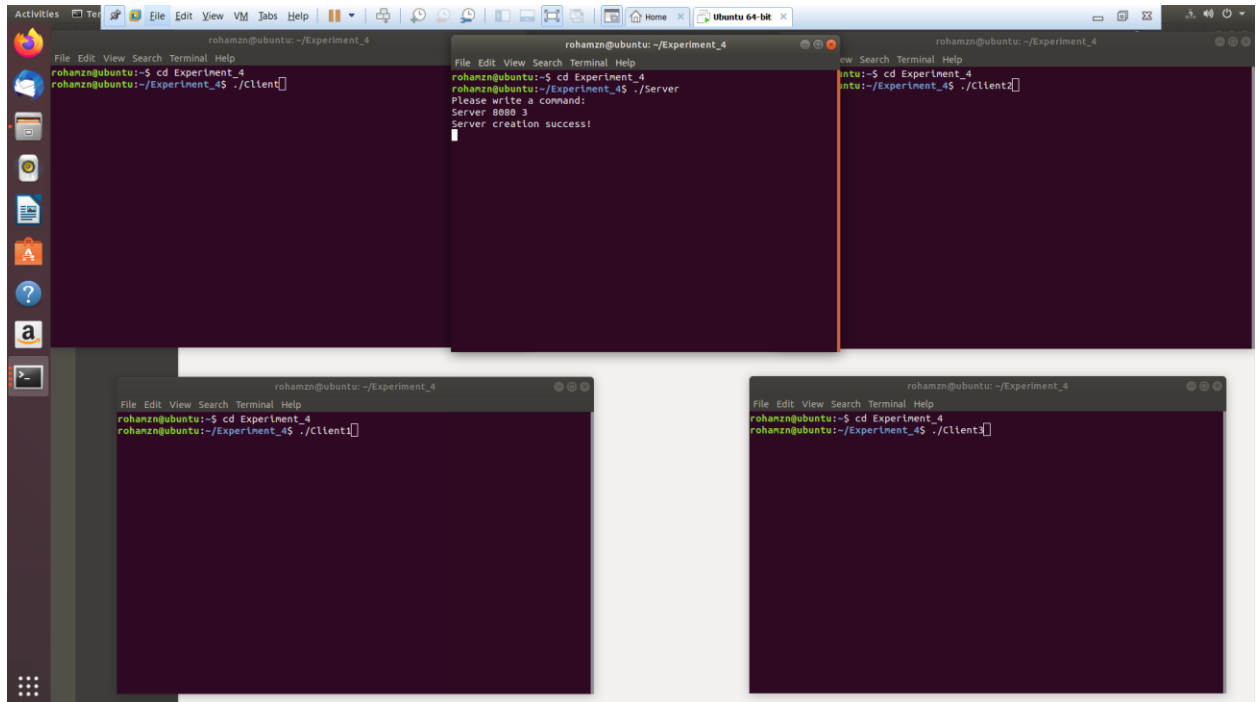


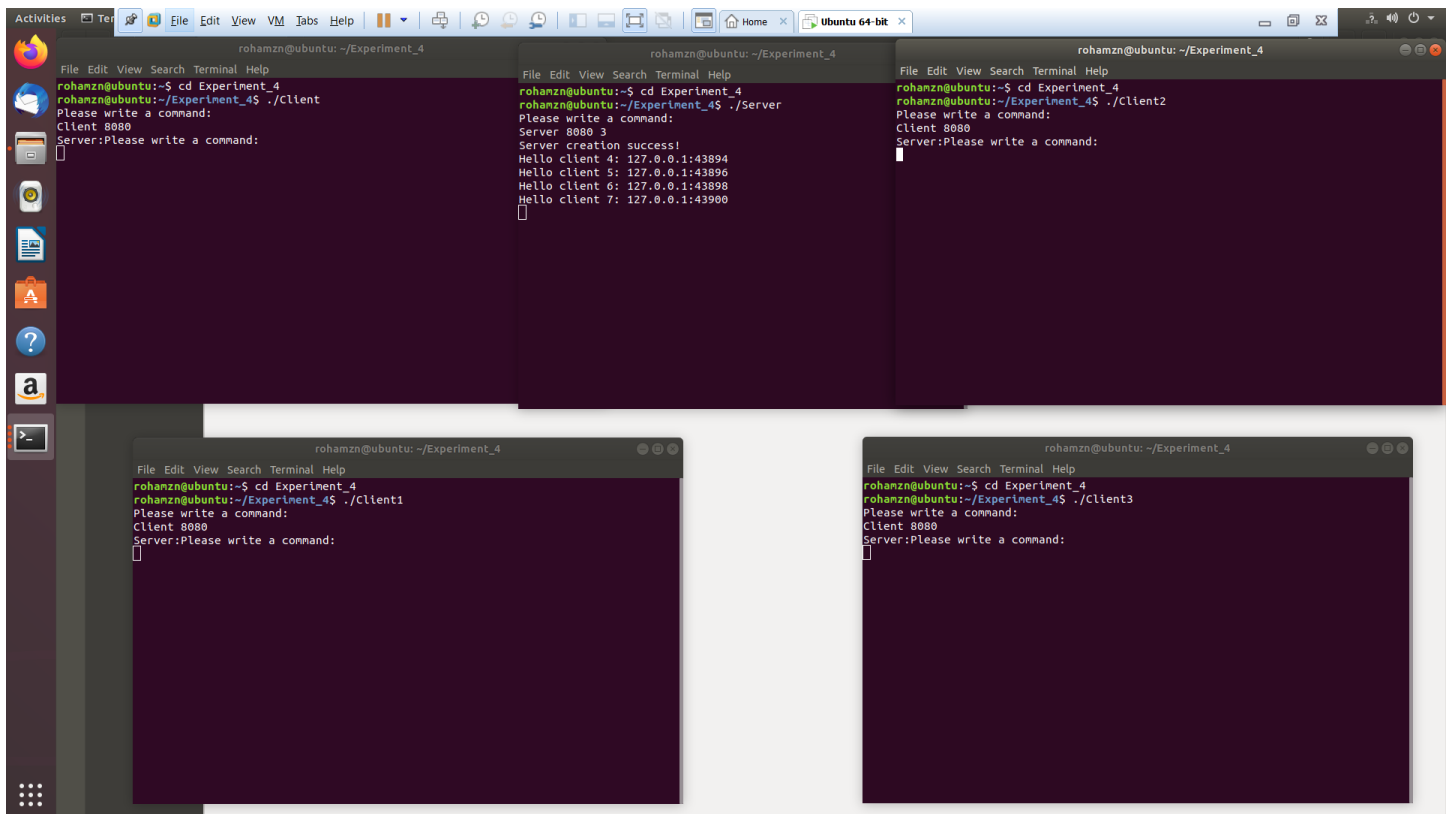
تمرین چهارم پارت 2

رهام زنده دل 9731088



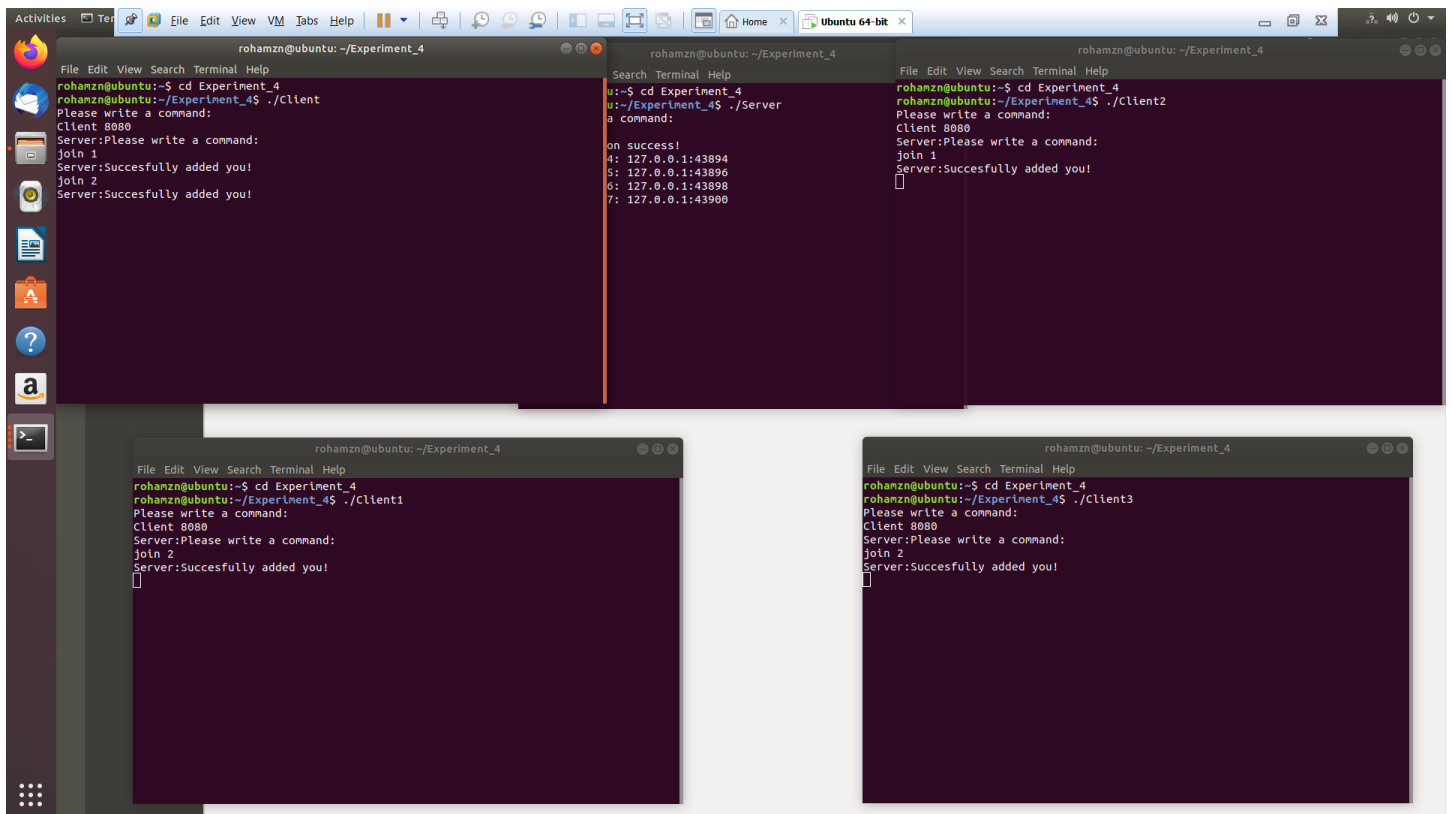
برنامه ای که نوشته ام قابلیت وصل به هر چندتا کلاینت را دارد و همه ی آنها می توانند همزمان و بدون بلاک شدن به سرور پیام بدهند و سرور برای بقیه کلاینت ها پیام ها را بفرستد.

در ابتدا در یکی از کنسول ها سرور را اجرا می کنیم و دستور 3 8080 Server را می دهیم. پورت سرور است و 3 تعداد گروه ها است.

