

## GTK+实现linux聊天室代码详解-server端\_海上的雨-CSDN博客

[查看原代码请点击此链接](#)

注意！！此聊天室对红帽无兼容。需在其他linux系统上运行，如“深度”。

我相信只要认识字就能学会。

GTK+实现linux聊天室代码详解-client端:[https://blog.csdn.net/qq\\_41915690/article/details/85160274](https://blog.csdn.net/qq_41915690/article/details/85160274)

- 1.
- 2.
- 3.
- 4.
- 5.
- 6.
- 7.
- 8.
- 9.
- 10.
- 11.
- 12.
- 13.
- 14.
- 15.
- 16.
- 17.
- 18.
- 19.
- 20.
- 21.
- 22.
- 23.
- 24.
- 25.
- 26.
- 27.
- 28.
- 29.
- 30.

```
31.
32.
33. void get_now_time(char *nt){
34.
35.
36. struct tm *tmp_ptr = NULL;
37.     tmp_ptr = localtime(&tmpcal_ptr);
38.     sprintf(nt,"%d:%d:%d", tmp_ptr->tm_hour, tmp_ptr->tm_min, tmp_ptr->tm_sec);
39.
40.
41. int init_sem(int sem_id, int init_value){
42.
43.
44.
45.
46.
47.
48.     sem_union.val = init_value;
49. if(semctl(sem_id, 0, SETVAL, sem_union) == -1){
50.     syslog(LOG_ERR, "Initialize semaphore");
51.     perror("Initialize semaphore");
52.
53.
54.
55.
56.
57.
58.
59.
60.
61.
62.
63.
64. if(semctl(sem_id, 1, IPC_RMID, sem_union)==-1){
```

```
65.             syslog(LOG_ERR, "Delete semaphore");
66.             perror("Delete semaphore");
67.
68.
69.
70.
71.
72.
73.
74.
75.             sem_b.sem_flg = SEM_UNDO;
76. if(semop(sem_id, &sem_b, 1)==-1){
77.             syslog(LOG_ERR, "P operation");
78.
79.
80.
81.
82.
83.
84.
85.
86.
87.
88.             sem_b.sem_flg = SEM_UNDO;
89. if(semop(sem_id, &sem_b, 1) == -1){
90.             syslog(LOG_ERR, "V operation");
91.
92.
93.
94.
95.
96.
97.
98.
99.
100. void send_all(char loadsend[BUFFER_SIZE]){
```

```
101.
102. for(i=0;i<THREAD_NUMBER;i++){
103.
104.             send(users[i].client_fd, loadsend, strlen(loadsend), 0);
105.
106.
107.
108.
109. void send_only(char name[50],char loadsend[BUFFER_SIZE]){
110.
111. for(j=0;j<THREAD_NUMBER;j++){
112. if((users[j].login==1)&&(strcmp(users[j].name,name)==0)){
113.             send(users[j].client_fd, loadsend, strlen(loadsend), 0);
114.
115.
116.
117.
118. void strdeal(char s[],char se_name[]){
119.
120.
121.
122.
123. char send_buf[BUFFER_SIZE];
124. char send_buf2[BUFFER_SIZE];
125.     memset(sign, 0, strlen(sign));
126.     memset(name, 0, strlen(name));
127.     memset(buf, 0, strlen(buf));
128.     memset(send_buf, 0, strlen(send_buf));
129.     memset(send_buf2, 0, strlen(send_buf2));
```

```
130.  
131.  
132.  
133.  
134.  
135.  
136.  
137.  
138.  
139.  
140.  
141.  
142.  
143.  
144.  
145.  
146.  
147.  
148.  
149.  
150.  
151.  
152.  
153.  
154.  
155.  
156.  
157.  
158.  
159.  
160.  
161.  
162.  
163.  
164.  
165.  
166.  
167.  
168.  
169.  
  
170. if(strcmp(sign,"All")==0){  
171.
```

```
172.         sprintf(send_buf,"%s用户< %s >群发消息->\t\t%s:\n\t%s","User:",se_name,nt,buf);
173.
174.
175.
176.         sprintf(send_buf,"%s用户< %s >----->\t\t%s:\n\t%s","User:",se_name,nt,buf);
177.         send_only(name,send_buf);
178.         sprintf(send_buf2,"User: %s: say\n\t%s",nt,buf);
179.         send_only(se_name,send_buf2);
180.
181.
