



计算机网络实验报告

实验三：Chat实验

数据科学与计算机学院 17大数据与人工智能

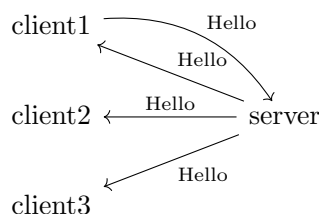
17341015 陈鸿峥

一、实验目的

掌握套接字的多线程编程方法，实现多人聊天。

二、实验介绍

利用客户/服务器(Client/Server或CS)模式实现一个多人聊天（群聊）程序，其功能是每个客户发送给服务器的消息都会传送给所有的客户端。



三、参考资料

- Introduction to Multi-Threaded Programming (Linux), <https://www.linuxjournal.com/article/3138>

四、实验环境

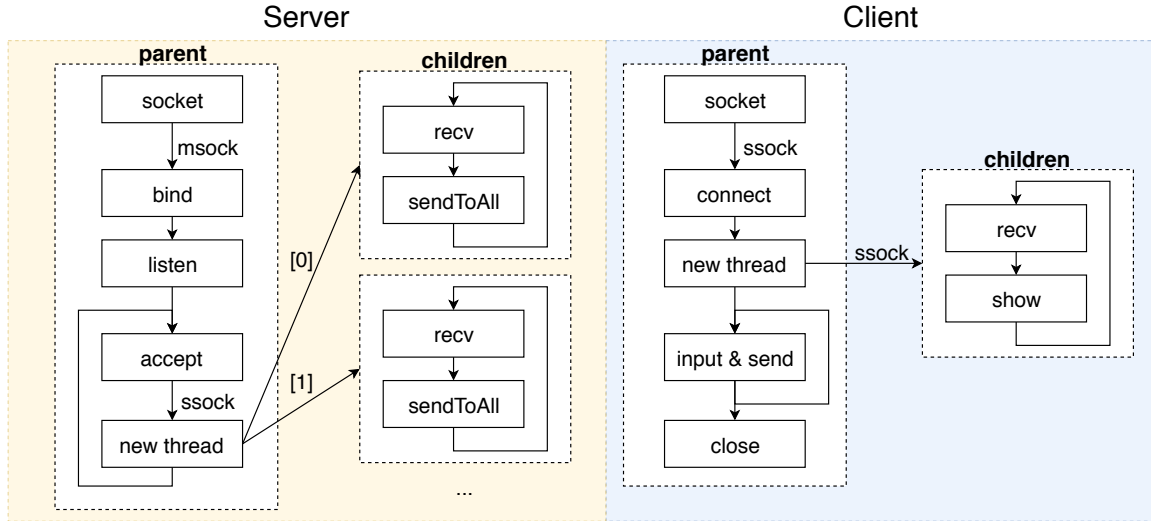
本机为Ubuntu 18.04 (LTS) + gcc 7.3.0

五、注意事项

- 依步骤完成以下实验内容，尽量多完成一些步骤
- 把出现的问题、解决方法和一些思考和体会写在实验体会部分
- 对典型的运行情况的客户和服务器控制台进行截屏并放在实验结果部分
- 截屏用按键(Ctrl+Alt+PrintScreen)单独截取控制台窗口。截屏应尽量windows绘图程序缩小尺寸(能看清就行)后粘入
- 端口号采用50500
- 字符串可以采用函数scanf或gets输入

六、实验内容

先阅读课件“套接字并发编程.PDF”，重点是读懂课件中“chat并发编程(服务器)”和“chat并发编程(客户端)”的流程图，然后完成下面步骤（截屏要同时显示服务器和至少两个客户端）。



1. 多人聊天程序

编写多人聊天程序，要求客户端和服务端都采用多线程方式进行编程。每个客户端都采用TCP协议连接服务器并保持连接。服务器同时与所有客户端建立和保持连接。每个客户端输入的消息都会通过服务器转发给所有客户。

Linux环境下线程调用需要引入头文件<pthread.h>，同时添加编译指令-pthread。创建线程为pthread_create，退出线程为pthread_exit，线程等待为pthread_join。