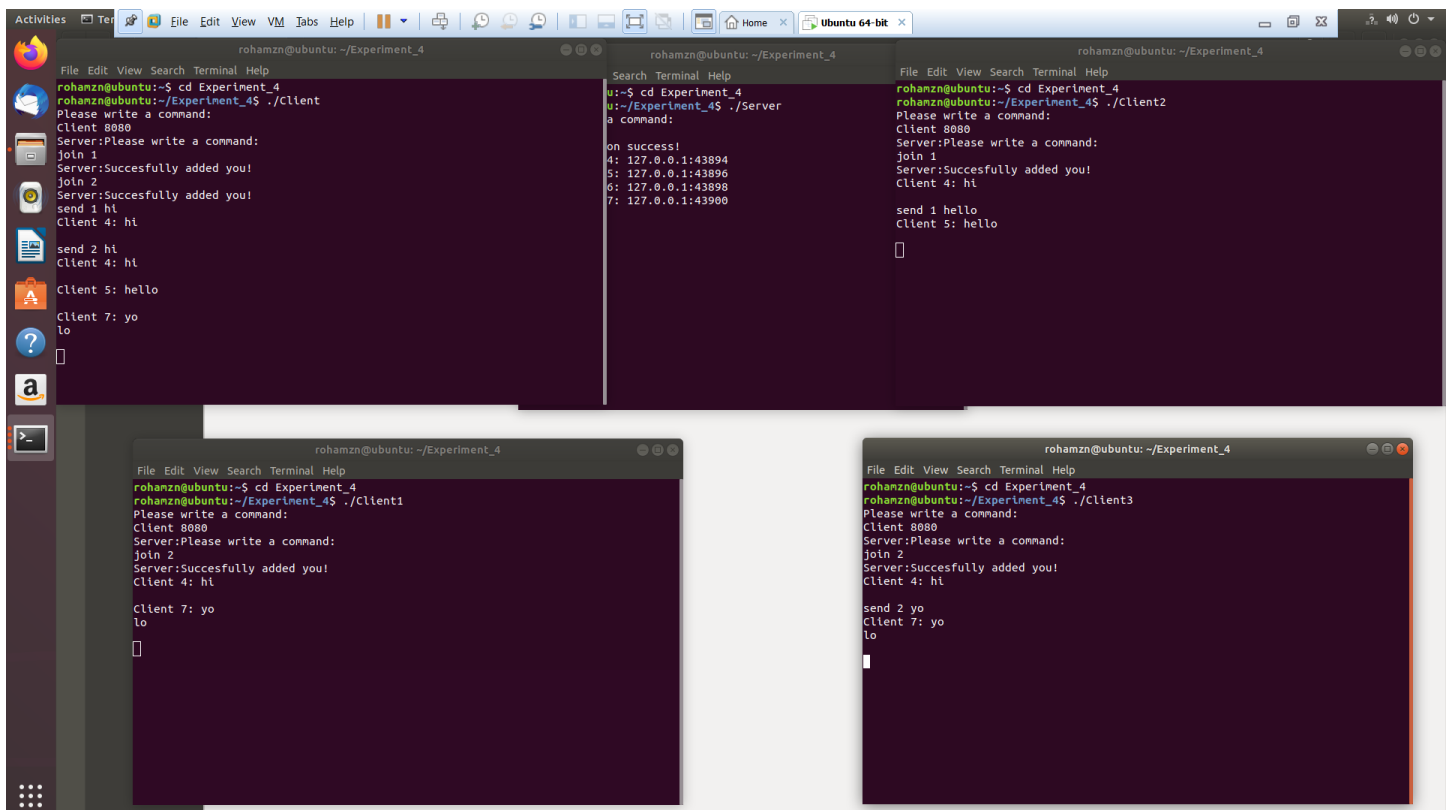


حال هر کدام از کلاینت ها را با دستور Client 8080 به سرور وصل می کنیم و از طرف سرور پیامی برای آنها می آید که دستوری را برای سرور بفرستند. همچنین در کنسول سرور مشخصات هر کلاینت وصل شده نشان داده می شود. هر کدام از کلاینت ها دستورات زیر را می توانند به سرور بفرستند:

join [groupID] – send [groupID][message] – leave [groupID] – quit



کلاينت های بالا به گروه 1 و کلاينت های پایین به گروه 2 وصل اضافه می شوند. کلاينت بالا سمت چپ به دو گروه 1 و 2 اضافه شده است.



حال کلاينت بالا سمت چپ به گروه یک و گروه دو پیام می فرستد. همچنین بقیه نیز پیام می فرستند.

```
rohamzn@ubuntu: ~/Experiment_4
File Edit View Search Terminal Help
rohamzn@ubuntu:~/Experiment_4$ cd Experiment_4
rohamzn@ubuntu:~/Experiment_4$ ./Client
Please write a command:
Client 8080
Server:Please write a command:
Join 1
Server:Successfully added you!
Join 2
Server:Successfully added you!
send 1 hi
Client 4: hi
send 2 hi
Client 4: hi
Client 5: hello
Client 7: yo
lo
leave 1
You left group 1.

rohamzn@ubuntu:~/Experiment_4
File Edit View Search Terminal Help
rohamzn@ubuntu:~/Experiment_4$ cd Experiment_4
rohamzn@ubuntu:~/Experiment_4$ ./Server
Please write a command:
on success!
4: 127.0.0.1:43894
5: 127.0.0.1:43896
6: 127.0.0.1:43898
7: 127.0.0.1:43900

rohamzn@ubuntu:~/Experiment_4
File Edit View Search Terminal Help
rohamzn@ubuntu:~/Experiment_4$ cd Experiment_4
rohamzn@ubuntu:~/Experiment_4$ ./Client2
Please write a command:
Client 8080
Server:Please write a command:
Join 1
Server:Successfully added you!
Client 4: hi
send 1 hello
Client 5: hello
Client 4 left the group!
```

حال کلاینت بالا سمت چپ از گروه 1 می رود و برای بقیه اعضای گروه 1 خبر رفتن کلاینت 4 می رود.