182.
183.
184. void *thrd_func(void *arg)
185.
186.
187.
188.
189.
190.         memset(users[i].buf , 0, sizeof(users[i].buf));
191. if ((recvbytes = recv(users[i].client_fd, users[i].buf, BUFFER_SIZE, 0)) <= 0)
192.
193.
194.
195.
196.         sprintf(end,"%s%s%s\n用户: %s%s\n","Inform:",nt,"-通知: ",users[i].name,"退出聊天室");
197.
198.
199.
200.         close(users[i].client_fd);
201.
202.
203. for(j;j<THREAD_NUMBER;j++){
```

```
204.
205.
206.
207.             printf("%s用户退出，还可以上线%d个\n",users[i].name, n);
208.
209.
210.             strdeal(users[i].buf,users[i].name);
211.
212.
213.
214.
215.
216. int bindPort(unsigned short int port)
217.
218.
219.
220. struct sockaddr_in my_addr;
221. if ((sockfd = socket(AF_INET,SOCK_STREAM,0))== -1)
222.
223.             syslog(LOG_ERR, "socket");
224.
225.
226.
227.
228.         bzero(&my_addr, sizeof(my_addr));
229.         my_addr.sin_family = AF_INET;
230.         my_addr.sin_port = htons(port);
231.         my_addr.sin_addr.s_addr = INADDR_ANY;
232.         memset(&(my_addr.sin_zero),0,8);
233.
234.
235.
```

```
236.         setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, &i, sizeof(i));
237. if (bind(sockfd, (struct sockaddr*)&my_addr, sizeof(struct sockaddr)) == -1)
238.
239.         syslog(LOG_ERR, "fail to bind");
240.
241.
242.
243.
244.
245.
246. void my_func(int sign_no)
247.
248.
249.         sprintf(loadsend,"%s%s","Inform:","over");
250.
251.
252.
253.
254.
255.
256. int main(int argc, char *argv[])
257.
258.
259.
260.
261.
262.
263.
264. struct sockaddr_in client_sockaddr;
265.
266.
267.         sockfd = bindPort(MYPORT);
268. for(i=0;i<THREAD_NUMBER;i++)
```



```
269.
270. if (listen(sockfd, THREAD_NUMBER) == -1)
271.
272.         syslog(LOG_ERR, "listen");
273.
274.
275.         printf("Listening....\n");
276.         sem_id = semget(ftok("/", 1), 1, 1, 0666|IPC_CREAT);
277.         init_sem(sem_id, THREAD_NUMBER);
278.
279.         openlog("daemon_syslog", LOG_PID, LOG_DAEMON);
280.
281.         signal(SIGQUIT, my_func);
282.
283.
284.
285.
286.
287.
288.
289.
290.
291.
292.
293.
294.
295.
296. for(j;j<THREAD_NUMBER;j++){
297.
298.
299.
300.         printf("已经上线%d个用户，还可以上线%d个\n",n,THREAD_NUMBER-n);
```

```
301.
302.
303. if ((users[i].client_fd = accept(sockfd,(struct sockaddr*)&client_sockaddr, &sin_size)) == -1){
304.         syslog(LOG_ERR, "accept");
305.
306.
307.
308.         inet_ntop(AF_INET, &client_sockaddr.sin_addr, users[i].address, sizeof(users[i].address));
309. if ((recvbytes=recv(users[i].client_fd, users[i].name, BUFFER_SIZE, 0)) <= 0){
310.
311.
312.
313.         printf("本次连接的是用户: %s\n",users[i].name);
314.
315.
316. for(j=0;j<THREAD_NUMBER;j++){
317.
318. if(strcmp(users[i].name,users[j].name)==0){
319.
320.
321.
322.
323.         send(users[i].client_fd, "g", strlen("g"), 0);
324.
325.
326.
327.
328.         send(users[i].client_fd, "Welcome", strlen("Welcome"), 0);
329.
330.
331.         memset(loadsend, 0, 100);
```