服务器端程序：（注意这里采用命令行参数选择不同的问题编号，如本题运行程序传./server 1）

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <sys/socket.h>
4  #include <netinet/in.h>
5  #include <arpa/inet.h>
6  #include <sys/types.h>
7  #include <unistd.h>
8  #include <string.h>
9  #include <error.h>
10 #include <time.h>
11 #include <pthread.h> // multi-thread
12
13 #define BUF_LEN 2000
14 #define MAX_CLIENT_NUM 10
15
16 void generateMsg(char* buffer){
17     char buf[BUF_LEN+1];
18     printf("消息: %s\n", buffer);
19     sprintf(buf, "消息: %s\n", buffer);

```

```
20     strcpy(buffer,buf);
21 }
22
23 void sendToAll(int* ssock, char* buf){
24     for (int i = 0; i < MAX_CLIENT_NUM; ++i)
25         if (ssock[i] != -1){
26             int cc = send(ssock[i], buf, strlen(buf), 0);
27         }
28 }
29
30 int findEmptySSock(int* ssock)
31 {
32     for (int i = 0; i < MAX_CLIENT_NUM; ++i)
33         if (ssock[i] == -1)
34             return i;
35     perror("No ssocks available!");
36     abort();
37     return -1;
38 }
39
40 typedef struct shared_data
41 {
42     int index;
43     int* ssock; // used for communication with each other
44     int mode;
45     struct in_addr client_addr;
46     int port;
47 } shared_data;
48
49 void* serve(void* arg);
50
51 int main(int argc, char *argv[])
52 {
53     struct sockaddr_in fsin;           /* the from address of a client */
54     int msock;                         /* master socket */
55     int ssock[MAX_CLIENT_NUM];        /* slaver sockets */
56     char *service = "50500";          /* port number */
57     struct sockaddr_in sin;            /* an Internet endpoint address */
58     int addlen;                        /* from-address length */
59
60     // create socket
61     msock = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP); // master sock
62
63     memset(&sin, '\0', sizeof(sin));
64     sin.sin_family = AF_INET;
```

```

65     sin.sin_addr.s_addr = INADDR_ANY;
66     sin.sin_port = htons((u_short)atoi(service));
67     bind(msock, (struct sockaddr *)&sin, sizeof(sin));
68
69     listen(msock, 5); // create queue
70
71     for (int i = 0; i < MAX_CLIENT_NUM; ++i)
72         ssock[i] = -1;
73
74     printf("服务器已启动! \n\n");
75
76     pthread_t pthid[MAX_CLIENT_NUM];
77
78     shared_data sdata;
79     while(1) {
80         addlen = sizeof(struct sockaddr);
81         // 如果在连接请求队列中有连接请求, 则接受连接请求并建立连接, 返回该连接的套接字
82         // 否则, 本语句被阻塞直到队列非空。fsin包含客户端IP地址和端口号
83         int index = findEmptySSock(ssock);
84         ssock[index] = accept(msock, (struct sockaddr *)&fsin, &addlen); // slaver
85         // sock
86
87         sdata.index = index;
88         sdata.ssock = ssock;
89         if (argc > 1)
90             sdata.mode = atoi(argv[1]);
91         else
92             sdata.mode = 1;
93         sdata.client_addr = fsin.sin_addr;
94         sdata.port = fsin.sin_port;
95         // pthread_create(&thrd1, NULL, (void *)&thread_function, (void *) &
96         // some_argument);
97         pthread_create(&pthid[index], NULL, serve, (void *)& sdata); // multi-thread
98     }
99     for (int i = 0; i < MAX_CLIENT_NUM; ++i)
100         pthread_join(pthid[i], NULL);
101     close(msock);
102     return 0;
103 }
104
105 void* serve(void* arg)
106 {
107     char buf[BUF_LEN+1];
108     shared_data* sdata = (shared_data*) arg;
109     int index = sdata->index;

```

