使用UDP完成套接字编程

1. 基本信息

姓名	学号	班级	是否编译成功	是否运行结果正确（逻辑也要正确）
庄佳强	202121331104	计算2114	是	是

如果运行结果正确，按以下方式来写：

- 结合关键代码，解释代码背后的原理（对于别人不能一眼就能明白你代码要做什么的，你要给出解释）
- 给出测试结果，并解释运行结果
- 文末，再给出完整的代码

以下是代码运行正确的报告格式：

2. 客户端函数

```
//使用到的结构体
typedef struct{
    int type;          //信息请求类型
    char username[20];//用户名
    char message[BUF_size];  //信息
}Packet;


//发送消息
void* Send_Message(void* arg) {
    //循环读取用户输入
    while (1) {
        printf("输入信息(输入q退出)：");
        scanf("%s", packet.message);
        if (strcmp(packet.message, "q") == 0)
            break;
        //构造消息并发送
        strcpy(packet.username, staticusername);//把用户名写入的packet的数据包中
        packet.type = 2;//发送请求
        if (sendto(sock, &packet, sizeof(packet), 0, (struct
sockaddr*)&serv_addr,
        sizeof(serv_addr)) == -1)//发送
        error_message("发送失败");
    }
    //结束聊天
    //发送登出请求
    packet.type = 2;//登出请求
    sprintf(packet.message, "%s 离开了聊天", packet.username);
        if (sendto(sock, &packet, sizeof(packet), 0, (struct
sockaddr*)&serv_addr,
        sizeof(serv_addr)) == -1)//发送
```

```
        error_message("发送失败");
    pthread_exit(NULL);
}


void* Recv_Message(void* arg){
    socklen_t serv_addr_size;  //一种数据类型，它其实和int差不多，记录长度
    //循环接收消息
    while (1) {
        serv_addr_size = sizeof(serv_addr);
        //接收
        if (recvfrom(sock, &packet, sizeof(packet), 0, (struct
sockaddr*)&serv_addr,
            &serv_addr_size) == -1)
            error_message("recvfrom error");
        //处理消息类型
        switch (packet.type)
        {
        case 1://登录请求
            printf("\n%s加入聊天", packet.username);//打到字符串中
            break;
        case 2://聊天信息
            printf("\n%s : %s", packet.username, packet.message);
            break;
    }
}
    pthread_exit(NULL);//关闭线程
}
```

Send_Message函数:整体来说一个线程函数，用于从标准输入中读取用户输入的信息并发送给服务器。函数会循环读取用户输入，直到输入了字符 "q"，然后发送一个登出请求告诉服务器该用户已经离开了聊天。

Recv_Message函数:接收从服务器发送过来的消息并打印到标准输出中。

在收到服务端发送的包后，解析packet中的type的类型，相对应输出内容。

3. 服务端函数

//服务端的send_Message函数和Recv_Message函数和客户端的相同

```
服务端比较多了是需要进行Bind连接。
if (bind(sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr)) == -1)
        error_message("bind error");

void* Send_Message(void* arg) {
    //循环读取用户输入
    while (1) {
        printf("输入信息(输入q退出)：");
        scanf("%s", packet.message);
        if (strcmp(packet.message, "q") == 0)
            break;
```

```
        //构造消息并发送
        strcpy(packet.username, staticusername);//把用户名写入的packet的数据包中
        packet.type = 2;//发送请求
        if (sendto(sock, &packet, sizeof(packet), 0, (struct
sockaddr*)&serv_addr,
        sizeof(serv_addr)) == -1)//发送
        error_message("发送失败");
    }
    //结束聊天
    //发送登出请求
    packet.type = 2;//登出请求
    sprintf(packet.message, "%s 离开了聊天", packet.username);
        if (sendto(sock, &packet, sizeof(packet), 0, (struct
sockaddr*)&serv_addr,
        sizeof(serv_addr)) == -1)//发送
        error_message("发送失败");
    pthread_exit(NULL);
}


void* Recv_Message(void* arg){
    socklen_t serv_addr_size;  //一种数据类型，它其实和int差不多，记录长度
    //循环接收消息
    while (1) {
        serv_addr_size = sizeof(serv_addr);
        //接收
        if (recvfrom(sock, &packet, sizeof(packet), 0, (struct
sockaddr*)&serv_addr,
            &serv_addr_size) == -1)
            error_message("recvfrom error");
        //处理消息类型
        switch (packet.type)
        {
        case 1://登录请求
            printf("\n%s加入聊天", packet.username);//打到字符串中
            break;
        case 2://聊天信息
            printf("\n%s : %s", packet.username, packet.message);
            break;
    }
}
    pthread_exit(NULL);//关闭线程
}
```

