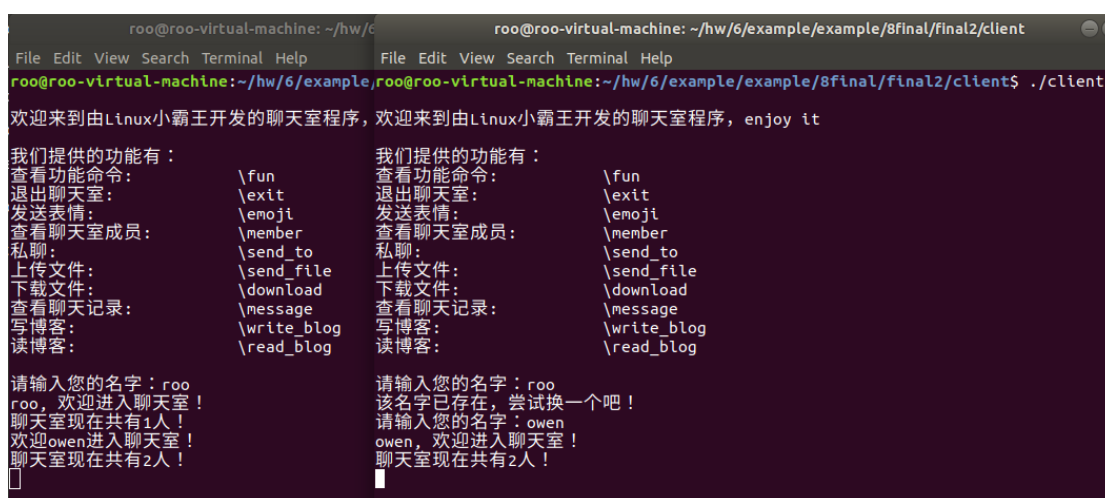
能检查重名的机制是先在服务器定义一个用户的结构体及其对应数组

```
1. typedef struct sockaddr SA;
2. typedef struct {
3.     int fd;
4.     char name[30];
5.     pthread_t thread;
6.     struct sockaddr_in add;
```

```
7.     socklen_t len;  
8. } user_t;  
9. user_t user[100];
```

只要有用户进行登录就对其注册的 **name** 进行判断

```
1. recv(user[i].fd, nameRecv, sizeof(nameRecv), 0);  
2. if (checkName(nameRecv) == 1){  
3.     send(user[i].fd, "change", 6, 0);  
4.     close(user[i].fd);  
5.     user[i].fd = -1;  
6. }
```



```
roo@roo-virtual-machine: ~/hw/c  
roo@roo-virtual-machine: ~/hw/6/example/example/8final/final2/client  
roo@roo-virtual-machine:~/hw/6/example,roo@roo-virtual-machine:~/hw/6/example/example/8final/final2/client$ ./client  
欢迎来到由Linux小霸王开发的聊天室程序, 欢迎来到由Linux小霸王开发的聊天室程序, enjoy it  
我们提供的功能有 :  
查看功能命令:      \fun  
退出聊天室:        \exit  
发送表情:          \emoji  
查看聊天室成员:    \member  
私聊:              \send_to  
上传文件:          \send_file  
下载文件:          \download  
查看聊天记录:      \message  
写博客:            \write_blog  
读博客:            \read_blog  
请输入您的名字: roo  
roo, 欢迎进入聊天室!  
聊天室现在共有1人!  
欢迎owen进入聊天室!  
聊天室现在共有2人!  
[ ]  
我们提供的功能有 :  
查看功能命令:      \fun  
退出聊天室:        \exit  
发送表情:          \emoji  
查看聊天室成员:    \member  
私聊:              \send_to  
上传文件:          \send_file  
下载文件:          \download  
查看聊天记录:      \message  
写博客:            \write_blog  
读博客:            \read_blog  
请输入您的名字: roo  
该名字已存在, 尝试换一个吧!  
请输入您的名字: owen  
owen, 欢迎进入聊天室!  
聊天室现在共有2人!  
[ ]
```

图 2 重名判断

进入聊天时候, 可以通过 `\member` 查看已登录的用户列表, 如图 3.



```
roo@roo-virtual-machine:~/hw/6/example/example/8final/final2/client$ ./client  
欢迎来到由Linux小霸王开发的聊天室程序, enjoy it  
我们提供的功能有 :  
查看功能命令:      \fun  
退出聊天室:        \exit  
发送表情:          \emoji  
查看聊天室成员:    \member  
私聊:              \send_to  
上传文件:          \send_file  
下载文件:          \download  
查看聊天记录:      \message  
写博客:            \write_blog  
读博客:            \read_blog  
请输入您的名字: owen  
owen, 欢迎进入聊天室!  
聊天室现在共有2人!  
\member  
现在的聊天室成员有 :  
1.roo  
2.owen (您)
```

图 3 成员列表

其实现原理是当服务器检测到来自客户端发送的\member 字段后，调用 listMember 函数遍历 user 的结构体数组。

```
1. void listMember(int fd) {  
2.     char listMsg[50];  
3.     int index = 1;  
4.     for (int i = 0; i < 100; ++i) {  
5.         if (user[i].fd == fd) {  
6.             memset(listMsg, 0, sizeof(listMsg));  
7.             sprintf(listMsg, "%d.%s (您) \n", index++, user[i].name);  
8.             send(fd, listMsg, strlen(listMsg), 0);  
9.         } else if (user[i].fd != -1) {  
10.            memset(listMsg, 0, sizeof(listMsg));  
11.            sprintf(listMsg, "%d.%s\n", index++, user[i].name);  
12.            send(fd, listMsg, strlen(listMsg), 0);  
13.        }  
14.    }  
15. }
```

