

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

[查看原代码请点击此超链接](#)

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”。

加油学习！

GTK+实现linux聊天室代码详解-server端: https://blog.csdn.net/qq_41915690/article/details/85160249

```
1. #include <gtk/gtk.h>    // 头文件
2.
3.
4. #include <netinet/in.h> //定义数据结构sockaddr_in
5. #include <sys/socket.h> //定义socket函数以及数据结构
6.
7.
8.
9.
10.
11.
12.
13.
14.
15.
16.
17.
18. struct sockaddr_in clientaddr;
19.
20. char fname[]="/var/tmp/";
21.
22.
23.
```

```
24. int init_sem(int sem_id, int init_value){
25.
26.
27.
28.
29.
30.
31.     sem_union.val = init_value;
32. if(semctl(sem_id, 0, SETVAL, sem_union) == -1){
33.     perror("Initialize semaphore");
34.
35.
36.
37.
38.
39.
40.
41.
42.
43.
44.
45.
46. if(semctl(sem_id, 1, IPC_RMID, sem_union)==-1){
47.     perror("Delete semaphore");
48.
49.
50.
51.
52.
53.
54.
55.
56.     sem_b.sem_flg = SEM_UNDO;
57. if(semop(sem_id, &sem_b, 1)==-1){
58.
59.
```

```
60.
61.
62.
63.
64.
65.
66.
67.

68.         sem_b.sem_flg = SEM_UNDO;
69. if(semop(sem_id, &sem_b, 1) == -1){
70.
71.
72.
73.
74.
75.
76.
77.

78. void deal_pressed(GtkWidget *button, gpointer entry){
79.
80.
81.
82.

83.         host = gethostbyname("127.0.0.1");
84.         buff = (char *)malloc(9);
85.

86. const gchar *text = gtk_entry_get_text(GTK_ENTRY(entry));
87.
88.
89.
90.

91. if ((clientfd = socket(AF_INET, SOCK_STREAM, 0)) == -1)
92.
93.         perror("fail to create socket");
```

```
94.
95.
96.         bzero(&clientaddr, sizeof(clientaddr));
97.         clientaddr.sin_family = AF_INET;
98.         clientaddr.sin_port = htons((uint16_t)atoi("8787"));
99.         clientaddr.sin_addr = *((struct in_addr *)host->h_addr);
100.
101. if (connect(clientfd, (struct sockaddr *)&clientaddr, sizeof(struct sockaddr)) == -1)
102.
103.         perror("fail to connect");
104.
105.
106. if ((sendbytes = send(clientfd, text, strlen(text), 0)) == -1)
107.
108.
109.
110.
111.
112. if (recv(clientfd, buff, 7, 0) == -1)
113.
114.
115.
116.
117.
118.
119.         gtk_widget_destroy(window);
120.
121.
122.
123.         dialog = gtk_message_dialog_new((gpointer)window,
124.         GTK_DIALOG_DESTROY_WITH_PARENT,
```

```
125.
126.
127.
128.             gtk_window_set_title(GTK_WINDOW(dialog), "拒绝");
129.             gtk_dialog_run(GTK_DIALOG(dialog));
130.             gtk_widget_destroy(dialog);
131.
132.
133.
134.
135.
136. void login(int argc, char *argv[]){
137.
138.
139.
140.     window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
141.
142.     gtk_window_set_title(GTK_WINDOW(window), "登录");
143.
144.     gtk_window_set_position(GTK_WINDOW(window), GTK_WIN_POS_CENTER);
145.
146.     gtk_widget_set_size_request(window, 300, 200);
147.
148.     gtk_window_set_resizable(GTK_WINDOW(window), FALSE);
149.
150.     g_signal_connect(window, "destroy", G_CALLBACK(gtk_main_quit), NULL);
151.
152.     GtkWidget *fixed = gtk_fixed_new();
153.
```

```
154.         gtk_container_add(GTK_CONTAINER (window), fixed);
155.
156.         GtkWidget *label_one = gtk_label_new("请输入昵称");
157.
158.         gtk_fixed_put(GTK_FIXED(fixed), label_one,120,30);
159.
160.         GtkWidget *entry = gtk_entry_new();
161.
162.         gtk_entry_set_max_length(GTK_ENTRY(entry),50);
163.
164.         gtk_editable_set_editable(GTK_EDITABLE(entry), TRUE);
165.
166.         gtk_fixed_put(GTK_FIXED(fixed), entry,70,60);
167.
168.         GtkWidget *button = gtk_button_new_with_label("  登录  ");
169.
170.         gtk_fixed_put(GTK_FIXED(fixed), button,130,110);
171.
172.         g_signal_connect(button, "pressed", G_CALLBACK(deal_pressed), entry);
173.
174.         gtk_widget_show_all(window);
175.