4. 运行结果及分析

服务端

```
./server 30104
输入名称：server
****server加入聊天****
clien加入聊天
输入信息(输入q退出)：你好
```

```
clien : 你好
clien : 6666
输入信息(输入q退出)：6666
输入信息(输入q退出)：
clien : 7777
7777
输入信息(输入q退出)：
......
q
u202121331104@jmu-cs:~/net_exp$
```

这里有一个小bug,当server未开启时相当与未连接两个端，所有server加入聊天这个回复没有发送到客户端，就发送在server服务端显示，之后当客户端连接上就正常了。

客户端

```
./client 127.0.0.1 30104
输入名称：clien
server : 你好
输入信息(输入q退出)：你好
输入信息(输入q退出)：6666
输入信息(输入q退出)：7777
server : 6666
输入信息(输入q退出)：
server : 7777
......
q
u202121331104@jmu-cs:~/net_exp$
```

连接之后，两个可以随便发消息。按下q后结束。

5. 挑战性任务

#####

先贴一开始书写的错误代码

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<sys/socket.h>
#include<arpa/inet.h>
#include<unistd.h>
#include<pthread.h>
#include <errno.h>
#include <netinet/in.h>
#define BUF_size 1024

#define serv_port 40104  //类外一个端口

typedef struct {
    int type;          //信息请求类型
    char username[20];//用户名
    char message[BUF_size];  //信息
}Packet;
void error_message(char* message) {
```

```
    fputs(message, stderr);
    fputc('\n', stderr);
    exit(1);
}

int main(int argc, char* argv[]) {
    if(2 != argc)
  {

       printf("Usage:%s portnumber\n", argv[0]);
       return EXIT_FAILURE;
     }
    int serv_sock;   //返回值
    Packet packet;   //发的包
    struct sockaddr_in serv_addr, rerv_addr;//定义套接字
    socklen_t serv_addr_size;

    char buf[BUF_size];
    //create套接字
    serv_sock = socket(PF_INET, SOCK_DGRAM, 0);
    if (serv_sock == -1)
        error_message("UDP创建失败");

    //connect  服务
    memset(&serv_addr, 0, sizeof(serv_addr));
    //
    int listen_port = atoi(argv[1]);

    //设定发送窗口
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = htonl(INADDR_BROADCAST);
    serv_addr.sin_port = htons(serv_port);
    if (bind(serv_sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr)) == -1)
        error_message("bind error");
    //设定接受窗口
    rerv_addr.sin_family = AF_INET;
    rerv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    rerv_addr.sin_port = htons(listen_port);

    while (1)
    {

        clnt_addr_size = sizeof(rerv_addr);
        //收到信息
        if (recvfrom(rerv_sock, &packet, sizeof(packet), 0, (struct
sockaddr*)&rerv_addr,
            &clnt_addr_size) == -1)
            error_message("recvfrom error");
        switch (packet.type)
        {
        case 1://登录请求
            sprintf(buf, "%s加入聊天\n", packet.username);//打到字符串中
            printf("%s\n", buf);
            strcpy(packet.message, buf);
            //广播到所有客户端
            //clnt_addr.sin_addr.s_addr = htonl(INADDR_BROADCAST);
            if (sendto(serv_sock, &packet, sizeof(packet), 0, (struct
sockaddr*)&serv_addr,
```

```
                sizeof(serv_addr)) == -1)//发送
                error_message("发送失败");
            break;
        case 2://聊天信息
            printf("%s : %s\n", packet.username, packet.message);
            //clnt_addr.sin_addr.s_addr = htonl(INADDR_BROADCAST);
            if (sendto(serv_sock, &packet, sizeof(packet), 0, (struct
sockaddr*)&serv_addr,
                sizeof(serv_addr)) == -1)//发送
                error_message("发送失败");
            break;
        default:
            break;
        }

    }
    close(serv_sock);
    return EXIT_SUCCESS;