```

108     int* ssock = sdata->ssock;
109     struct in_addr client_addr = sdata->client_addr;
110     int port = sdata->port;
111     if (sdata->mode >= 3){
112         generateEnterMsg(buf, index, (unsigned char*) &client_addr, port);
113         sendToAll(ssock,buf);
114     }
115     while (1){
116         // 返回值:(>0)接收到的字节数,(=0)对方已关闭,<0)连接出错
117         int cc = recv(ssock[index], buf, BUF_LEN, 0);
118         if (cc <= 0){ // error or closed
119             if (sdata->mode >= 4){
120                 generateExitMsg(buf,ssock,index);
121                 sendToAll(ssock,buf);
122             }
123             ssock[index] = -1; // set null
124             break;
125         }
126         else if (cc > 0) {
127             buf[cc] = '\0';
128
129             switch (sdata->mode){
130                 case 1: generateMsg(buf);break;
131                 case 2: // problem 2
132                 case 3: // problem 3
133                 case 4: // problem 4
134                 default: generateMsg2(buf, (unsigned char *) &client_addr, port);
135                             ↪ break;
136             }
137             sendToAll(ssock,buf);
138         }
139     }
140     close(ssock[index]);
141     pthread_exit(0);
142 }

```

客户端程序:

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <sys/types.h>
4 #include <sys/socket.h>
5 #include <netinet/in.h>
6 #include <arpa/inet.h>
7 #include <unistd.h>
8 #include <string.h>

```

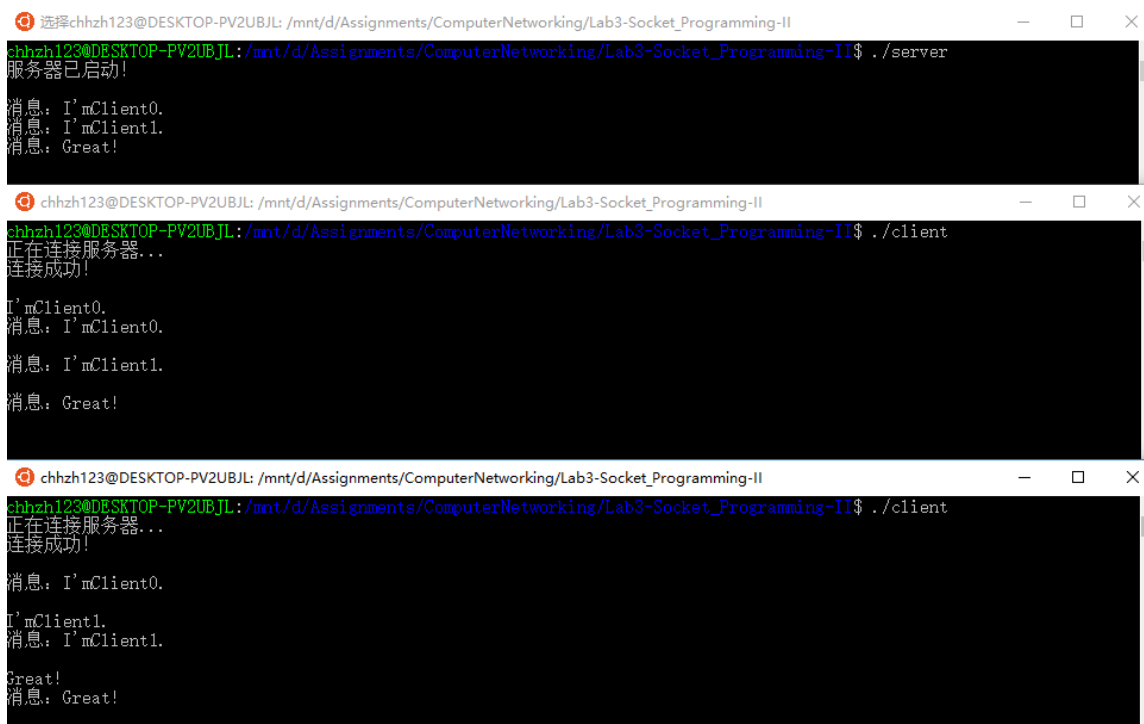
```
9  #include <error.h>
10 #include <pthread.h>
11
12 #define BUF_LEN 2000
13
14 void* receive(void* arg);
15
16 int main(int argc, char *argv[])
17 {
18     char    *host = "127.0.0.1";    /* server IP to connect    */
19     char    *service = "50520";    /* server port to connect */
20     struct sockaddr_in sin;    /* an Internet endpoint address */
21     char    buf[BUF_LEN+1];    /* buffer for one line of text */
22     int      sock;    /* socket descriptor    */
23     int      cc;    /* recv character count    */
24
25     // create socket
26     sock = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
27
28     printf("正在连接服务器...\n");
29     memset(&sin, 0, sizeof(sin));
30     sin.sin_family = AF_INET;
31     sin.sin_addr.s_addr = inet_addr(host);
32     sin.sin_port = htons((u_short)atoi(service));
33     int ret = connect(sock, (struct sockaddr *)&sin, sizeof(sin));
34     if (ret == 0)
35         printf("连接成功! \n\n");
36     else {
37         perror("Error: 连接失败! \n");
38         abort();
39     }
40
41     pthread_t pt;
42     pthread_create(&pt, NULL, receive, &sock); // multi-thread
43
44     while (1){
45         scanf("%s", buf);
46         if (strcmp(buf, "exit") == 0)
47             break;
48         // 返回值为实际发送的字节数, 出错或对方关闭时返回SOCKET_ERROR。
49         cc = send(sock, buf, strlen(buf), 0);
50         if (cc <= 0){
51             perror("Error: Server!\n");
52             return 0;
53         }
54     }
```

```
54     }
55
56     close(sock);
57
58     printf("按回车键继续...\n");
59     getchar();
60     return 0;
61 }
62
63 void* receive(void* arg)
64 {
65     char buf[BUF_LEN+1];
66     int* sock = (int*) arg;
67     while (1){
68         int cc = recv(*sock, buf, BUF_LEN, 0);
69         if (cc <= 0){
70             perror("Error: Server!\n");
71             abort();
72             break;
73         }
74         buf[cc] = '\0';
75         printf("%s\n", buf);
76     }
77     pthread_exit(0);
78 }
```