176.
177.
178.
179.
180.
181. GtkWidget *bufferuser;
```

```
182. GtkTextBuffer *buffernote;
183.
184. void savelinetxt(char buf[]){
185.
186.     lock3.l_whence = SEEK_SET;
187.
188.
189.
190.
191.
192.     b_file = open(fname,O_WRONLY|O_CREAT|O_APPEND, S_IRUSR|S_IWUSR|S_IRGRP|S_IROTH);
193.     fcntl(b_file, F_SETLKW, &lock3);
194.     write(b_file, buf, strlen(buf));
195.     fcntl(b_file, F_UNLCK, &lock3);
196.
197.
198.
199.
200. void sendtouser(GtkButton *button, gpointer entry){
201.
202.     buf = (char *)malloc(1024);
203.
204.
205. const gchar *text = gtk_entry_get_text(GTK_ENTRY(entry));
206. const char *but = gtk_button_get_label(button);
207.
208.
209.
210.
211. if(strcmp(but,"--发送--")==0){
```

```
212. const gchar *name = gtk_entry_get_text(GTK_ENTRY(entryname));
213.
214.
215.
216.
217.             sprintf(buf,"User:%d:%s%s\n",strlen(name),name,text);
218. if ((sendbytes = send(clientfd, buf, strlen(buf), 0)) == -1)
219.
220.
221.
222.
223.
224.             sprintf(buf,"%s%s\n","All::",text);
225. if ((sendbytes = send(clientfd, buf, strlen(buf), 0)) == -1)
226.
227.
228.
229.
230.
231.
232.
233.
234.
235. void savetxt(GtkButton *button, gpointer entry){
236.
237.         lock1.l_whence = SEEK_SET;
238.
239.
240.
241.
242.
243.
244.         lock2.l_whence = SEEK_SET;
245.
246.
```



```
247.
248.
249.
250.

251. unsigned char buff[1024];

252.
253.

254.     sprintf(txt_name,"%s%s","./msgsave_",user_name);
255.     src_file = open(fname, O_RDONLY);
256.     dest_file = open(txt_name,O_WRONLY|O_CREAT, S_IRUSR|S_IWUSR|S_IRGRP|S_IROTH);
257. if (src_file< 0 || dest_file< 0)

258.
259.
260.

261.     fcntl(dest_file, F_SETLKW, &lock1);
262.     fcntl(src_file, F_SETLKW, &lock2);
263. while ((real_read_len = read(src_file, buff, sizeof(buff))) > 0)

264.

265.         write(dest_file, buff, real_read_len);

266.

267.     fcntl(dest_file, F_UNLCK, &lock1);
268.     fcntl(src_file, F_UNLCK, &lock2);

269.
270.
271.
272.
273.

274. void readtxt(GtkButton *button, gpointer entry){

275.
276.
```

```
277.         sprintf(txt_name,"%s%s", "./msgsave_", user_name);
278.
279.
280. if((dest_file=fopen(txt_name,"r"))==NULL){
281.
282.
283.
284.
285.         fgets(StrLine,1024,dest_file);
286.         gtk_text_buffer_get_bounds(GTK_TEXT_BUFFER(buffers), &start, &end);
287.         gtk_text_buffer_insert(GTK_TEXT_BUFFER(buffers), &end, StrLine, strlen(StrLine));
288.
289.
290.
291.
292. void *strdeal(void *arg){
293.
294.
295.
296.
297.         lock.l_whence = SEEK_SET;
298.
299.
300.
301.
302.
303.
304.         memset(sign, 0, strlen(sign));
305.         memset(buf, 0, strlen(buf));
306. if(recv(clientfd, s, 1024, 0) <= 0)
307.
308.
309.
```

```
310.
311.
312.
313.
314.
315.
316.
317.
318.
319.
320.
321.
322.
323.
324.
325.
326.
327.
328.
329.
330. if(strcmp(sign,"User")==0){
331.         b_file = open(fname,O_WRONLY|O_CREAT|O_APPEND, S_IRUSR|S_IWUSR|S_IRGRP|S_IROTH);
332.         fcntl(b_file, F_SETLKW, &lock);
333.         write(b_file, buf, strlen(buf));
334.         fcntl(b_file, F_UNLCK, &lock);
335.
336.
337.
338.         gtk_text_buffer_get_bounds(GTK_TEXT_BUFFER(bufferuser),&start,&end);
339.         gtk_text_buffer_insert(GTK_TEXT_BUFFER(bufferuser),&start,buf,strlen(buf));
340.
341.
342.
343.         gtk_text_buffer_get_bounds(GTK_TEXT_BUFFER(buffernotice),&start,&end);
344. if(strcmp(buf,"over")==0){
```

```
345.                 strcpy(buf,"服务停止，程序即将退出\n");
346.
347.                 gtk_text_buffer_insert(GTK_TEXT_BUFFER(buffernotice),&start,buf,strlen(buf));
348.
349.
350.
351.
352.
353.
354.
355.                 gtk_text_buffer_insert(GTK_TEXT_BUFFER(buffernotice),&start,buf,strlen(buf));