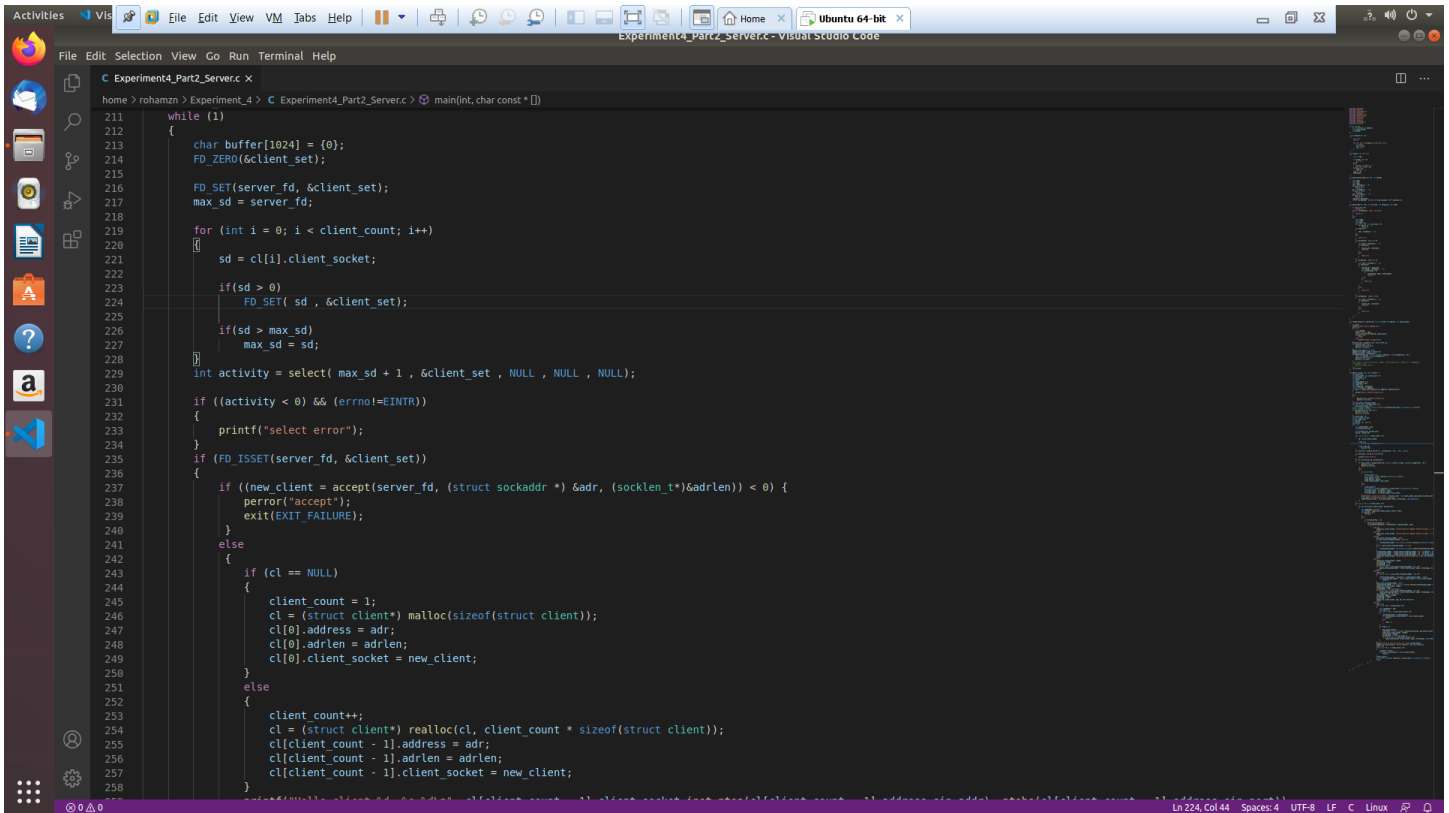
```
rohamzn@ubuntu: ~/Experiment_4
File Edit View Search Terminal Help
rohamzn@ubuntu:~/Experiment_4$ ./Client
Please write a command:
Client 8080
Server:Please write a command:
Join 1
Server:Successfully added you!
Join 2
Server:Successfully added you!
send 1 hi
Client 4: hi
send 2 hi
Client 4: hi
Client 5: hello
Client 7: yo
lo
leave 1
You left group 1.
quit
Server:Goodbye!
rohamzn@ubuntu:~/Experiment_4$

rohamzn@ubuntu:~/Experiment_4
File Edit View Search Terminal Help
rohamzn@ubuntu:~/Experiment_4$ cd Experiment_4
rohamzn@ubuntu:~/Experiment_4$ ./Server
Please write a command:
Server 8080 3
Server creation success!
Hello client 4: 127.0.0.1:43894
Hello client 5: 127.0.0.1:43896
Hello client 6: 127.0.0.1:43898
Hello client 7: 127.0.0.1:43900
Client 7 left the server.
Client 4 left the server.

rohamzn@ubuntu:~/Experiment_4
File Edit View Search Terminal Help
rohamzn@ubuntu:~/Experiment_4$ cd Experiment_4
rohamzn@ubuntu:~/Experiment_4$ ./Client2
Please write a command:
Client 8080
Server:Please write a command:
Join 2
Server:Successfully added you!
Client 4: hi
send 2 yo
Client 7: yo
lo
quit
Server:Goodbye!
rohamzn@ubuntu:~/Experiment_4$
```