使用\exit 可退出聊天室，如图 4，可见一个用户的退出并不影响其他用户的使用。

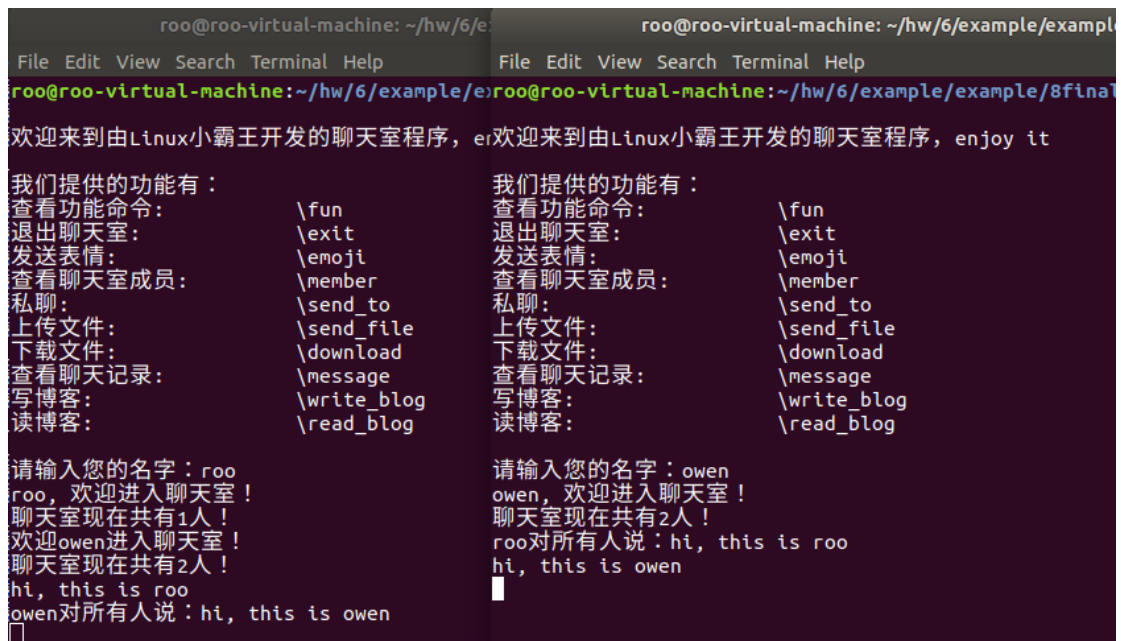


```
roo@roo-virtu roo@roo-virt roo@roo-virtual-machine: ~/hw/6/example/example/8final/final2/client  
File Edit View Search Tern File Edit View Search Ter File Edit View Search Terminal Help  
roo@roo-virtual-machine:roo@roo-virtual-machine:~/hw/6/example/example/8final/final2/client$ ./client  
服务器启动监听...  
连接到客户端 Clientfd = 欢迎来到由Linux小霸王开欢迎来到由Linux小霸王开发的聊天室程序, enjoy it  
服务器群发完成! 除了cli  
服务器群发完成! 我们提供的功能有:  
连接到客户端 Clientfd = 查看功能命令: \fun  
连接到客户端 Clientfd = 退出聊天室: \exit  
服务器群发完成! 除了cli发送表情: \emoji  
服务器群发完成! 查看聊天室成员: \member  
服务器群发完成! 除了cli私聊: \send_to  
服务器群发完成! 除了cli上传文件: \send_file  
服务器群发完成! 除了cli下载文件: \download  
检测到客户端退出 fd = 5 查看聊天记录: \message  
写博客: \write_blog  
读博客: \read_blog  
  
请输入您的名字: roo  
roo, 欢迎进入聊天室!  
聊天室现在共有1人!  
欢迎owen进入聊天室!  
聊天室现在共有2人!  
owen退出了聊天室  
聊天室现在共有1人!  
  
我们提供的功能有:  
查看功能命令: \fun  
退出聊天室: \exit  
发送表情: \emoji  
查看聊天室成员: \member  
私聊: \send_to  
上传文件: \send_file  
下载文件: \download  
查看聊天记录: \message  
写博客: \write_blog  
读博客: \read_blog  
  
请输入您的名字: roo  
该名字已存在, 尝试换一个吧!  
请输入您的名字: owen  
owen, 欢迎进入聊天室!  
聊天室现在共有2人!  
owen退出了聊天室  
聊天室现在共有1人!  
roo@roo-virtual-machine:~/hw/6/example/example/8final/final2/client$
```

图 4 退出界面

3.2 群发与私发

正常情况下，一个用户发出的消息所有用户都能看到，即群发消息。某用户发出的信息不会回显至该用户，但会广播到其他用户，如图 5。



```
roo@roo-virtual-machine: ~/hw/6/e/
File Edit View Search Terminal Help
roo@roo-virtual-machine:~/hw/6/example/example/8final
欢迎来到由Linux小霸王开发的聊天室程序，enjoy it

我们提供的功能有：
查看功能命令：      \fun
退出聊天室：        \exit
发送表情：          \emoji
查看聊天室成员：    \member
私聊：              \send_to
上传文件：          \send_file
下载文件：          \download
查看聊天记录：      \message
写博客：            \write_blog
读博客：            \read_blog

请输入您的名字：roo
roo，欢迎进入聊天室！
聊天室现在共有1人！
欢迎owen进入聊天室！
聊天室现在共有2人！
hi, this is roo
owen对所有人说：hi, this is owen

roo@roo-virtual-machine: ~/hw/6/example/example/8final
File Edit View Search Terminal Help
欢迎来到由Linux小霸王开发的聊天室程序，enjoy it

我们提供的功能有：
查看功能命令：      \fun
退出聊天室：        \exit
发送表情：          \emoji
查看聊天室成员：    \member
私聊：              \send_to
上传文件：          \send_file
下载文件：          \download
查看聊天记录：      \message
写博客：            \write_blog
读博客：            \read_blog

请输入您的名字：owen
owen，欢迎进入聊天室！
聊天室现在共有2人！
roo对所有人说：hi, this is roo
hi, this is owen
```

图 5 群发消息

其实现原理其实就是将某用户的 id 传入广播消息的函数，使广播时当循环变量与 id 相等时，跳过消息的发送。

```
1. /*client 发出的消息不会回显给自己*/
2. void SendMsgToAll(char* msg, int exception){
3.     for(int i = 0; i < 100; ++i)
4.         if(user[i].fd != exception && user[i].fd != -1){
5.             send (user[i].fd, msg, strlen(msg), 0);
6.         }
7.     if (exception == -1){
8.         printf("服务器群发完成！\n");
9.     }
10.    } else {
11.        printf("服务器群发完成！除了 Clientfd = %d\n", exception);
12.    }
13. }
```

除此之外，用户还可以输入 `\send_to` 命令，选择用户进行私聊，如图 6，目前聊天室中有三名用户：roo、owen 和 frank。roo 此时选择了 frank 进行私聊，frank 收到了 roo 发送的消息，而 owen 没有收到。

```
请输入您的名字：roo      请输入您的名字：owen      请输入您的名字：frank
roo, 欢迎进入聊天室！    owen, 欢迎进入聊天室！    frank, 欢迎进入聊天室！
聊天室现在共有1人！      聊天室现在共有2人！      聊天室现在共有3人！
欢迎owen进入聊天室！     欢迎frank进入聊天室！    roo对你说：owen is so stupid!
聊天室现在共有2人！      聊天室现在共有3人！      
欢迎frank进入聊天室！    聊天室现在共有3人！      
聊天室现在共有3人！      
\send to
请选择您的私聊对象（输入
1.owen
2.roo（您）
3.frank
3
输入您要发送的内容
owen is so stupid!

```

图 6 私发消息

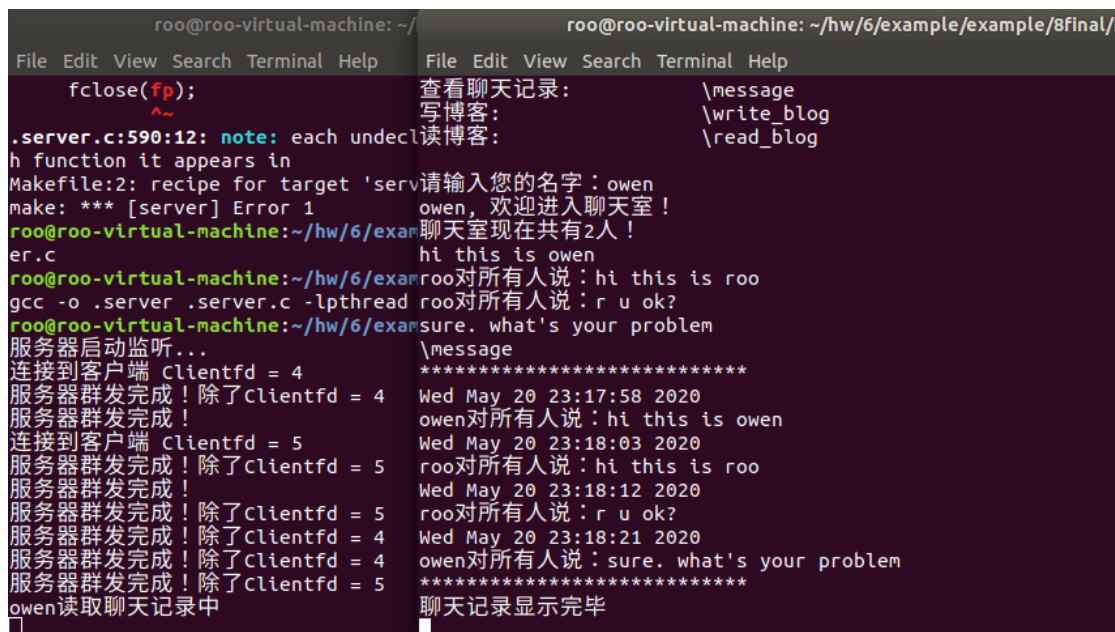
私聊实现的原理是首先调用之前`member`功能中写好的`listMember`函数，将当前的用户列表传送给调用此功能的用户，并接受用户输入的选项，接着遍历用户数组结构体中找到私聊对象，向其发送私聊信息。注意：如果输入的序号不在当前用户范围内，服务器会提示当前用户不存在。