356.
357.
358.
359.
360.
361.
362. void homepage(int argc,char *argv[]){
363.
364.         buf = (char *)malloc(1024);
365.
366.
367.
368.
369.
370.         home = gtk_window_new(GTK_WINDOW_TOPLEVEL);
371.
372.         gtk_window_set_title(GTK_WINDOW(home), "欢迎");
373.
374.         gtk_window_set_position(GTK_WINDOW(home), GTK_WIN_POS_CENTER);
375.
376.         gtk_widget_set_size_request(home, 820, 400);
```

```
377.  
378.         gtk_window_set_resizable(GTK_WINDOW(home), FALSE);  
379.  
380.         g_signal_connect(home, "destroy", G_CALLBACK(gtk_main_quit), NULL);  
381.  
382.         GtkWidget *fixed = gtk_fixed_new();  
383.         gtk_container_add(GTK_CONTAINER(home), fixed);  
384.  
385.  
386.  
387.  
388.         label_one = gtk_label_new("聊天内容: ");  
389.  
390.         gtk_fixed_put(GTK_FIXED(fixed), label_one,20,10);  
391.  
392.  
393.         label_two = gtk_label_new("系统通知: ");  
394.         gtk_fixed_put(GTK_FIXED(fixed), label_two,320,10);  
395.  
396.         label_two = gtk_label_new("发送用户: ");  
397.         gtk_fixed_put(GTK_FIXED(fixed), label_two,24,295);  
398.  
399.         label_two = gtk_label_new("发送内容: ");  
400.         gtk_fixed_put(GTK_FIXED(fixed), label_two,24,335);  
401.  
402.         label_two = gtk_label_new("读取消息记录: ");  
403.         gtk_fixed_put(GTK_FIXED(fixed), label_two,500,10);
```

```
404.
405.
406.     entryname = gtk_entry_new();
407.
408.     gtk_entry_set_max_length(GTK_ENTRY(entryname),500);
409.     gtk_editable_set_editable(GTK_EDITABLE(entryname), TRUE);
410.     gtk_fixed_put(GTK_FIXED(fixed), entryname,90,290);
411.
412.     GtkWidget *entry = gtk_entry_new();
413.     gtk_entry_set_max_length(GTK_ENTRY(entry),500);
414.     gtk_editable_set_editable(GTK_EDITABLE(entry), TRUE);
415.     gtk_fixed_put(GTK_FIXED(fixed), entry,90,330);
416.
417.     GtkWidget *bsend = gtk_button_new_with_label("--发送--");
418.     gtk_fixed_put(GTK_FIXED(fixed), bsend,325,300);
419.
420.     GtkWidget *send_all = gtk_button_new_with_label("--群发--");
421.     gtk_fixed_put(GTK_FIXED(fixed), send_all,390,300);
422.
423.     GtkWidget *save = gtk_button_new_with_label("--保存记录--");
424.     gtk_fixed_put(GTK_FIXED(fixed), save,300,340);
425.
426.     GtkWidget *read_m = gtk_button_new_with_label("--读取记录--");
427.     gtk_fixed_put(GTK_FIXED(fixed), read_m,390,340);
428.
429.
```

```
430.         g_signal_connect(bsend, "pressed", G_CALLBACK(sendtouser), entry);
431.         g_signal_connect(send_all, "pressed", G_CALLBACK(sendtouser), entry);
432.         g_signal_connect(save, "pressed", G_CALLBACK(savetxt), entry);
433.         g_signal_connect(read_m, "pressed", G_CALLBACK(readtxt), entry);
434.
435.
436.         GtkWidget *view = gtk_text_view_new();
437.         gtk_widget_set_size_request (view, 280, 250);
438.         gtk_text_view_set_cursor_visible(GTK_TEXT_VIEW(view), FALSE);
439.         gtk_fixed_put(GTK_FIXED(fixed), view, 20, 30);
440.
441.         bufferuser=gtk_text_view_get_buffer(GTK_TEXT_VIEW(view));
442.
443.
444.         GtkWidget *name_view = gtk_text_view_new();
445.         gtk_widget_set_size_request (name_view, 150, 230);
446.         gtk_text_view_set_cursor_visible(GTK_TEXT_VIEW(name_view), FALSE);
447.         gtk_fixed_put(GTK_FIXED(fixed), name_view, 320, 30);
448.         buffernotice=gtk_text_view_get_buffer(GTK_TEXT_VIEW(name_view));
449.
450.
451.
452.         GtkWidget *viewtxt = gtk_text_view_new();
453.         gtk_widget_set_size_request (viewtxt, 300, 350);
454.         gtk_text_view_set_cursor_visible(GTK_TEXT_VIEW(viewtxt), FALSE);
455.         gtk_fixed_put(GTK_FIXED(fixed), viewtxt, 500, 30);
456.
```