با نوشتن دستور quit نیز سرور از کلاینت ها خداحافظی می کند.

برای هندل کردن کلاینت ها و سرور ها به صورتی که همدیگر را بلاک نکنند از select استفاده می کنیم.



```
211 while (1)
212 {
213     char buffer[1024] = {0};
214     FD_ZERO(&client_set);
215     FD_SET(server_fd, &client_set);
216     max_sd = server_fd;
217
218     for (int i = 0; i < client_count; i++)
219     {
220         sd = cl[i].client_socket;
221
222         if(sd > 0)
223             FD_SET( sd , &client_set);
224
225         if(sd > max_sd)
226             max_sd = sd;
227     }
228     int activity = select( max_sd + 1 , &client_set , NULL , NULL , NULL);
229
230     if ((activity < 0) && (errno!=EINTR))
231     {
232         printf("select error");
233     }
234     if (FD_ISSET(server_fd, &client_set))
235     {
236         if ((new_client = accept(server_fd, (struct sockaddr *) &adr, (socklen_t*)&adrlen)) < 0) {
237             perror("accept");
238             exit(EXIT_FAILURE);
239         }
240         else
241         {
242             if (cl == NULL)
243             {
244                 client_count = 1;
245                 cl = (struct client*) malloc(sizeof(struct client));
246                 cl[0].address = adr;
247                 cl[0].adrlen = adrlen;
248                 cl[0].client_socket = new_client;
249             }
250             else
251             {
252                 client_count++;
253                 cl = (struct client*) realloc(cl, client_count * sizeof(struct client));
254                 cl[client_count - 1].address = adr;
255                 cl[client_count - 1].adrlen = adrlen;
256                 cl[client_count - 1].client_socket = new_client;
257             }
258         }
259     }
```

هر کلاینت به مجموعه `client_set` که نوعی `file descriptor` است با هر چرخش حلقه اضافه می شود و برای این `client_set` یک `listener` تعریف می کنیم که همان `select` است.

حال برای گرفتن `event` از دستور `FD_ISSET` استفاده می کنیم.

`FD_ZERO` مجموعه `client_set` را `restart` می کند.

`FD_SET` نیز ساکتی را به `client_set` اضافه می کند.

بقیه کد برای هندل کردن بخش های مختلف کد است و مطالب جدیدی ندارد.