在本实验中我还编写了Makefile文件，方便自动编译。

```
1 CC = gcc
2 FLAGS = -pthread
3 ALL = server client
4
5 all: $(ALL)
6
7 % : %.c
8     $(CC) $(FLAGS) $< -o $@
9
10 .PHONY : clean
11
12 clean :
13     -rm -f *.o $(ALL)
```

运行截屏：（上面为服务器端，下面两个为客户端）



```

chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ComputerNetworking/Lab3-Socket_Programming-II$ ./server
服务器已启动!
消息: I'mClient0.
消息: I'mClient1.
消息: Great!

chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ComputerNetworking/Lab3-Socket_Programming-II$ ./client
正在连接服务器...
连接成功!
I'mClient0.
消息: I'mClient0.
消息: I'mClient1.
消息: Great!

chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ComputerNetworking/Lab3-Socket_Programming-II$ ./client
正在连接服务器...
连接成功!
消息: I'mClient0.
I'mClient1.
消息: I'mClient1.
Great!
消息: Great!

```

2. 添加信息后转发

服务器程序转发某个客户端发来的消息时都在消息前面加上该客户端的IP地址和端口号以及服务器的当前时间。要求服务器程序把转发的消息也显示出来。

服务器程序（修改部分）：

```

1 void generateMsg2(char* buffer, unsigned char *bytes, u_short port)
2 {
3     char buf[BUF_LEN+1];
4     // inet_ntoa
5     snprintf(buf, sizeof(buf), "客户端IP地址: %d.%d.%d.%d\n",
6             bytes[0], bytes[1], bytes[2], bytes[3]);
7     printf("%s", buf);
8
9     char buf2[BUF_LEN+1];
10    sprintf(buf2, "客户端端口号: %d\n", port);
11    printf("%s", buf2);
12    strcat(buf, buf2);
13
14    time_t now; /* current time */
15    time(&now);
16    char* pts = (char *)ctime(&now);
17    printf("服务器时间: %s", pts);
18    sprintf(buf2, "服务器时间: %s", pts);
19    strcat(buf, buf2);

```



```

20
21     printf("收到信息: %s\n", buffer);
22     sprintf(buf2, "内容: %s\n", buffer);
23     strcat(buf, buf2);
24
25     printf("\n");
26     strcpy(buffer, buf);
27 }

```

运行截屏：（上面为服务器端，下面两个为客户端）

```

chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ComputerNetworking/Lab3-Socket_Programming-II$ ./server 2
服务器已启动!
客户端IP地址: 127.0.0.1
客户端端口号: 27365
服务器时间: Tue Mar 19 13:33:11 2019
收到信息: test1

客户端IP地址: 127.0.0.1
客户端端口号: 28133
服务器时间: Tue Mar 19 13:33:16 2019
收到信息: test2

chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ComputerNetworking/Lab3-Socket_Programming-II$ ./client
正在连接服务器...
连接成功!

test1
客户端IP地址: 127.0.0.1
客户端端口号: 27365
服务器时间: Tue Mar 19 13:33:11 2019
内容: test1

test2
客户端IP地址: 127.0.0.1
客户端端口号: 28133
服务器时间: Tue Mar 19 13:33:16 2019
内容: test2

chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ComputerNetworking/Lab3-Socket_Programming-II$
客户端IP地址: 127.0.0.1
客户端端口号: 27365
服务器时间: Tue Mar 19 13:33:11 2019
内容: test1

test2
客户端IP地址: 127.0.0.1
客户端端口号: 28133
服务器时间: Tue Mar 19 13:33:16 2019
内容: test2