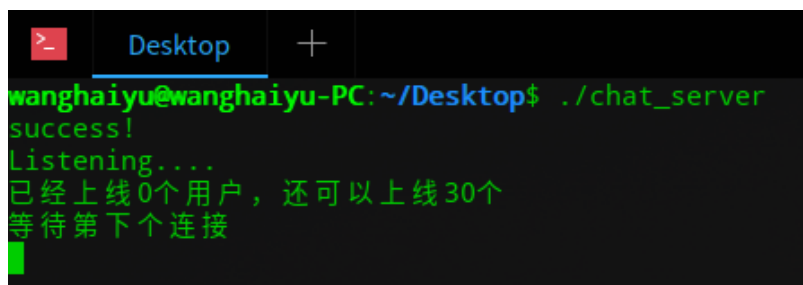
```
457.         buffers=gtk_text_view_get_buffer(GTK_TEXT_VIEW(viewtxt));
458.
459.
460.
461.         gtk_widget_show_all(home);
462.
463.
464.
465.
466.         res = pthread_create(&thread, NULL, strdeal, NULL);
467.
468.
469.
470.
471.
472.
473.
474.
475. int main(int argc,char *argv[])
476.
477.
478.
479.
480.
481.
482.
483.         sem_id = semget(ftok("/", 'a'), 1, 0666|IPC_CREAT);
484.
485.
486.
487.
488.
489.
490.
491.
492.
493.
```