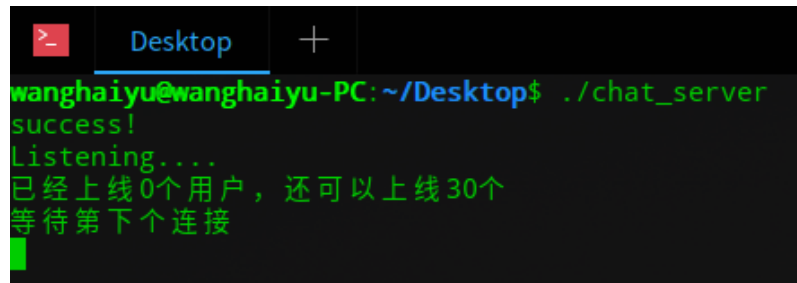
```
332.  
333.             sprintf(loadsend,"%s%s%s\n用户: %s%s\n","Inform:",nt,"-通知: ",users[i].name,"-上线了!\n\t大家欢迎!");  
334.  
335.             res = pthread_create(&users[i].thread, NULL, thrd_func, (void*)i);  
336.  
337.  
338.             syslog(LOG_ERR, "Create thread failed");  
339.             perror("Create thread failed");  
340.  
341.  
342.  
343.  
344.  
345.  
346.  
347.  
348.  
349.  
350.  
351.  
352.  
353.  
354.  
355.  
356.  
357.  
358.  
359.  
360.
```

展示（注：后期有所优化，所提供代码与展示效果细节处有所差别，但影响不大。）

运行结果截图

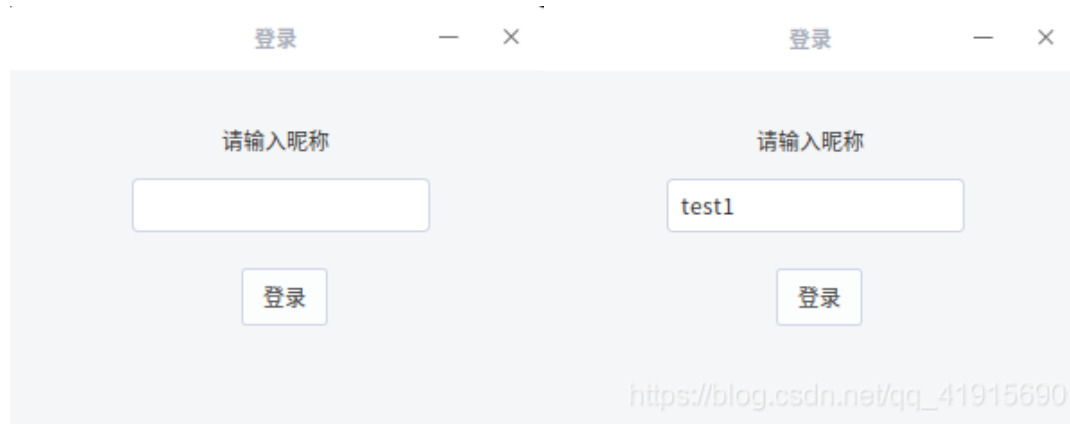
不运行服务器而运行客户端程序时不可登录，客户端自动退出。

运行服务器：

A terminal window titled 'Desktop' with a red icon. The prompt is 'wanghaiyu@wanghaiyu-PC: ~/Desktop\$'. The command './chat\_server' has been executed, resulting in the following output: 'success!', 'Listening....', '已经上线0个用户，还可以上线30个', and '等待第下个连接'. A green cursor is visible on the line following the last message.

```
wanghaiyu@wanghaiyu-PC: ~/Desktop$ ./chat_server
success!
Listening....
已经上线0个用户，还可以上线30个
等待第下个连接
```

运行客户端:

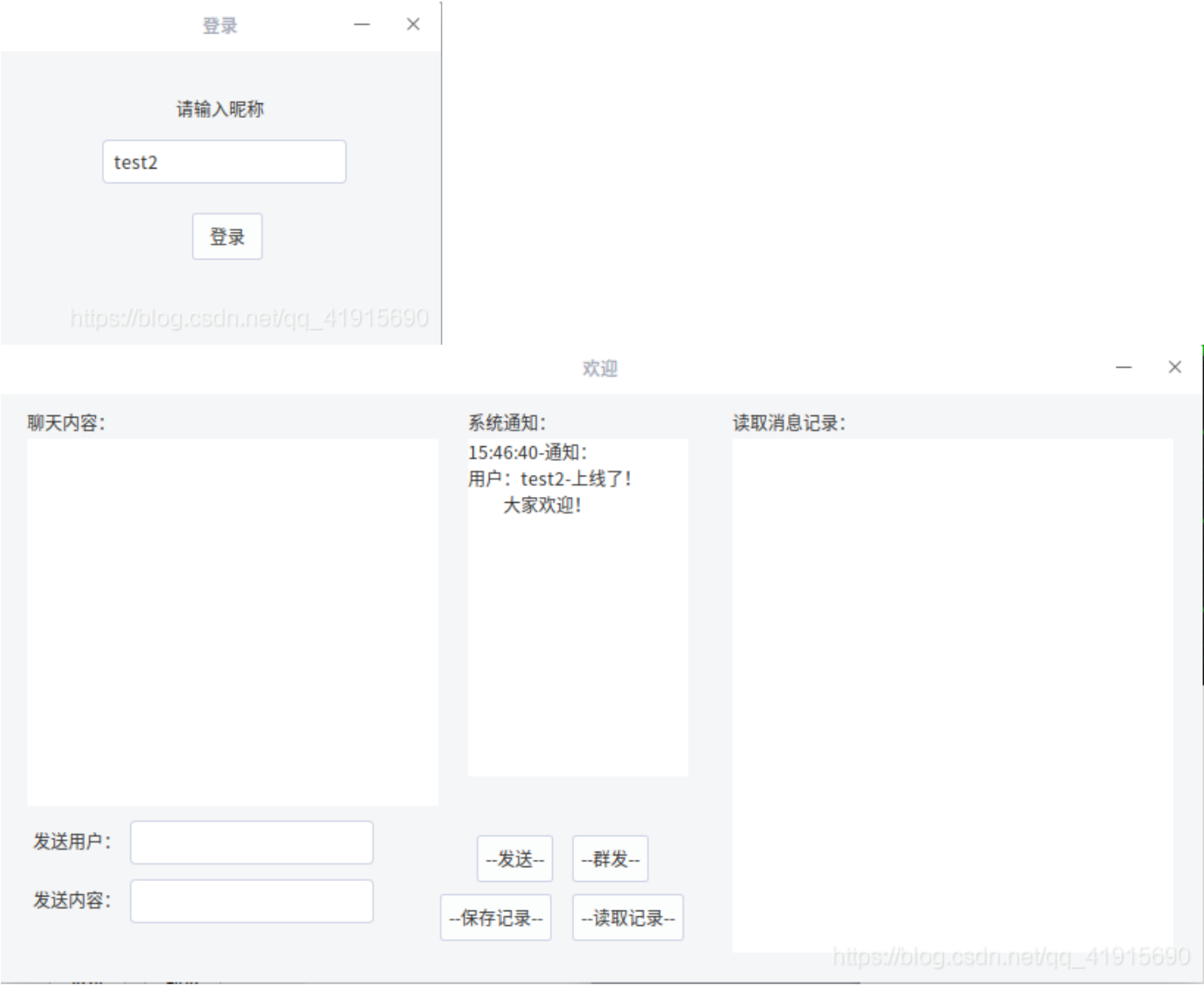




运行第二个客户端时若用户已登录:



登录第二个客户端:



A terminal window titled 'wanghaiyu@wanghaiyu-PC: Desktop' with a red close button. The terminal shows the execution of './chat\_server' which outputs 'success!' and 'Listening....'. It then shows three consecutive connections from users 'test1' and 'test2', with status updates on the number of online users (0 to 30, 1 to 29, 2 to 28) and prompts to wait for the next connection. A small square cursor is visible at the bottom left.

```
wanghaiyu@wanghaiyu-PC: Desktop  
wanghaiyu@wanghaiyu-PC: ~/Desktop$ ./chat_server  
success!  
Listening....  
已经上线0个用户，还可以上线30个  
等待第下个连接  
本次连接的是用户：test1  
已经上线1个用户，还可以上线29个  
等待第下个连接  
本次连接的是用户：test1  
已经上线1个用户，还可以上线29个  
等待第下个连接  
本次连接的是用户：test2  
已经上线2个用户，还可以上线28个  
等待第下个连接  
□
```

[https://blog.csdn.net/qq\\_41915690](https://blog.csdn.net/qq_41915690)