```

3. “进入”信息发送

新客户刚连接时服务器端把enter消息（包含客户端IP地址和端口号）发送给所有客户端。

服务器程序（修改部分）：

```

1 void generateEnterMsg(char* buffer, int index, unsigned char *bytes, u_short port)
2 {
3     char buf[BUF_LEN+1];
4     sprintf(buf, "%d号客户端进入\n", index);

```

```

5     printf("%s", buf);
6
7     // inet_ntoa
8     char buf2[BUF_LEN+1];
9     snprintf(buf2, sizeof(buf2), "客户端IP地址: %d.%d.%d.%d\n",
10             bytes[0], bytes[1], bytes[2], bytes[3]);
11     printf("%s", buf2);
12     strcat(buf, buf2);
13
14     sprintf(buf2, "客户端端口号: %d\n", port);
15     printf("%s", buf2);
16     strcat(buf, buf2);
17
18     strcpy(buffer, buf);
19 }

```

运行截屏：（上面为服务器端，下面两个为客户端）

```

chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ComputerNetworking/Lab3-Socket_Programming-II
chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ComputerNetworking/Lab3-Socket_Programming-II$ ./server 3
服务器已启动!
0号客户端进入
客户端IP地址: 127.0.0.1
客户端端口号: 30949
1号客户端进入
客户端IP地址: 127.0.0.1
客户端端口号: 31205

chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ComputerNetworking/Lab3-Socket_Programming-II
chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ComputerNetworking/Lab3-Socket_Programming-II$ ./client
正在连接服务器...
连接成功!
0号客户端进入
客户端IP地址: 127.0.0.1
客户端端口号: 30949
1号客户端进入
客户端IP地址: 127.0.0.1
客户端端口号: 31205

chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ComputerNetworking/Lab3-Socket_Programming-II
chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ComputerNetworking/Lab3-Socket_Programming-II$ ./client
正在连接服务器...
连接成功!
1号客户端进入
客户端IP地址: 127.0.0.1
客户端端口号: 31205

```

4. “退出”信息发送

客户端输入exit时退出客户端程序（正常退出），或者客户端直接关闭窗口退出（异常退

出)，服务器都会把该客户leave的消息广播给所有客户。

由于客户端进入时已经发送其客户端IP地址及端口号，故这里退出时不再重复发送。

服务器程序（修改部分）：

```
1 void generateExitMsg(char* buf, int* ssock, int index)
2 {
3     sprintf(buf, "客户端%d离开!\n", index);
4     printf("%s\n", buf);
5 }
```

运行截屏：（上面为服务器端，下面两个为客户端，其中第一个正常退出，第二个异常退出）

```
chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ComputerNetworking/Lab3-Socket_Programming-II$ ./server 4
服务器已启动!

0号客户端进入
客户端IP地址: 127.0.0.1
客户端端口号: 34533

1号客户端进入
客户端IP地址: 127.0.0.1
客户端端口号: 34789

客户端IP地址: 127.0.0.1
客户端端口号: 34533
服务器时间: Tue Mar 19 13:38:02 2019
收到信息: saysomething

客户端IP地址: 127.0.0.1
客户端端口号: 34789
服务器时间: Tue Mar 19 13:38:06 2019
收到信息: hihihi

客户端0离开!
客户端1离开!
```

```
chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ComputerNetworking/Lab3-Socket_Programming-II$ ./client
正在连接服务器...
连接成功!

0号客户端进入
客户端IP地址: 127.0.0.1
客户端端口号: 34533

1号客户端进入
客户端IP地址: 127.0.0.1
客户端端口号: 34789

saysomething
客户端IP地址: 127.0.0.1
客户端端口号: 34533
服务器时间: Tue Mar 19 13:38:02 2019
内容: saysomething

客户端IP地址: 127.0.0.1
客户端端口号: 34789
服务器时间: Tue Mar 19 13:38:06 2019
内容: hihihi

exit
按回车键继续...
chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ComputerNetworking/Lab3-Socket_Programming-II$
```

```
chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ComputerNetworking/Lab3-Socket_Programming-II$ ./client
正在连接服务器...
连接成功!

1号客户端进入
客户端IP地址: 127.0.0.1
客户端端口号: 34789

客户端IP地址: 127.0.0.1
客户端端口号: 34533
服务器时间: Tue Mar 19 13:38:02 2019
内容: saysomething

hihihi
客户端IP地址: 127.0.0.1
客户端端口号: 34789
服务器时间: Tue Mar 19 13:38:06 2019
内容: hihihi

客户端0离开!

^C
chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ComputerNetworking/Lab3-Socket_Programming-II$
```