}
```

错误分析：

一开始其实直接开始做附加题的，但是一开始我不知道要怎么让UDP广播到所有连接的客户端，后来网上说可以把`clnt_addr.sin_addr.s_addr 设置为htonl(INADDR_BROADCAST);`这样就可以实现发送到广播地址局域网内的255.255.255.255，但尝试之后可以正常的发送，但无法实现广播，即无法回复到客户端，而是不断在本身服务端循环发送。通过查看资料后，广播地址其实这个方法也不可取，局域网内不知道有几台机子，而且广播也不是很好。之后同学跟我讲可以开两个端口，一个用于发送，一个用于接受，但我实际后还只能发送消息无法接送消息，后面我有发现了一篇文章，他讲述了怎么用UDP搭建一个群聊天室，我发现要想发送个所有的客户端，就应该发送n次消息而不只发送一次。

```
while (1)
{
    memset(&msg, 0, sizeof(msg));//清空操作
    memset(&clientaddr, 0, sizeof(clientaddr));//清空操作
    if ((recvfrom(sockfd, &msg, sizeof(msg), 0, (struct sockaddr *)&clientaddr,
&clientaddr_len)) == -1)

typedef struct _NODE
{
    struct sockaddr_in c_addr;//数据域
    struct _NODE *next;//指针域
} node_t;
while (p->next != NULL)
    {
        p = p->next;
        //判断链表客户端信息是否是自己
        //是自己就不发送
        if (memcmp(&(p->c_addr), &clientaddr, sizeof(clientaddr)))
        {
            if (sendto(sockfd, &msg, sizeof(msg_t), 0, (struct sockaddr *)&(p-
>c_addr), sizeof(p->c_addr)) == -1)
            {
                PRINT_ERR("recvfrom error");
            }
        }
```

```
    }
    在这里的代码中，他把客户端的消息存在链表中，在和服务端的比较信息，不同是就转发。
```

看完后我才真正理解了sockaddr_in 结构体中的是什么东西和sendto函数和recvfrom函数进行完后的变化。

sendto函数中的`(struct sockaddr *)&addr`是目的网络地址的数据，只要对应上指定的网络信息就可以发送到指定的主机1中。

recvfrom函数中的`(struct sockaddr *)&addr`是发送方的网络地址信息，他存储着主机端网络信息，只要保存这些信息，就可以让服务端转发到指定的主机。

了解了这两个函数的意义后就变的很简单，只要在客户端登录时保存好发送方的`recv_addr`数据，在发送给客户端时一个个发送给存储好的数据地址就好了。

正解（重点）：

服务端：服务器中对应为`serverplus.c`

```
typedef struct _NODE
{
    struct sockaddr_in c_addr;//数据域
    struct _NODE *next;//指针域
} node_t;

void error_message(char* message) {
    fputs(message, stderr);
    fputc('\n', stderr);
    exit(1);
}
void creat_link(node_t **head);
void do_register(int sock,Packet packet, struct sockaddr_in recv_addr, node_t
*phead);
void do_group_chat(int sock,Packet packet, struct sockaddr_in recv_addr, node_t
*phead);
void quit_group_chat(int sock,Packet packet, struct sockaddr_in recv_addr,
node_t *phead);

void creat_link(node_t **head)
{
    *head = (node_t *)malloc(sizeof(node_t));
}
void do_register(int sock,Packet packet, struct sockaddr_in recv_addr, node_t
*phead)
{
    //遍历链表将登录信息发送给所以人
    node_t *p = phead;
    while (p->next != NULL)
    {
        p = p->next;
        if (sendto(sock, &packet, sizeof(packet),0, (struct sockaddr *)&(p-
>c_addr), sizeof(p->c_addr)) == -1)
        {
            error_message("recvfrom error");
        }
    }
```

```
    //将登录的客户端信息插入保存在链表
    //头插法
    //定义一个新的指针保存客户端信息
    node_t *newp = NULL;
    creat_link(&newp);
    newp->c_addr = recv_addr;
    newp->next = phead->next;
    phead->next = newp;
}
void do_group_chat(int sock,Packet packet, struct sockaddr_in recv_addr, node_t
*phead)
{
    //遍历链表，将消息发给除自己之外的所有人
    node_t *p = phead;
    while (p->next != NULL)
    {
        p = p->next;
        //判断链表客户端信息是否是自己
        //是自己就不发送
        if (memcmp(&(p->c_addr), &recv_addr, sizeof(recv_addr)))
        {
            if (sendto(sock, &packet, sizeof(packet), 0, (struct sockaddr *)&(p-
>c_addr), sizeof(p->c_addr)) == -1)
            {
                error_message("recvfrom error");
            }
        }
    }