用户test1向test2单发消息:



A terminal window titled 'wanghaiyu@wanghaiyu-PC: Desktop' with a red icon on the left and a '+' icon on the right. The terminal text is as follows:

```
wanghaiyu@wanghaiyu-PC:~/Desktop$ ./chat_server
success!
Listening....
已经上线0个用户，还可以上线30个
等待第下个连接
本次连接的是用户：test1
已经上线1个用户，还可以上线29个
等待第下个连接
本次连接的是用户：test1
已经上线1个用户，还可以上线29个
等待第下个连接
本次连接的是用户：test2
已经上线2个用户，还可以上线28个
等待第下个连接
□
```

[https://blog.csdn.net/qq\\_41915690](https://blog.csdn.net/qq_41915690)

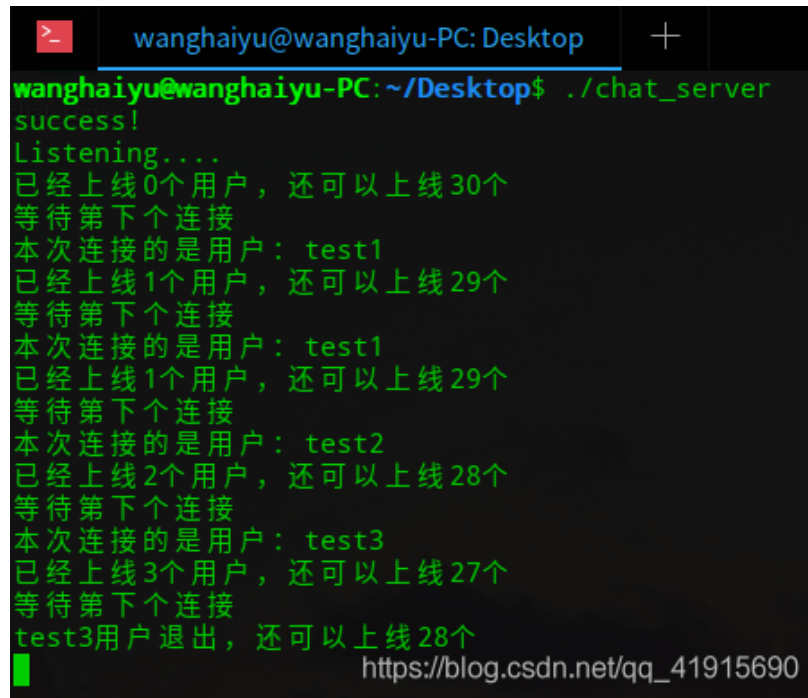


登录第三个客户端后使用第三个客户端群发消息:



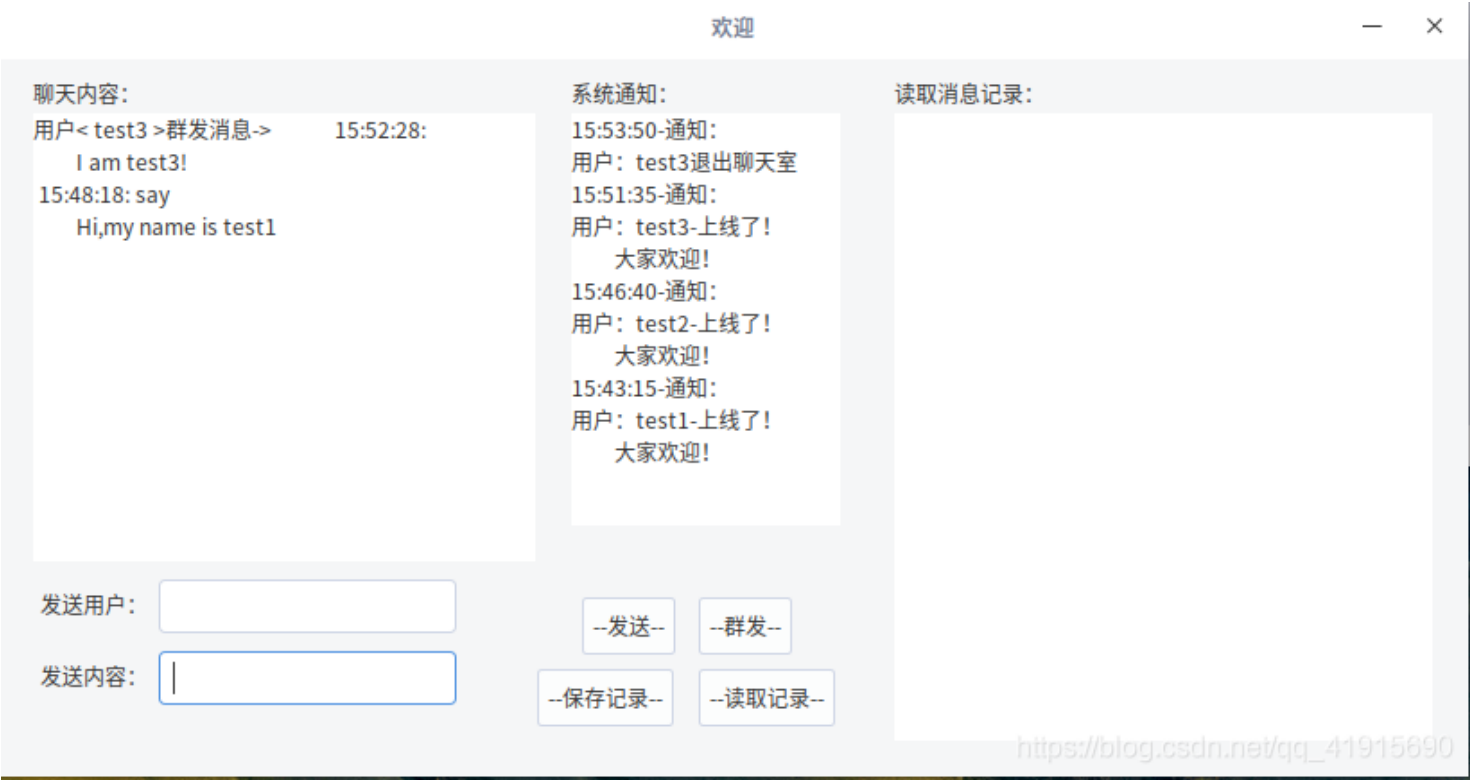




A terminal window titled 'wanghaiyu@wanghaiyu-PC: Desktop' with a red icon and a '+' button. The terminal shows the execution of './chat\_server' which outputs 'success!' and 'Listening....'. It then shows three successful connections from users 'test1', 'test2', and 'test3', each increasing the online user count by one. The output for 'test3' shows 'test3用户退出, 还可以上线 28个' followed by a green cursor.

```
wanghaiyu@wanghaiyu-PC: Desktop
wanghaiyu@wanghaiyu-PC: ~/Desktop$ ./chat_server
success!
Listening....
已经上线0个用户, 还可以上线30个
等待第下个连接
本次连接的是用户: test1
已经上线1个用户, 还可以上线29个
等待第下个连接
本次连接的是用户: test1
已经上线1个用户, 还可以上线29个
等待第下个连接
本次连接的是用户: test2
已经上线2个用户, 还可以上线28个
等待第下个连接
本次连接的是用户: test3
已经上线3个用户, 还可以上线27个
等待第下个连接
test3用户退出, 还可以上线 28个
█
```