5. 与老师服务器连接

运行客户端程序测试与老师的服务器程序的连接（172.18.187.9:50500）。

运行截屏（客户端）：

```

chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ComputerNetworking/Lab3-Socket_Programming-II
chhzh123@DESKTOP-PV2UBJL:/mnt/d/Assignments/ComputerNetworking/Lab3-Socket_Programming-II$ ./client
正在连接服务器...
连接成功!

ip: 172.18.166.63 port: 39909
time: Tue Mar 19 13:42:58 2019
message: Enter!

>>
welllllllllll
ip: 172.18.166.63 port: 39909
time: Tue Mar 19 13:43:17 2019
message: welllllllllll

>>
interesting
ip: 172.18.166.63 port: 39909
time: Tue Mar 19 13:43:29 2019
message: interesting

>>
hahaha
ip: 172.18.166.63 port: 39909
time: Tue Mar 19 13:44:00 2019
message: hahaha

>>

```

6. 与同学互测

与同学的程序进行相互测试，一个人可以与多人测试，截屏选择其中一个。

同学的学号姓名（可以多人）：17341111刘学海

作为服务器运行截屏：（注意本程序是在个人服务器上运行；并且由于互测了多次，换了多个网络，所以截图的IP地址会经常改变）

```

0号客户端进入
客户端ip地址：172.18.166.63
客户端端口号：64996

客户端ip地址：172.18.166.63
客户端端口号：64996
服务器时间：Tue Mar 19 13:14:57 2019
收到信息：emmm

客户端ip地址：172.18.166.63
客户端端口号：64996
服务器时间：Tue Mar 19 13:15:01 2019
收到信息：great!

客户端0离开！

1号客户端进入
客户端ip地址：172.26.127.25
客户端端口号：39629

客户端ip地址：172.26.127.25
客户端端口号：39629
服务器时间：Tue Mar 19 13:57:42 2019
收到信息：oyaooya

客户端ip地址：172.26.127.25
客户端端口号：39629
服务器时间：Tue Mar 19 13:57:48 2019
收到信息：aliealie

客户端1离开！

```

作为客户端运行截屏：

[illegible]

七、 完成情况

是否完成以下步骤? (✓完成 ✕未做)

1. ☒ 2. ☒ 3. ☒
4. ☒ 5. ☒ 6. ☒

八、实验体会

本次实验非常有趣，学会了并发编程的一些基础理论，也明白如何缓解计算机网络中出现的堵塞延迟等问题。

由于有了第二次实验的基础，所以这个实验添加多线程元素并不是很复杂。本次实验再次印证了Linux环境下的优越性，调用多线程库进行网络编程非常的方便。

自己与自己测试时基本没有出现什么问题，与同学互测却出现了很多问题。首先是互相之间是否有办法ping通，往往由于各自电脑防火墙的缘故，导致端口并不对外开放。查找资料并设置好防火墙就花费了很长时间。而且由于是我的Linux系统是Windows子系统，系统内核可能存在一些bug，与同学的Windows客户端互连时会出现接不通的问题，这一点还没找到原因，最后是将我的程序放上纯Ubuntu系统的服务器才得以解决。还有一点则是Linux和Windows环境下命令行的编码问题，常常会导致中文无法正常显示。（Windows默认GBK，Linux默认UTF-8）

总的来说，本次实验收获良多，更加深入理解了多线程编程及套接字的原理。