}
void quit_group_chat(int sock,Packet packet, struct sockaddr_in recv_addr,
node_t *phead)
{
    node_t *p = phead;

    while (p->next != NULL)
    {
        //判断链表客户端信息是否是自己
        //是自己就不发送并且将自己的客户端信息在链表内删除
        if (memcmp(&(p->next->c_addr), &recv_addr, sizeof(recv_addr)))
        {
            p = p->next;
            if (sendto(sock, &packet, sizeof(packet), 0, (struct sockaddr *)&(p-
>c_addr), sizeof(p->c_addr)) == -1)
            {
                error_message("recvfrom error");
            }
        }
        else
        {
            node_t *pnew;
            pnew = p->next;
            p->next = pnew->next;
            pnew->next = NULL;
            free(pnew);
            pnew = NULL;
        }
    }
```

```
}
```

客户端：服务器对应`clientplus.c`

和client.c几乎没有变化，只是增加了退出聊天室的type

```
void* Send_Message(void* arg) {
    while (1) {
        printf("输入信息(输入q退出)：");
        scanf("%s", packet.message);
        if (strcmp(packet.message, "q") == 0)
            break;
        strcpy(packet.username, staticusername);
        packet.type = 2;//发送请求
        if (sendto(sock, &packet, sizeof(packet), 0, (struct
sockaddr*)&serv_addr,
        sizeof(serv_addr)) == -1)//发送
        error_message("发送失败");
    }
    packet.type = 3;//登出请求
    sprintf(packet.message, "%s 离开了聊天", packet.username);
        if (sendto(sock, &packet, sizeof(packet), 0, (struct
sockaddr*)&serv_addr,
        sizeof(serv_addr)) == -1)//发送
        error_message("发送失败");
    pthread_exit(NULL);
}
void* Recv_Message(void* arg){
    socklen_t serv_addr_size;  //一种数据类型，它其实和int差不多，记录长度
    while (1) {
        serv_addr_size = sizeof(serv_addr);
        //接收
        if (recvfrom(sock, &packet, sizeof(packet), 0, (struct
sockaddr*)&serv_addr,
            &serv_addr_size) == -1)
            error_message("recvfrom error");
        switch (packet.type)
        {
        case 1://登录请求
            printf("\n%s加入聊天\n", packet.username);//打到字符串中
            break;
        case 2://聊天信息
            printf("\n%s : %s\n", packet.username, packet.message);
            break;
        case 3://退出信息
        printf("\n%s退出聊天\n", packet.username);//打到字符串中
            break;
    }
}
    pthread_exit(NULL);//关闭线程
}
```

其中链表部分借鉴模仿了那篇文章的方法，非常的好用。

运行结果及分析

测试了一段时间，目前没有什么bug。

服务端：

```
u202121331104@jmu-cs:~/net_exp$ ./serverplus 30104

laoba 加入聊天

laoliu 加入聊天

laoba : 你好

laoliu : 你好
```

客户端(老六)：

```
./clientplus 127.0.0.1 30104
输入名称：laoliu
输入信息(输入q退出)：
laoba : 你好
你好
输入信息(输入q退出)：这是老六
输入信息(输入q退出)：
laoba : 这是老八
```

客户端(老八)：

```
u202121331104@jmu-cs:~/net_exp$ ./clientplus 127.0.0.1 30104
输入名称：laoba
输入信息(输入q退出)：
laoliu加入聊天
你好
输入信息(输入q退出)：
laoliu : 你好