代码如下：

```
1. void send_to(char *name, int fd){
2.     char choice[5], Msg[100], SendtoMsg[150];
3.     int choiceNumber, i, index = 0;
4.     memset(choice, 0, sizeof(choice));
5.     memset(Msg, 0, sizeof(Msg));
6.     memset(SendtoMsg, 0, sizeof(SendtoMsg));
7.     send(fd, "请选择您的私聊对象（输入序号）\n", strlen("请选择您的私聊对象（输入序号）\n"), 0);
8.     listMember(fd);
9.     recv(fd, choice, sizeof(choice), 0);
10.    choice[strlen(choice) - 1] = '\0';
11.    choiceNumber = atoi(choice);
12.    send(fd, "输入您要发送的内容\n", strlen("输入您要发送的内容\n"), 0);
13.    recv(fd, Msg, sizeof(Msg), 0);
14.    for (i = 0; i < 100; ++i) {
15.        if (user[i].fd != -1) index++;
16.        if (index == choiceNumber) break;
17.    }
18.    sprintf(SendtoMsg, "%s 对你说: %s", name, Msg);
19.    send(user[i].fd, SendtoMsg, strlen(SendtoMsg), 0);
20. }
```

3.3 查看聊天记录

查看聊天记录需要用户输入`\message` 命令, 如图 7. 可以看到聊天记录按时间顺序呈现用户间的对话。



```
roo@roo-virtual-machine: ~/
File Edit View Search Terminal Help
fclose(fp);
^~
.server.c:590:12: note: each undeclared identifier is declared here
h function it appears in
Makefile:2: recipe for target 'server' failed:
make: *** [server] Error 1
roo@roo-virtual-machine: ~/hw/6/example/8final/
er.c
roo@roo-virtual-machine: ~/hw/6/example/8final/
gcc -o .server .server.c -lpthread
roo@roo-virtual-machine: ~/hw/6/example/8final/
服务器启动监听...
连接到客户端 Clientfd = 4
服务器群发完成! 除了Clientfd = 4
服务器群发完成!
连接到客户端 Clientfd = 5
服务器群发完成! 除了Clientfd = 5
服务器群发完成!
服务器群发完成! 除了Clientfd = 5
服务器群发完成! 除了Clientfd = 4
服务器群发完成! 除了Clientfd = 4
服务器群发完成! 除了Clientfd = 5
owen读取聊天记录中

roo@roo-virtual-machine: ~/hw/6/example/8final/
File Edit View Search Terminal Help
查看聊天记录: \message
写博客: \write_blog
读博客: \read_blog
请输入您的名字: owen
owen, 欢迎进入聊天室!
聊天室现在共有2人!
hi this is owen
roo对所有人说: hi this is roo
roo对所有人说: r u ok?
sure. what's your problem
\message
*****
Wed May 20 23:17:58 2020
owen对所有人说: hi this is owen
Wed May 20 23:18:03 2020
roo对所有人说: hi this is roo
Wed May 20 23:18:12 2020
roo对所有人说: r u ok?
Wed May 20 23:18:21 2020
owen对所有人说: sure. what's your problem
*****
聊天记录显示完毕
```

图 7 聊天记录

聊天记录的具体实现是通过创建`.message` 文件记录群发消息, 且每群发一次消息`.message` 打开并关闭一次, 防止时间顺序出现错乱。在聊天记录的读取过程中, 调用的`Message` 函数使用 `fread` 函数并通过循环发送数据, 使聊天记录从服务器输出至客户端。

```
1. //Message 函数主要部分
2. if ((fp = fopen(".message", "r")) == NULL) { //打开文件
3.     send(fd, "读取聊天记录失败\n", strlen("读取聊天记录失败\n"), 0);
4.     perror("读取聊天记录失败\n");
5.     return;
6. }
7. printf("%s 读取聊天记录中\n", name);
8. send(fd, margin, strlen(margin), 0);
9. while ((read_len = fread(buf, sizeof(char), 100, fp)) > 0 ) {
10.     send_len = send(fd, buf, read_len, 0);
11.     if ( send_len < 0 ) {
12.         send(fd, "发送聊天记录失败\n", strlen("发送聊天记录失败\n"), 0);
13.         perror("发送聊天记录失败\n");
14.         return;
15.     }
16.     bzero(buf, 100);
```

```
17. }
```

3.4 表情包

表情包需要用户输入\emoji 命令，并输入表情包选项，如图 8。

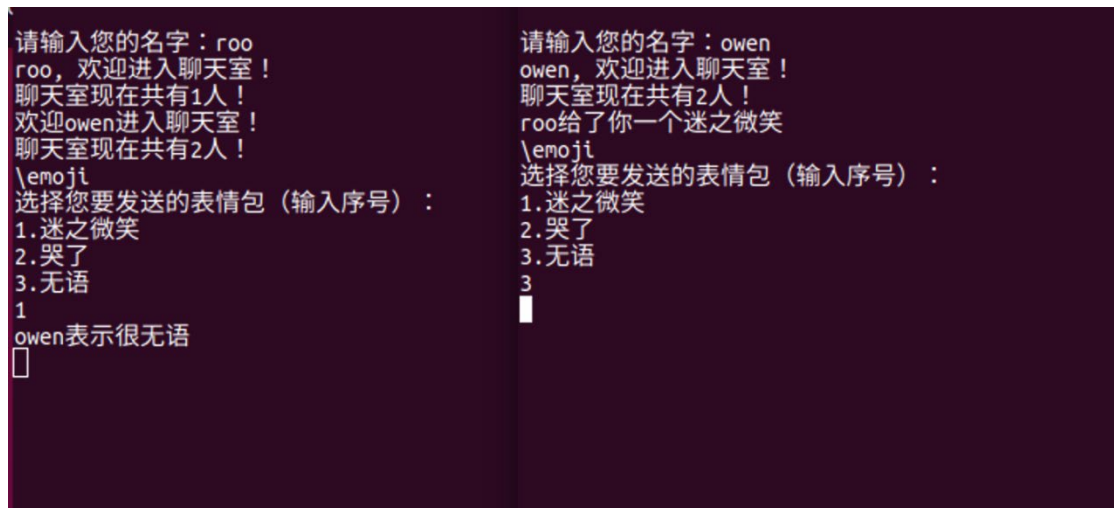


图 8 表情包

其原理是在服务端创建了一个表情包数组，包括了当前所有的表情包（如有需要可以继续扩充）。服务端先将表情包信息传给客户端，待客户端输入选项之后，将对应的信息群发。因为是群发，所以表情包信息也会被记录到聊天记录中。（暂未开发私发表情包，后续会继续进行优化）。注意：如果输入的选项不在当前表情包选项范围内，会提示重新输入\emoji 命令发送表情包。代码部分如下：

```
1. void emoji(char* name, int fd)
2. {
3.     char* emo_value[3] = {"给了你一个迷之微笑", "哭了", "表示很无语"};
4.     char choiceList[100];
5.     char choice[3];
6.     char emojiMsg[100];
7.     char errorMsg[50];
8.     int choiceNumber;
9.     memset(choiceList, 0, sizeof(choiceList));
10.    memset(choice, 0, sizeof(choice));
11.    memset(emojiMsg, 0, sizeof(emojiMsg));
12.    sprintf(choiceList, "选择您要发送的表情包（输入序号）：\n1.迷之微笑\n2.哭\n3.无语\n");
13.    send(fd, choiceList, strlen(choiceList), 0);
```

```
14.     recv(fd, choice, sizeof(choice), 0);
15.     choice[strlen(choice) - 1] = '\0';
16.     choiceNumber = atoi(choice);
17.     sprintf(emojiMsg, "%s%s\n", name ,emo_value[choiceNumber - 1]);
18.     SendMsgToAll(emojiMsg, fd);
19. }
```

3.5 发送与下载文件

发送文件需要用户输入\send_file 命令，并根据提示输入待上传文件，并等待上传成功信息的返回。待发送文件如图 9. 发送过程如图 10. 服务器接受的上传文件如图 11.