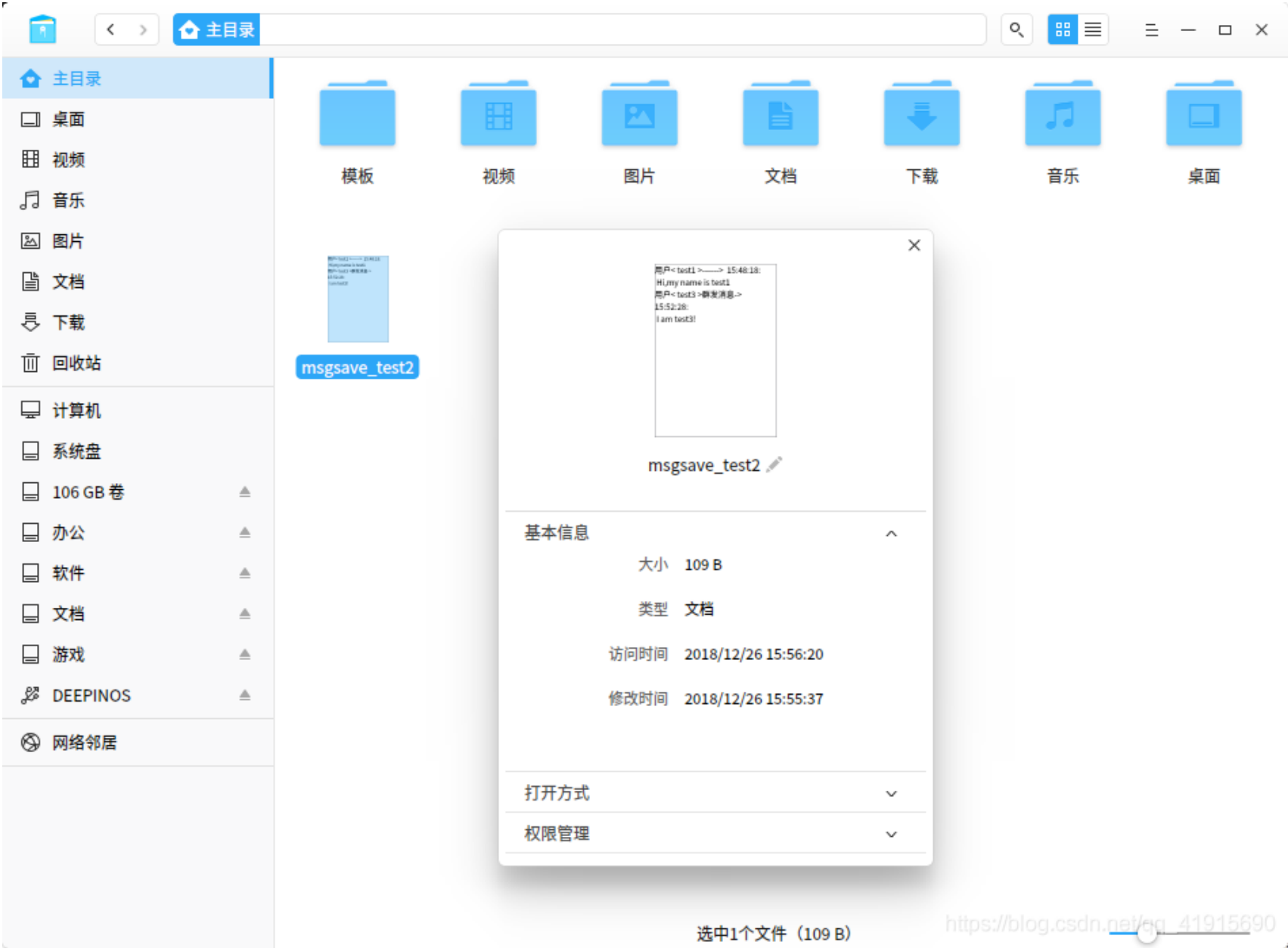
退出一个客户端:





保存记录:





打开(O) + msgsave\_test2 保存(S) | - □ ×

用户< test1 >-----> 15:48:18:  
Hi,my name is test1  
用户< test3 >群发消息-> 15:52:28:  
I am test3!

纯文本 制表符宽度: 3 第 4 行, 第 20 列 插入



读取记录

若退出后重新登录后再读取记录:





服务器退出:





A terminal window titled 'Desktop' with a '+' icon. The prompt is 'wanghaiyu@wanghaiyu-PC: ~/Desktop\$'. The command './chat\_server' has been executed, resulting in the following output:

```
wanghaiyu@wanghaiyu-PC: ~/Desktop$ ./chat_server
success!
Listening....
已经上线0个用户，还可以上线30个
等待第下个连接
本次连接的是用户：test1
已经上线1个用户，还可以上线29个
等待第下个连接
本次连接的是用户：test1
已经上线1个用户，还可以上线29个
等待第下个连接
本次连接的是用户：test2
已经上线2个用户，还可以上线28个
等待第下个连接
本次连接的是用户：test3
已经上线3个用户，还可以上线27个
等待第下个连接
test3用户退出，还可以上线28个
^Z即将退出服务器
wanghaiyu@wanghaiyu-PC: ~/Desktop$
```

[https://blog.csdn.net/qq\\_41915690](https://blog.csdn.net/qq_41915690)