laoliu : 这是老六
这是老八
输入信息(输入q退出)：^C
u202121331104@jmu-cs:~/net_exp$
```

6. 实验过程中遇到的问题及解决方法

一开始做的是加分部分，很煎熬。

首先那两个函数什么都不懂，而且sockaddr_in这个结构体是什么都不懂?

解决方法：查资料，通过资料得知了sockaddr_in这个结构体的用法

第一部分算解决了。

但通过上面部分可知加分部分失败了，所以先做了普通部分。而普通部分很简单，没什么问题。

本来是放弃了加分部分，但想想还是做了。

之后就出现了上面的问题，一直无法收到消息，后来我查看了文章后得知其实我没有把收到的网络地址保存下了所以发送的地址为0,客户端当然没有接受到。

有了那篇文章，书写起来就很简单。

7.参考文章：

【Linux网络编程】基于UDP实现多人聊天室_夜猫徐的博客-CSDN博客

sockaddr和sockaddr_in详解_爱橙子的OK绷的博客-CSDN博客

8. 附完整源代码

server.c(非附加)

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<sys/socket.h>
#include<arpa/inet.h>
#include<unistd.h>
#include<pthread.h>
#include<errno.h>
#include<netinet/in.h>
//#include<time.h>
#define BUF_size 1024


typedef struct{
    int type;          //信息请求类型
    char username[20];//用户名
    char message[BUF_size];  //信息
}Packet;

char staticusername[20];

void* Send_Message(void *arg);
void* Recv_Message(void *arg);

void error_message(char* message) {
    fputs(message, stderr);
    fputc('\n', stderr);
    exit(1);
}

int sock;   //返回值
Packet packet;   //发的包
struct sockaddr_in serv_addr,recv_addr2;//定义套接字

int main(int argc,char *argv[]) {
    //socklen_t clnt_addr_size;
```

```
    pthread_t send_thread, recv_thread;//定义线程
    if(2 != argc)

     {

          printf("Usage:%s portnumber\n", argv[0]);

          return EXIT_FAILURE;

     }
    //create套接字
    sock = socket(PF_INET, SOCK_DGRAM, 0);
    if (sock == -1)
        error_message("UDP创建失败");
    //connect  服务
    memset(&serv_addr, 0, sizeof(serv_addr));
    int port = atoi(argv[1]);
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    serv_addr.sin_port = htons(port);
    if (bind(sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr)) == -1)
        error_message("bind error");
    printf("输入名称：");
    scanf("%s", packet.username);
    strcpy(staticusername, packet.username);//复制到全局中
    packet.type = 1;  //当前信息为登录信息
    if (sendto(sock, &packet, sizeof(packet), 0, (struct sockaddr*)&serv_addr,
        sizeof(serv_addr)) == -1)//发送
        error_message("发送失败");
        // int pt1, pt2;
    //建立发送信息线程
    pthread_create(&send_thread, NULL, Send_Message, (void*)&sock);
    //建立收到信息线程
    pthread_create(&recv_thread, NULL, Recv_Message, (void*)&sock);
    //阻塞线程，让主进程等待
    pthread_join(send_thread, NULL);
    //pthread_join(recv_thread, NULL);
    close(sock);
    return EXIT_SUCCESS;
}
void* Send_Message(void* arg) {

    while (1) {
        printf("输入信息(输入q退出)：");
        scanf("%s", packet.message);
        if (strcmp(packet.message, "q") == 0)
            break;
        strcpy(packet.username, staticusername);
        packet.type = 2;//发送请求
        if (sendto(sock, &packet, sizeof(packet), 0, (struct
sockaddr*)&serv_addr,
        sizeof(serv_addr)) == -1)//发送
        error_message("发送失败");
    }
    packet.type = 2;//登出请求
    sprintf(packet.message, "%s 离开了聊天", packet.username);
        if (sendto(sock, &packet, sizeof(packet), 0, (struct
sockaddr*)&serv_addr,
```

```
        sizeof(serv_addr)) == -1)//发送
        error_message("发送失败");
    pthread_exit(NULL);
}
void* Recv_Message(void* arg){
    socklen_t serv_addr_size;  //一种数据类型，它其实和int差不多，记录长度
    while (1) {
        serv_addr_size = sizeof(serv_addr);
        //接收
        if (recvfrom(sock, &packet, sizeof(packet), 0, (struct
sockaddr*)&serv_addr,
            &serv_addr_size) == -1)
            error_message("recvfrom error");
        switch (packet.type)
        {
        case 1://登录请求
            printf("\n%s加入聊天\n", packet.username);//打到字符串中
            break;
        case 2://聊天信息
            printf("\n%s : %s\n", packet.username, packet.message);
            break;
    }
}
    pthread_exit(NULL);//关闭线程
}
```

client.c(非附加)