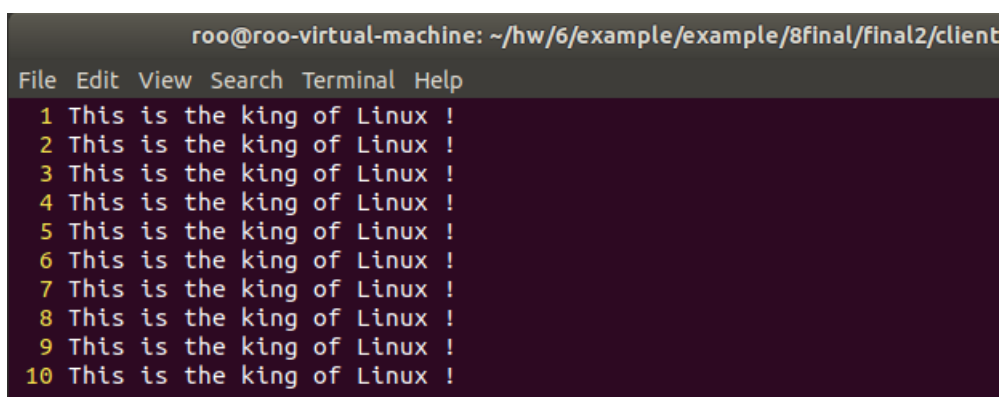


图 9 待发送文件

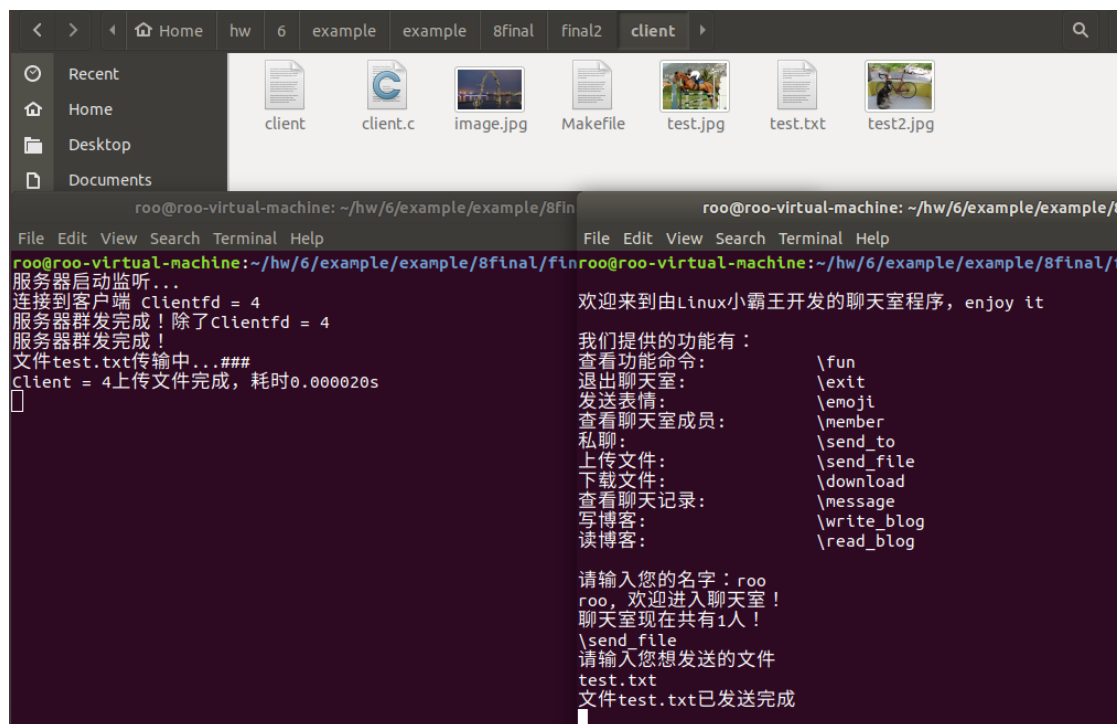


图 10 发送过程

```
roo@roo-virtual-machine: ~/hw/6/example/example/8final/final2/server
File Edit View Search Terminal Help
1 This is the king of Linux !
2 This is the king of Linux !
3 This is the king of Linux !
4 This is the king of Linux !
5 This is the king of Linux !
6 This is the king of Linux !
7 This is the king of Linux !
8 This is the king of Linux !
9 This is the king of Linux !
10 This is the king of Linux !
```

图 11 已上传文件

可以看到，`\send_file` 命令使文件从客户端上传至服务器，上传过程中并没有出现信息的丢失。具体实现原理如下：客户端在接收到服务器发送的提示信息后，输入文件名并发送至服务器。需要注意的是，由于输入结束敲击了 `\n` 回车键，从 `stdin` 读入的文件名末尾有多余的 `\n`，需要对其进行清除。

```
1. fgets(file, sizeof(file), stdin);
2. file[strlen(file) - 1] = '\0';
```

之后客户端调用 `fopen` 函数打开待发送文件，通过循环读取的方式发送文件内容。注意，在传输完文件后，必须再发送结束提示字符串“OK！”至服务器，否则服务器将无法判别何时算是传输完毕，而将接下来客户端发送的群发消息误以为是文件内容的一部分。客户端的主体代码如下：

```
1. start = clock();
2. while (recv_len = recv(fd, buf, 100, 0)) {
3.     if(recv_len < 0) {
4.         printf("文件接收失败\n");
5.         break;
6.     }
7.     printf("#");
8.     int i = 0;
9.     for(i = 0; i < recv_len; i++){
10.        if(strcmp(&buf[i], "OK!") == 0) //暴力检查传输结束标志
11.            break;
12.    }
13.    write_len = fwrite(buf, sizeof(char), (i < recv_len)? i : recv_len, fp);
14.    if(i < recv_len){
15.        end = clock();
16.        printf("\nClient = %d 上传文件完成，耗时%f\n", fd, (double)(end - start)/CLOCKS_PER_SEC);
```

```
17.         fclose(fp);  
18.     }  
19. }
```

下载调用指令\download，过程原理类似。只是由于下载过程处于客户端的接收线程，因此必须再接收线程调用下载的函数，这就需要服务器发出下载提示符“DL”使客户端的接收线程进入 Download 函数。此外，为了提供更好的交互效果，我们设计将服务器端可供下载的文件通过 ls 形式呈现并发送至客户端。值得注意的是，为了防止客户端恶意下载服务器端相关配置文件，我们将用户无下载权限的文件全设为隐藏文件。

```
1. send(fd, "请输入您想下载的文件\n", strlen("请输入您想下载的文件\n"), 0);  
2. system("ls > .file_list.txt");  
3. fp = fopen(".file_list.txt", "r");  
4. fread(file_list, sizeof(char), 100, fp);  
5. send(fd, file_list, strlen(file_list), 0);  
6. fclose(fp);  
7. system("rm -f .file_list.txt");
```

下载过程如图 12 显示，成功下载的文件如图 13。

可见，下载的文件也是保真的。实际上，下载大文件也不是问题，大小达 25M 的图片文件如图 14，也能迅速下载，过程如图 15。大文件的下载速度达到惊人的 $25\text{M} \div 0.2\text{s} \approx 125\text{Mbps}$ ，这么快的下载速度完全可以取代下载速度受限的百度云网盘，可以说由 Linux 小霸王开发的超级聊天室前景还是巨大的。