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'.

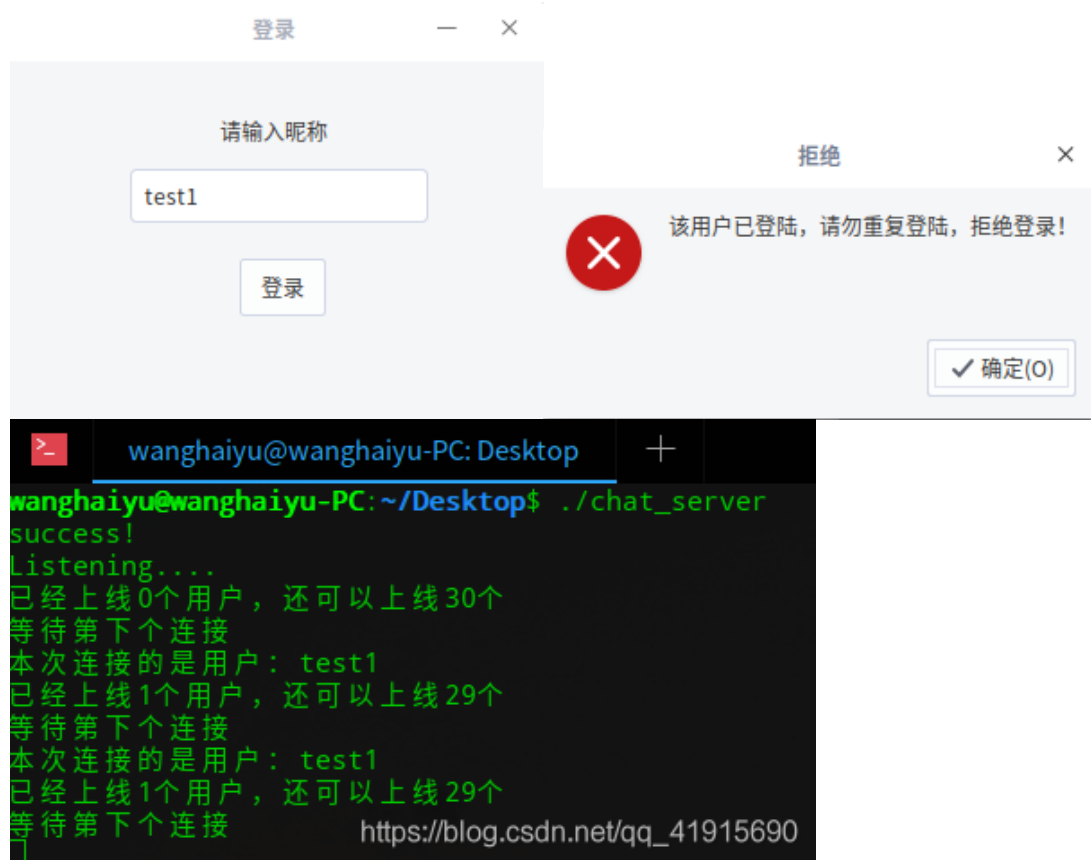
```
wanghaiyu@wanghaiyu-PC: ~/Desktop$ ./chat_server
success!
Listening....
已经上线0个用户，还可以上线30个
等待第下个连接
```

