今日任务：

> 注意：在完成下面之前，老师修改了今天写的一个程序中的bug，大家从github上克隆一下新的；
>
> 主要改动了udp_connect()里：将socket_udp(); 改成了socket_create_udp(port);与之对应的，sever.c中的port改为了全局变量。
>
> 原因是：这socket_create_udp里设置了非阻塞和地址重用，我们需要这些

1. 完成客户端对应着server端的udp_accept函数的部分；
    1. 实现客户端登录，收到服务端发回的信息；

> **注意，一定要看这里**，变量在今天的基础上，做了一定的修改，加了变量；

```c
/*************************************************************************
 > File Name: client.c
 > Author: suyelu
 > Mail: suyelu@126.com
 > Created Time: Wed 08 Jul 2020 04:32:12 PM CST
 ************************************************************************/

#include "head.h"

int server_port = 0;
char server_ip[20] = {0};
int team = -1;
char name[20] = {0};
char log_msg[512] = {0};
char *conf = "./football.conf";
int sockfd = -1;

int main(int argc, char **argv) {
    int opt;
    struct LogRequest request;        //HERE
    struct LogResponse response;     //HERE
    while ((opt = getopt(argc, argv, "h:p:t:m:n:")) != -1) {
        switch (opt) {
            case 't':
                request.team = atoi(optarg); //HERE
                break;
            case 'h':
                strcpy(server_ip, optarg);
                break;
            case 'p':
                server_port = atoi(optarg);
```

```
                break;
            case 'm':
                strcpy(request.msg, optarg); //HERE
                break;
            case 'n':
                strcpy(request.name, optarg); //HERE
                break;
            default:
                fprintf(stderr, "Usage : %s [-hptmn]!\n", argv[0]);
                exit(1);
        }
    }


    if (!server_port) server_port = atoi(get_conf_value(conf, "SERVERPORT"));
    if (!request.team) team = atoi(get_conf_value(conf, "TEAM")); //HERE
    if (!strlen(server_ip)) strcpy(server_ip, get_conf_value(conf, "SERVERIP"));
    if (!strlen(request.name)) strcpy(name, get_conf_value(conf, "NAME"));//HERE
    if (!strlen(request.msg)) strcpy(log_msg, get_conf_value(conf,
"LOGMSG"));//HERE


    DBG("<"GREEN"Conf Show"NONE"> : server_ip = %s, port = %d, team = %s, name =
%s\n%s",\
        server_ip, server_port, request.team ? "BLUE": "RED", request.name,
request.msg);

    struct sockaddr_in server;
    server.sin_family = AF_INET;
    server.sin_port = htons(server_port);
    server.sin_addr.s_addr = inet_addr(server_ip);

    socklen_t len = sizeof(server);

    if ((sockfd = socket_udp()) < 0) {
        perror("socket_udp()");
        exit(1);
    }

    sendto(sockfd, (void *)&request, strlen(request), 0, (struct sockaddr
*)&server, len);//HERE
    /*********************************以下为需要实现的内容**********************/
    //上一行已经发送request，接下来要等待response
    //使用select实现，等待5秒，如果5秒内还没收到数据，则判定为服务端不在线，退出程序
        /*
        实现细节（伪代码）
        fd_set rfds;
        fd_ZERO rfds;
        FD_SET sockfd  TO  rfds
        struct timeval tv
        tv.tv_sec 为5
        tv.tv_usec = 0
```

```
80
81          select blocked，判断是否有文件就绪
82
83          判断返回值为0， 没有就绪的，就退出，
84          返回这为非零，说明sockfd一定就绪，接受信息到response
85          判断response大小是否合法，type是否为1，如果不合法，或type为1，则服务端拒绝接入，答应
    response.msg
86
87          */
88
89      //调用connect连接到服务端，相当于建立“连接”
90
91      //发送一信息给服务端，查看从反应堆的，do_work是否返回
92
93      return 0;
94  }
```

2. 完成函数add_to_sub_reactor()

```
1   //file: udp_epoll.c
2
3   extern struct User *rteam;
4   extern struct User *bteam;
5   extern int repollfd, bepollfd;
6
7   void add_event_ptr(int epollfd, int fd, int events, struct User *user)  {
8       struct epoll_event ev;
9       ev.events = events;
10      ev.data.ptr = (void *)user;
11      epoll_ctl(epollfd, EPOLL_CTL_ADD, fd, &ev);
12  }
13
14  void del_event(int epollfd, int fd) {
15      epoll_ctl(epollfd, EPOLL_CTL_DEL, fd, NULL);
16  }
17  int find_sub(struct User *team) {
18      for (int i = 0; i < MAX; i++) {
19          if (!team[i].online) retrun i;
20      }
21      return -1;
22  }
23
24  void add_to_sub_reactor(struct User *user) {
25      //根据user里的team变量判断是红队还是蓝队，进而知道用户存储的数组是rteam，还是bteam
26      //find_sub(team);
27      //将user指向的用户信息存放在team[sub]中
28      //根据user->team不同，将用户加到不同的从反应堆中，使用add_event_ptr函数。注册EPOLLIN
    和 EPOLLET事件
29  }
30
```

在完成过程中，应该积极输出调试信息

触发测试点：

> 测试服务将在准备好通知各位测评

1. 成功登录教师服务端，加入聊天 100分
2. 给教师服务端成功发送一条聊天信息 100分