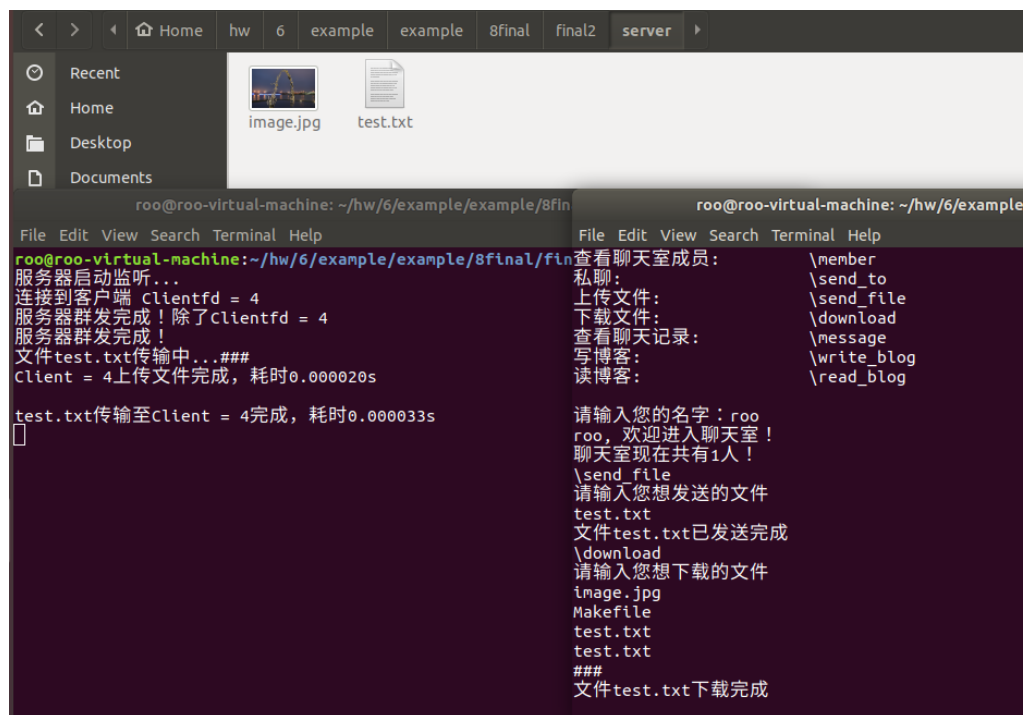


图 12 下载过程


```
roo@roo-virtual-machine: ~/hw/6/example/example/8final/final2/client
File Edit View Search Terminal Help
1 This is the king of Linux !
2 This is the king of Linux !
3 This is the king of Linux !
4 This is the king of Linux !
5 This is the king of Linux !
6 This is the king of Linux !
7 This is the king of Linux !
8 This is the king of Linux !
9 This is the king of Linux !
10 This is the king of Linux !
```

图 13 下载文件

```
hw 6 example example 8final final3(5.23) server
image.jpg test.txt
roo@roo-virtual-machine: ~/hw/6/example/example/8final/final3(5.23)/server
File Edit View Search Terminal Help
roo@roo-virtual-machine:~/hw/6/example/example/8final/final3(5.23)/server$ ll
total 25232
drwxrwxrwx 3 roo roo 4096 May 23 10:26 ./
drwxr-xr-x 4 roo roo 4096 May 20 16:44 ../
-rw-r--r-- 1 roo roo 25744771 May 21 00:58 image.jpg
-rwxrwx-rw- 1 roo roo 103 May 20 16:20 .Makefile*
-rw-r--r-- 1 roo roo 56 May 23 10:02 .message
-rw-r--r-- 1 roo roo 142 May 20 22:53 .message1
-rwxr-xr-x 1 roo roo 31128 May 23 10:05 .server*
-rw-r--r-- 1 roo roo 25720 May 23 10:05 .server.c
-rw-r--r-- 1 roo roo 280 May 23 10:02 test.txt
drwxrwxrwx 4 roo roo 4096 May 21 09:56 ../
```

图 14 待下载图片

```
roo@roo-virtual-machine: ~/hw/6/example/e roo@roo-virtual-machine: ~/hw/6/example/example/8final/final3(5.23)/client
File Edit View Search Terminal Help File Edit View Search Terminal Help
roo@roo-virtual-machine:~/hw/6/example/exampl#####
roo@roo-virtual-machine:~/hw/6/example/exampl#####
server#####
输入端口号 (按回车为默认10222号端口)#####
服务器后动监听...#####
连接到客户端 Clientfd = 4#####
服务器群发完成! 除了Clientfd = 4#####
服务器群发完成!#####
image.jpg传输至client = 4完成,耗时0.173119s#####
服务器群发完成! 除了Clientfd = 4#####
检测到客户端退出 fd = 4#####
#####
#####
文件image.jpg下载完成
\exit
roo@roo-virtual-machine:~/hw/6/example/example/8final/final3(5.23)/cli
age.jpg
-rw-r--r-- 1 roo roo 25744771 May 23 10:33 image.jpg
```

图 15 已下载图片

3.6 图片的目标识别

为模拟现代高性能服务器的人工智能服务，我们特意引入 yolo-v3 目标识别算法至服务器端，详细配置方法参考官方网址 <https://pjreddie.com/darknet/yolo/>。我们将 yolo 的权重及相关配置文件存放在服务器端的.yolo 文件夹，如图 16。

当服务器检测到上传的文件是图片时，会向客户端询问是否进行目标识别。若用户输入 yes, 服务器会将识别结果返回, 用户可以通过download 指令下载识别结果 predictions.jpg, 具体操作如图 17。

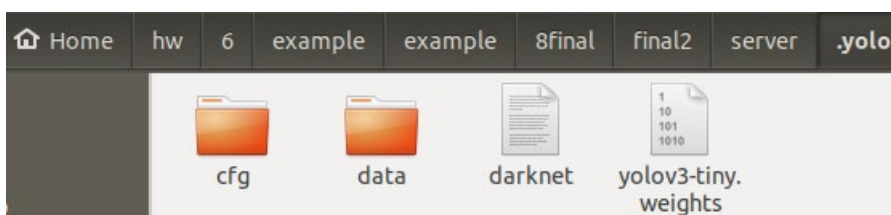


图 16 yolo 算法配置文件

```
roo@roo-virtual-machine: ~/hw/6/example/example/8final/final2/server
File Edit View Search Terminal Help
10 conv 512 3 x 3 / 1 13 x 13 x 256 -> 13 x 13 x 512 聊天室现在共有1人！
OPs \send_file
11 max 2 x 2 / 1 13 x 13 x 512 -> 13 x 13 x 512 请输入您想发送的文件
12 conv 1024 3 x 3 / 1 13 x 13 x 512 -> 13 x 13 x 1024 test2.jpg
OPs 文件test2.jpg已发送完成
13 conv 256 1 x 1 / 1 13 x 13 x 1024 -> 13 x 13 x 256 是否要进行图片的目标识别？(yes/no)
OPs yes
14 conv 512 3 x 3 / 1 13 x 13 x 256 -> 13 x 13 x 512 目标识别已完成，请下载predictions.jpg获取识别结果
OPs \download
15 conv 255 1 x 1 / 1 13 x 13 x 512 -> 13 x 13 x 255 请输入您想下载的文件
OPs image.jpg
16 yolo predictions.jpg
17 route 13 test2.jpg
18 conv 128 1 x 1 / 1 13 x 13 x 256 -> 13 x 13 x 128 test.txt
OPs predictions.jpg
19 upsample 2x 13 x 13 x 128 -> 26 x 26 x 128 #####
20 route 19 8 #####
21 conv 256 3 x 3 / 1 26 x 26 x 384 -> 26 x 26 x 256 #####
OPs #####
22 conv 255 1 x 1 / 1 26 x 26 x 256 -> 26 x 26 x 255 #####
OPs #####
23 yolo #####
Loading weights from yolov3-tiny.weights...Done! #####
test2.jpg: Predicted in 0.811493 seconds. #####
dog: 57% #####
car: 52% #####
truck: 56% #####
car: 62% #####
bicycle: 59% #####
predictions.jpg传输至client = 4完成，耗时0.001737s 文件predictions.jpg下载完成
```