运行客户端：

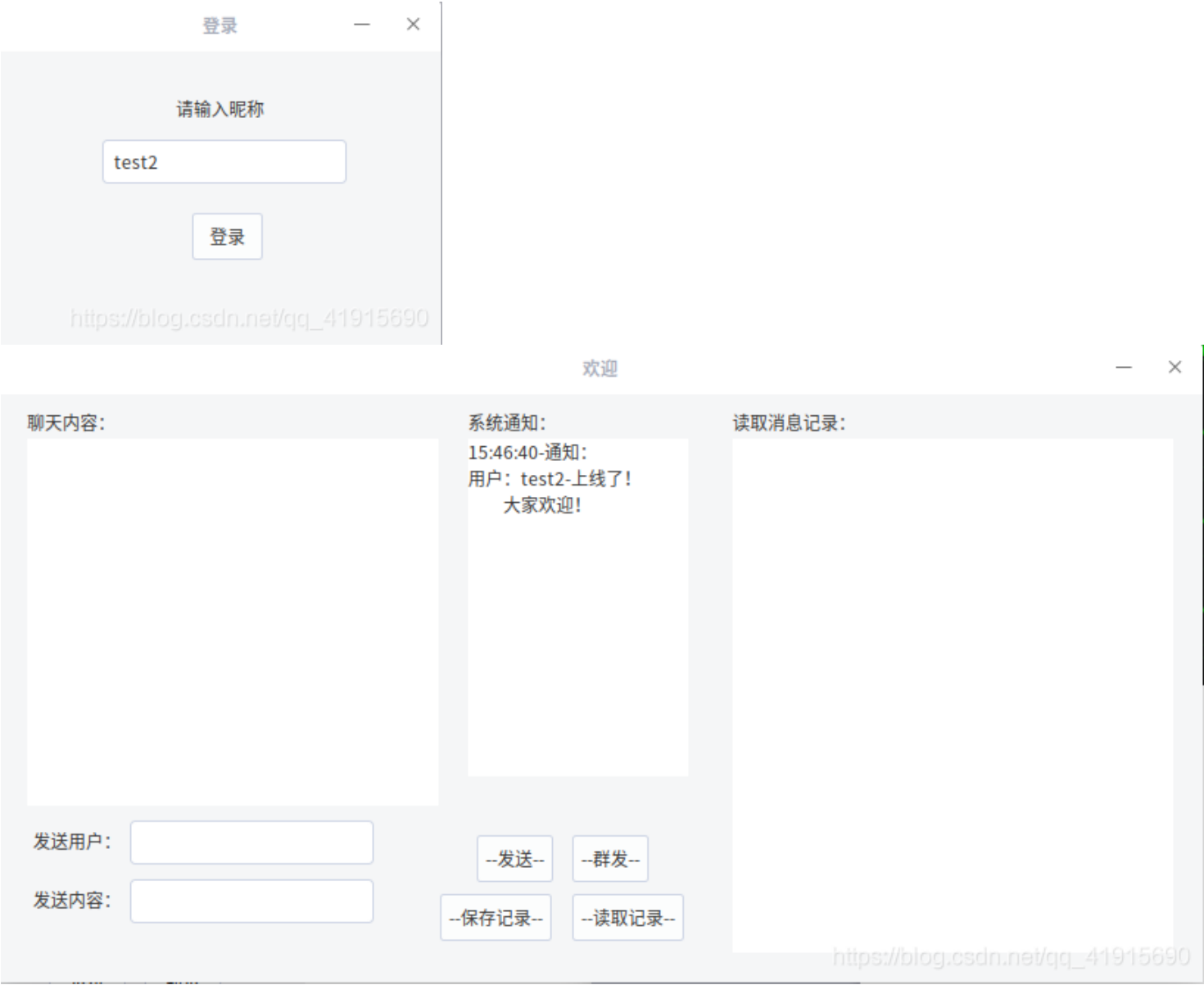




运行第二个客户端时若用户已登录:



登录第二个客户端:





```
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

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 plus sign on the right. The terminal text is green on a black background. It shows the execution of './chat_server' resulting in 'success!' and 'Listening....'. It then shows three consecutive connections from users 'test1', 'test1', and 'test2', with the number of active users increasing from 0 to 2. The window has a small square icon in the bottom left corner.

```
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

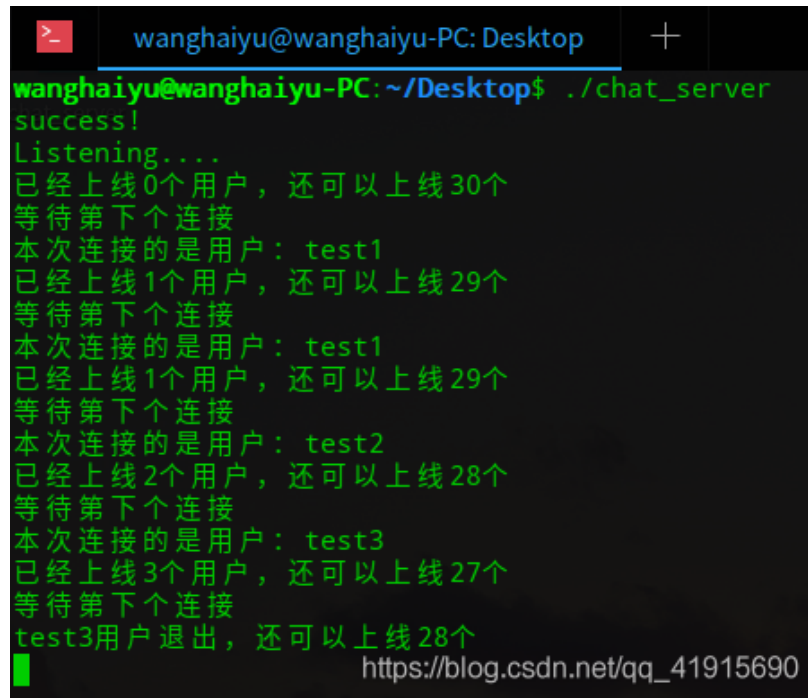


登录第三个客户端后使用第三个客户端群发消息:



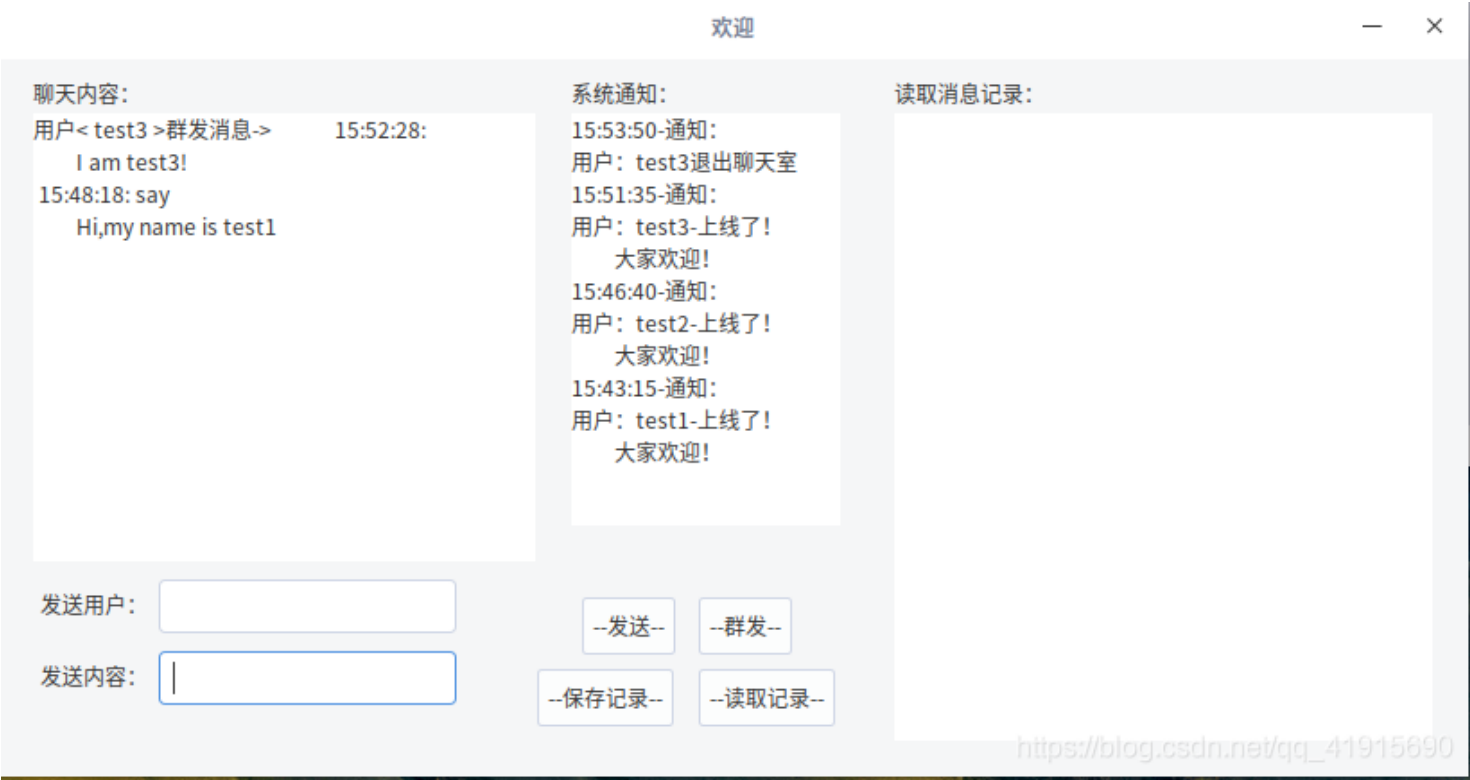




A terminal window titled 'wanghaiyu@wanghaiyu-PC: Desktop' with a red icon and a '+' button. The terminal shows the execution of './chat_server' which outputs 'success!' and 'Listening....'. It then shows three successful connections from users 'test1', 'test2', and 'test3', each increasing the online user count by one. The current state shows 3 online users and 27 more can connect. The prompt 'test3用户退出, 还可以上线 28个' is visible, followed by a green cursor.