图 17 目标识别过程

我们可以打开客户端上传的 test2.jpg 和下载的 predictions.jpg 做个对比，如图 18.可以看到，目标识别结果令人满意。

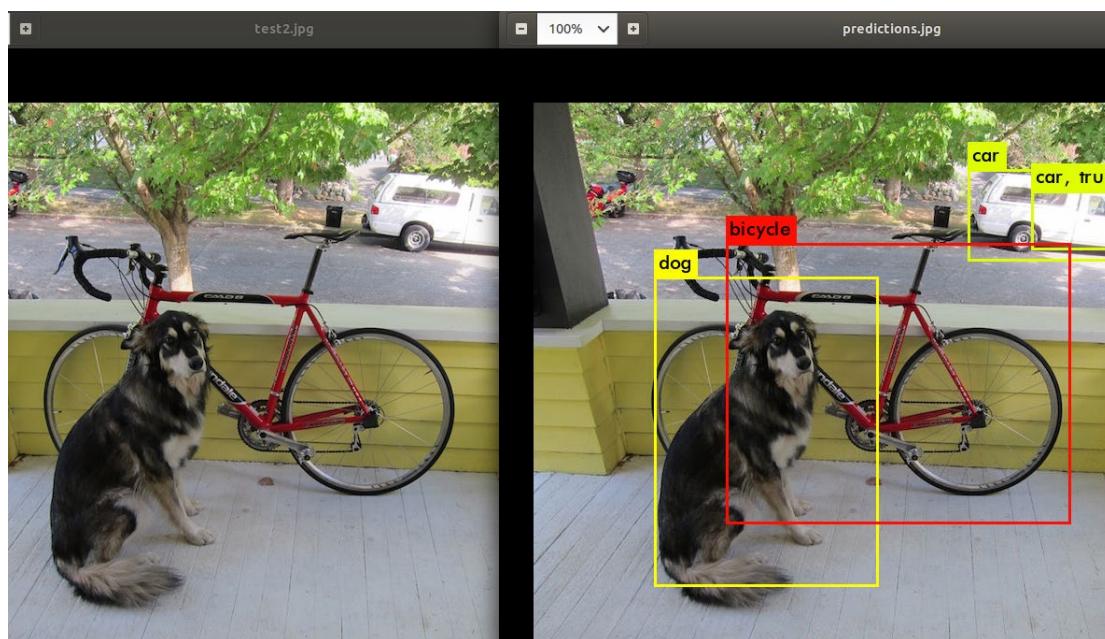


图 18 目标识别结果

3.7 读写博客功能

读写博客功能分为`\read_blog`和`\write_blog`命令，分别是读博客和写博客。

首先介绍`\write_blog`——写博客功能。用户需要输入`\write_blog`进入该功能。在写博客之前，首先需要确定当前博客的标题。除此之外，我们还增加了访问权限设置，分为所有人可见和仅自己可见。图 19 展示了所有人可见时的效果，写完博客之后服务端会告诉当前聊天室所有用户有人上传了博客。而图 20 中选择了仅自己可见，此时服务端不会告知其他用户有人上传了博客。



图 19 写一篇所有人可见的博客

```
\write_blog
欢迎来到小霸王博客, 请给您的博客起一个标题吧:
小秘密
谁可以看?(输入序号)
1.所有人可见
2.仅自己可见
2
请输入《小秘密》的内容:(输入\finish结束)
希望彭成磊老师不会发现我们程序的bug。
hhhhhhh
\finish
您上传了一篇博客《小秘密》, 请用"\read_blog"命令:

请输入您的名字:owen
owen, 欢迎进入聊天室!
聊天室现在共有2人!
roo上传了一篇博客《今日更新》, 请用"\read_blog"命令查看详细内容。
```

图 20 写一篇仅自己可见的博客

写博客功能的基本原理, 就是将用户输入的内容记录到一个 **txt** 文本文件中。所有博客都存放在服务端当前文件夹下的 **.blog** 隐藏文件夹中。如果某位用户写了一篇博客, 服务端会在 **blog** 文件夹中创建与当前用户名同名的文件夹, 并将 **txt** 文本文件存放在当前文件夹下。文件结构如图 21 所示。

```
owen@ubuntu: ~/Desktop/final/server/.blo
File Edit View Search Terminal Help
owen@ubuntu:~/Desktop/final/server$ ls -a
.  ..  .blog  .DS_Store  .Makefile  .server  .server.c  .yolo
owen@ubuntu:~/Desktop/final/server$ cd .blog
owen@ubuntu:~/Desktop/final/server/.blog$ ls -a
.  ..  roo
owen@ubuntu:~/Desktop/final/server/.blog$ cd roo
owen@ubuntu:~/Desktop/final/server/.blog/roo$ ls -a
.  ..  今日更新.txt  小秘密.txt
owen@ubuntu:~/Desktop/final/server/.blog/roo$
```

图 21 博客文件结构

除此博客内容之外, 我们还在文件中写入了标题、作者以及创建时间。针对阅读权限设置的功能, 我们在 **txt** 文件的第一行加入了校验码。如果该博客是所有人可见, 则在第一行写入 **\$all**, 如果该博客是仅自己可见, 则在第一行写入 **\$private+用户名**, 例如当前用户 **roo** 的《小秘密》, 文件开头的校验码应为 **\$privateroo**。

代码如下:

```
1. void write_blog(char *name, int fd){
2.     FILE *blogFp;
3.     FILE *readFp;
4.     char title[50], prompt[150], filename[50], blogInfo[150], context[100]
        , choice[5], command[50];
5.     int flag = 1;
6.     time_t currentTime;
7.
```

```
8.    memset(title, 0, sizeof(title));
9.    memset(prompt, 0, sizeof(prompt));
10.   memset(filename, 0, sizeof(filename));
11.   memset(context, 0, sizeof(context));
12.   memset(choice, 0, sizeof(choice));
13.   memset(command, 0, sizeof(command));
14.   send(fd, "欢迎来到小霸王博客, 请给您的博客起一个标题吧: \n",
15.         sizeof("欢迎来到小霸王博客, 请给您的博客起一个标题吧: \n"), 0);
16.   recv(fd, title, sizeof(title), 0);
17.   title[strlen(title) - 1] = '\0';
18.   system("mkdir .blog");
19.   sprintf(command, "mkdir ../blog/%s", name);
20.   system(command);
21.   sprintf(filename, "../blog/%s/%s.txt", name, title);
22.   blogFp = fopen(filename, "w");
23.   time(&tTime);
24.
25.   send(fd, "谁可以看? (输入序号)\n1.所有人可见\n2.仅自己可见\n",
26.         strlen("谁可以看? (输入序号)\n1.所有人可见\n2.仅自己可见\n"), 0);
27.   recv(fd, choice, sizeof(choice), 0);
28.   if (strcmp(choice, "1\n") == 0) {
29.       sprintf(blogInfo, "$all\n标题: %s\n作者: %s\n日期: %s 正文: \n", title, name, ctime(&tTime));
30.       fputs(blogInfo, blogFp);
31.       sprintf(prompt, "请输入《%s》的内容: (输入\\finish结束)\n", title);
32.       send(fd, prompt, strlen(prompt), 0);
33.       while (flag) {
34.           recv(fd, context, sizeof(context), 0);
35.           if (strcmp(context, "\\finish\n") == 0) {
36.               flag = 0;
37.           } else fputs(context, blogFp);
38.           memset(context, 0, sizeof(context));
39.       }
40.       fclose(blogFp);
41.
42.       readFp = fopen(filename, "r");
43.
44.       char Msg[100];
45.       memset(Msg, 0, sizeof(Msg));
46.       sprintf(Msg, "%s 上传了一篇博客《%s》, 请用\\"read_blog\\"命令查看详细内容.\n", name, title);
47.       SendMsgToAll(Msg, fd);
```

```
48.  
49.     memset(Msg, 0, sizeof(Msg));  
50.     sprintf(Msg, "您上传了一篇博客《%s》，请用\\"read_blog\\"命令查看详  
    细内容。\\n",title);  
51.     send(fd, Msg, strlen(Msg),0);  
52.  
53. } else if (strcmp(choice, "2\\n") == 0) {  
54.     sprintf(blogInfo, "$private%s\\n 标题: %s\\n 作者: %s\\n 日期: %s 正  
    文: \\n", name, title, name, ctime(&tTime));  
55.     fputs(blogInfo, blogFp);  
56.     sprintf(prompt, "请输入《%s》的内容：（输入\\"finish结束）  
    \\n", title);  
57.     send(fd, prompt, strlen(prompt), 0);  
58.     while (flag) {  
59.         recv(fd, context, sizeof(context), 0);  
60.         if (strcmp(context, "\\finish\\n") == 0) {  
61.             flag = 0;  
62.         } else fputs(context, blogFp);  
63.         memset(context, 0, sizeof(context));  
64.     }  
65.     fclose(blogFp);  
66.  
67.     char Msg[100];  
68.     memset(Msg, 0, sizeof(Msg));  
69.     sprintf(Msg, "您上传了一篇博客《%s》，请用\\"read_blog\\"命令查看详  
    细内容。\\n",title);  
70.     send(fd, Msg, strlen(Msg),0);  
71. }  
72. }
```