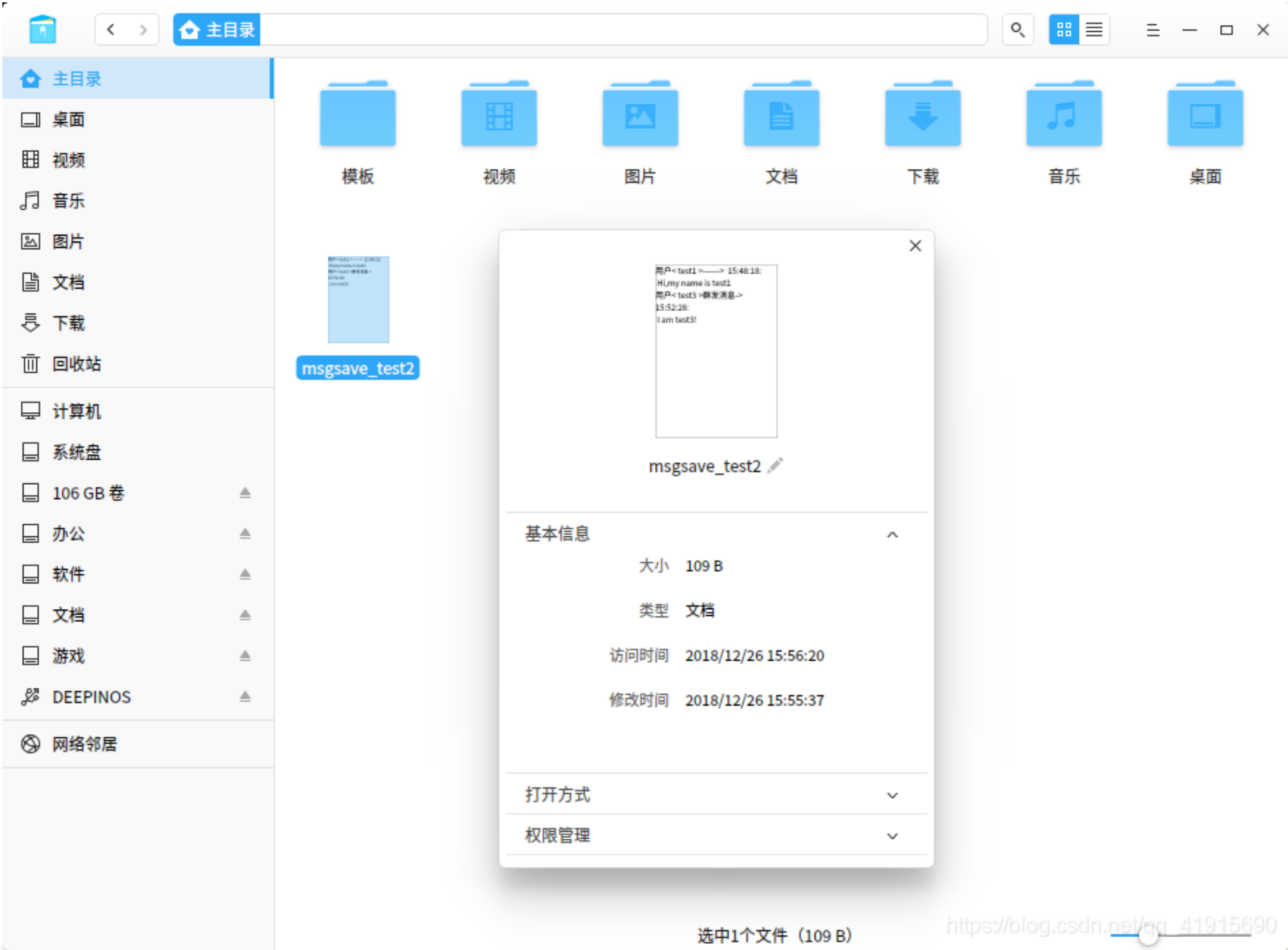
```
wanghaiyu@wanghaiyu-PC: Desktop
wanghaiyu@wanghaiyu-PC: ~/Desktop$ ./chat_server
success!
Listening....
已经上线0个用户, 还可以上线30个
等待第下个连接
本次连接的是用户: test1
已经上线1个用户, 还可以上线29个
等待第下个连接
本次连接的是用户: test1
已经上线1个用户, 还可以上线29个
等待第下个连接
本次连接的是用户: test2
已经上线2个用户, 还可以上线28个
等待第下个连接
本次连接的是用户: test3
已经上线3个用户, 还可以上线27个
等待第下个连接
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 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', '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'. The prompt is now 'wanghaiyu@wanghaiyu-PC: ~/Desktop\$'. A watermark 'https://blog.csdn.net/qq_41915690' is visible at the bottom right of the terminal 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$
```