写完了博客，接下来就到了阅读博客的时候了。用户需要输入`\read_blog`来阅读博客。对于所有人可见的博客，聊天室内所有成员皆可阅读。而对于仅自己可见的博客，作者之外的用户阅读时系统会提示无权查看。我们还加入了点赞功能，当阅读完博客之后，输入 `like`，服务端会将点赞信息传送给博客的作者。具体效果如图 22（左边为 `roo` 用户，右边为 `owen` 用户）。

<pre>owen点赞了您的博客《今日更新》！ \read_blog 请选择您要查看的用户：(输入序号) 1.owen 2.roo (您) 2 输入您要查看的博客：(不需要带后缀) 今日更新.txt 小秘密.txt 今日更新 《今日更新》内容如下： 标题：今日更新 作者：roo 日期：Fri May 22 20:42:07 2020 正文： 今日更新了读写博客功能， 请各位尽情体验。 喜欢的话就点个赞吧！（输入like点赞，回车跳过） like</pre>	<pre>roo上传了一篇博客《今日更新》，请用"\read_blog"命令查 \read_blog 请选择您要查看的用户：(输入序号) 1.owen (您) 2.roo 2 输入您要查看的博客：(不需要带后缀) 今日更新.txt 小秘密.txt 今日更新 《今日更新》内容如下： 标题：今日更新 作者：roo 日期：Fri May 22 20:42:07 2020 正文： 今日更新了读写博客功能， 请各位尽情体验。 喜欢的话就点个赞吧！（输入like点赞，回车跳过） like</pre>
<pre>\read_blog 请选择您要查看的用户：(输入序号) 1.owen 2.roo (您) 2 输入您要查看的博客：(不需要带后缀) 今日更新.txt 小秘密.txt 小秘密 《小秘密》内容如下： 标题：小秘密 作者：roo 日期：Fri May 22 20:47:21 2020 正文： 希望彭成磊老师不会发现我们程序的bug。 hhhhhhh 喜欢的话就点个赞吧！（输入like点赞，回车跳过）</pre>	<pre>\read_blog 请选择您要查看的用户：(输入序号) 1.owen (您) 2.roo 2 输入您要查看的博客：(不需要带后缀) 今日更新.txt 小秘密.txt 小秘密 这是一篇私密博客，您无权访问！</pre>

图 22 读博客功能

可以看到，owen 用户可以阅读《今日更新》，并且可以进行点赞，roo 用户此时也收到了点赞消息。当 owen 用户想要阅读《小秘密》时，则遭到了服务端的拒绝。而此时 roo 用户却能读取顺利，原因就是《小秘密》是 roo 用户的私密博客，他人无权阅读。注意：如果选择的用户没有博客，或者选择的博客标题不存在，系统都会进行提示，如图 23。

```
\read_blog
请选择您要查看的用户：(输入序号)
1.owen (您)
2.roo
1
该用户还没有发过博客哦！
\read_blog
请选择您要查看的用户：(输入序号)
1.owen (您)
2.roo
2
输入您要查看的博客：(不需要带后缀)
今日更新.txt
小秘密.txt
今日
博客不存在！请重新输入命令"\read_blog查找！"
```

图 23 特殊情况提示

读博客功能的原理，首先是调用 `listMember()` 函数，输出当前的用户列表，待用户输入选项之后，服务端确定出选项对应的用户，进入用户的博客文件夹读取文件列表。这里用到了 `popen()` 函数执行 `ls` 命令。`popen()` 函数通过创建一个管道，调用 `fork()` 产生一个子进程，执行一个 `shell` 以运行命令来开启一个进程。与 `system()` 和 `exec` 系列函数不同的是，`popen()` 可以用来获取命令的结果信息。大致流程为：提前创建一个文件描述符，调用 `popen()` 函数运行命令，通过 `fread()` 将结果信息读出，最后调用 `pclose()` 来关闭文件描述符。

对于阅读权限的判断，我们首先需要用 `fgets()` 函数将文件第一行校验码读出，并与当前需要阅读博客的用户进行比较。此时文件指针已经指向了第二行，如果满足访问权限，则用 `freads()` 继续读取接下来的全部内容，如果不满足权限，则提示用户无法查看。

代码如下：

```
1. void read_blog(char *name, int fd){
2.     char choice[5],command[50];
3.     int choiceNumber,i,index=0;
4.     FILE *comStatus;
5.     memset(choice, 0, sizeof(choice));
6.     send(fd, "请选择您要查看的用户：(输入序号)\n", strlen("请选择您要查看的用户：(输入序号)\n"), 0);
7.     listMember(fd);
8.     recv(fd, choice, sizeof(choice), 0);
9.     choice[strlen(choice) - 1] = '\0';
10.    choiceNumber = atoi(choice);
11.    for (i = 0; i < 100; ++i) {
12.        if (user[i].fd != -1) index++;
13.        if (index == choiceNumber) break;
14.    }
15.    sprintf(command,"ls ../blog/%s/",user[i].name);
16.    char blogList[100];
17.    memset(blogList, 0, sizeof(blogList));
18.    comStatus = popen(command, "r");
19.    fread(blogList, sizeof(char), sizeof(blogList), comStatus);
20.    if (blogList[0] == '\0') {
21.        send(fd, "该用户还没有发过博客哦！\n",strlen("该用户还没有发过博客哦！\n"), 0);
22.        pclose(comStatus);
23.        return;
24.    } else {
25.        char title[50],fileName[50];
26.        memset(title,0, sizeof(title));
27.        memset(fileName,0, sizeof(fileName));
```

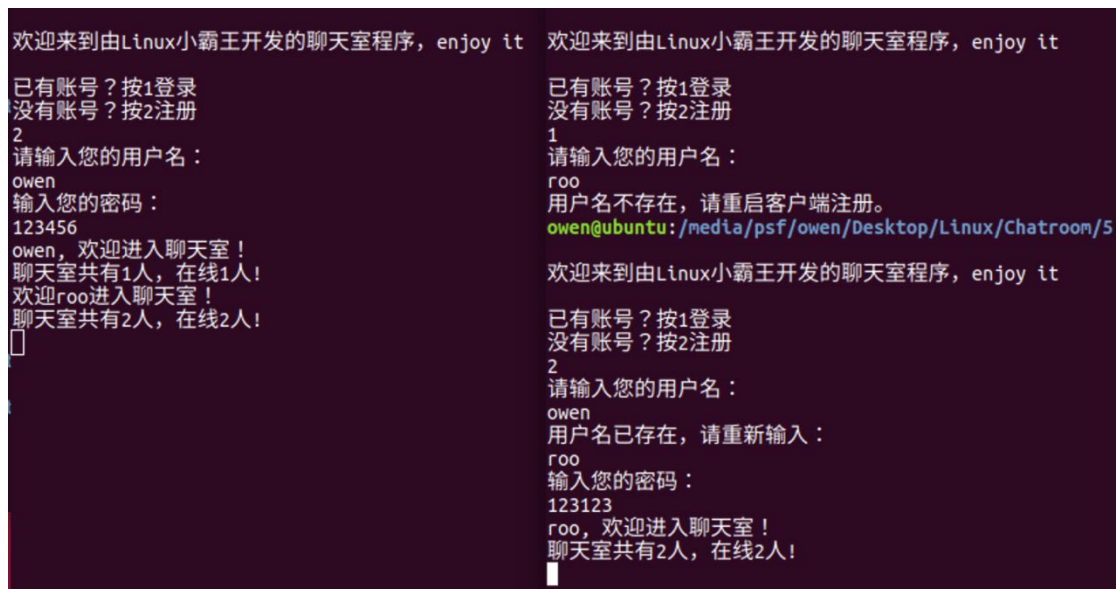
```
28.         send(fd, "输入您要查看的博客: (不需要带后缀)\n", strlen("输入您要查看的
           博客: (不需要带后缀)\n"), 0);
29.         send(fd, blogList, strlen(blogList), 0);
30.         recv(fd, title, sizeof(title), 0);
31.         title[strlen(title) - 1] = '\0';
32.         pclose(comStatus);
33.         FILE *readFp;
34.         sprintf(fileName, "./.blog/%s/%s.txt", user[i].name, title);
35.         readFp = fopen(fileName, "r");
36.         if (readFp == NULL) {
37.             send(fd, "博客不存在! 请重新输入命令\\read_blog 查找! \n\n",
38.                 strlen("博客不存在! 请重新输入命令\\read_blog 查找!
           \n\n"), 0);
39.             return;
40.         } else {
41.             char buf[1024], Msg[50], check[50];
42.             memset(buf, 0, sizeof(buf));
43.             memset(Msg, 0, sizeof(Msg));
44.             memset(check, 0, sizeof(check));
45.             sprintf(check, "$private%s\n", name);
46.             fgets(buf, sizeof(buf), readFp);
47.             if (buf[1] == 'p' && strcmp(check, buf) != 0) {
48.                 send(fd, "这是一篇私密博客, 您无权访问! \n", sizeof("这是一
           篇私密博客, 您无权访问! \n"), 0);
49.             } else {
50.                 memset(buf, 0, sizeof(buf));
51.                 fread(buf, sizeof(char), sizeof(buf), readFp);
52.                 sprintf(Msg, "《%s》内容如下:\n", title);
53.                 send(fd, Msg, strlen(Msg), 0);
54.                 send(fd, buf, strlen(buf), 0);
55.                 char like[10];
56.                 memset(like, 0, sizeof(like));
57.                 send(fd, "喜欢的话就点个赞吧! (输入 like 点赞, 回车跳过)
           \n",
58.                     strlen("喜欢的话就点个赞吧! (输入 like 点赞, 回车跳
           过)\n"), 0);
59.                 recv(fd, like, sizeof(like), 0);
60.                 if (strcmp(like, "like\n") == 0) {
61.                     char likeMsg[50];
62.                     sprintf(likeMsg, "%s 点赞了您的博客《%s》!
           \n", name, title);
63.                     send(user[i].fd, likeMsg, strlen(likeMsg), 0);
64.                 }
65.             }
```



```
66.     }  
67.     fclose(readFp);  
68.     }  
69. }
```

3.8 登录注册功能(Beta)

在进入聊天室之前，我们新加入了登录注册的功能，并且保留了查重功能。已有账户的用户通过按 1 输入账号密码登录，还没有账户的用户通过按 2 进行注册。如图 24。



```
欢迎来到由Linux小霸王开发的聊天室程序, enjoy it  
已有账号?按1登录  
没有账号?按2注册  
2  
请输入您的用户名:  
owen  
输入您的密码:  
123456  
owen, 欢迎进入聊天室!  
聊天室共有1人, 在线1人!  
欢迎roo进入聊天室!  
聊天室共有2人, 在线2人!  
□  
欢迎来到由Linux小霸王开发的聊天室程序, enjoy it  
已有账号?按1登录  
没有账号?按2注册  
1  
请输入您的用户名:  
roo  
用户名不存在, 请重启客户端注册。  
owen@ubuntu:/media/psf/owen/Desktop/Linux/Chatroom/5  
欢迎来到由Linux小霸王开发的聊天室程序, enjoy it  
已有账号?按1登录  
没有账号?按2注册  
2  
请输入您的用户名:  
owen  
用户名已存在, 请重新输入:  
roo  
输入您的密码:  
123123  
roo, 欢迎进入聊天室!  
聊天室共有2人, 在线2人!  
□
```

图 24 登录注册功能

除此之外，我们在 `listMember` 中加入了在线和离线状态的显示，对离线的用户，私发功能将不可使用。如图 25。



```
\member  
聊天室成员有:  
1.owen (离线)  
2.roo (在线)  
3.frank (您)  
\send to  
请选择您的私聊对象: (输入序号)  
1.owen (离线)  
2.roo (在线)  
3.frank (您)  
1  
输入您要发送的内容:  
hi  
owen当前不在线!
```

图 25 状态显示

注：当前功能正在测试过程中，暂时不展示源码。

目前遇到的问题：该功能在多台 **mac** 电脑的终端和 **mac** 版本的 **ubuntu** 虚拟机中皆成功运行，但在 **PC** 上遇到了注册之后退出聊天室就无法再登录的问题。具体原因暂不得知。后续会针对 **PC** 平台进行优化。

CHAPTER 4

实验总结

本项目自 4 月 25 日动工至今，共耗时约一个月，期间各组员目标明晰、分工明确，工作效率令人满意。项目各功能具体时间线如下：

`void emoji(char* name, int fd);` //表情包功能 最后更新：4.29

`void listMember(int fd);` //输出成员列表功能 最后更新：5.11

`void send_file(char* name, int fd);` //tcp 上传文件 最后更新：5.23

`void download(char* name, int fd);` //tcp 下载文件 最后更新：5.23

`void send_to(char *name, int fd);` //私发消息 最后更新：5.11

`void message(char *name, int fd);` //查看聊天记录 最后更新：5.20

`void object_detection(char* name, int fd, char *file);` //目标识别 最后更新：5.18

`void func(int fd);` //查看功能

`void write_blog(char *name, int fd);` //写博客 最后更新：5.20

`void read_blog(char *name, int fd);` //读博客 最后更新：5.20

我们除实现聊天室最基本的聊天功能，还提供了传文件、写博客甚至神经网络目标识别的现代功能，颇具创新性。组员在进行技术交流的过程中学会了如何实现软件的开发与管理，受益匪浅。

在实现如此丰富功能的背后是组员不懈的努力。由于我们对网络函数、线程函数所知甚少，编写时常常遇到一些未知的错误，在网上也并不能找到很全面的解释。因此，我们周而复始地在程序中插入断点进行 `debug`，几次熬到了凌晨。即使是在 5 月 20 日推出了完整的版本后，公测时仍然遇到零星的错误。时至 5 月 23 日才勉强将所有大大小小的 `bug` 进行了修复，推出我们最终的版本。最后，再次感谢组员们的兢兢业业。