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<sys/socket.h>
#include<arpa/inet.h>
#include<unistd.h>
#include<pthread.h>
#include<errno.h>
#include<netinet/in.h>
//#include<time.h>
#define BUF_size 1024


typedef struct{
    int type;          //信息请求类型
    char username[20];//用户名
    char message[BUF_size];  //信息
}Packet;

char staticusername[20];

void* Send_Message(void *arg);
void* Recv_Message(void *arg);

void error_message(char* message) {
    fputs(message, stderr);
    fputc('\n', stderr);
```

```
    exit(1);
}

int sock;   //返回值
Packet packet;   //发的包
struct sockaddr_in serv_addr;//定义套接字

int main(int argc,char *argv[]) {
    //socklen_t clnt_addr_size;
    pthread_t send_thread, recv_thread;//定义线程

    if (argc != 3) {//ip转为10进制
        printf("Usage:%s hostname portnumber\n", argv[0]);
        return EXIT_FAILURE;
    }
    //char buf[BUF_size];
    //create套接字
    sock = socket(PF_INET, SOCK_DGRAM, 0);
    if (sock == -1)
        error_message("UDP创建失败");

    //connect  服务
    memset(&serv_addr, 0, sizeof(serv_addr));
    int port = atoi(argv[2]);
    //用于发送信息
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    serv_addr.sin_port = htons(port);
    printf("输入名称：");
    scanf("%s", packet.username);
    strcpy(staticusername, packet.username);//复制到全局中
    packet.type = 1;  //当前信息为登录信息
    if (sendto(sock, &packet, sizeof(packet), 0, (struct sockaddr*)&serv_addr,
        sizeof(serv_addr)) == -1)//发送
        error_message("发送失败");
        // int pt1, pt2;

    //建立发送信息线程
    pthread_create(&send_thread, NULL, Send_Message, (void*)&sock);
    //建立收到信息线程
    pthread_create(&recv_thread, NULL, Recv_Message, (void*)&sock);
    //阻塞线程，让主进程等待
    pthread_join(send_thread, NULL);
    //pthread_join(recv_thread, NULL);
    close(sock);
    return EXIT_SUCCESS;
}

void* Send_Message(void* arg) {
    while (1) {
        printf("输入信息(输入q退出)：");
        scanf("%s", packet.message);
        if (strcmp(packet.message, "q") == 0)
            break;
        strcpy(packet.username, staticusername);
        packet.type = 2;//发送请求
        if (sendto(sock, &packet, sizeof(packet), 0, (struct
sockaddr*)&serv_addr,
```

```
        sizeof(serv_addr)) == -1)//发送
        error_message("发送失败");
    }
    packet.type = 2;//登出请求
    sprintf(packet.message, "%s 离开了聊天", packet.username);
        if (sendto(sock, &packet, sizeof(packet), 0, (struct
sockaddr*)&serv_addr,
        sizeof(serv_addr)) == -1)//发送
        error_message("发送失败");
    pthread_exit(NULL);
}
void* Recv_Message(void* arg){
    socklen_t serv_addr_size;  //一种数据类型，它其实和int差不多，记录长度
    while (1) {
        serv_addr_size = sizeof(serv_addr);
        //接收
        if (recvfrom(sock, &packet, sizeof(packet), 0, (struct
sockaddr*)&serv_addr,
            &serv_addr_size) == -1)
            error_message("recvfrom error");
        switch (packet.type)
        {
        case 1://登录请求
            printf("\n%s加入聊天\n", packet.username);//打到字符串中
            break;
        case 2://聊天信息
            printf("\n%s : %s\n", packet.username, packet.message);
            break;
    }
}
    pthread_exit(NULL);//关闭线程
}
```

server.c(附加)

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<sys/socket.h>
#include<arpa/inet.h>
#include<unistd.h>
#include<pthread.h>
#include <errno.h>
#include <netinet/in.h>
#define BUF_size 1024



typedef struct {
    int type;          //信息请求类型
    char username[20];//用户名
    char message[BUF_size];  //信息
}Packet;

typedef struct _NODE
```

```
{
    struct sockaddr_in c_addr;//数据域
    struct _NODE *next;//指针域
} node_t;

void error_message(char* message) {
    fputs(message, stderr);
    fputc('\n', stderr);
    exit(1);
}
void creat_link(node_t **head);
void do_register(int sock,Packet packet, struct sockaddr_in recv_addr, node_t
*phead);
void do_group_chat(int sock,Packet packet, struct sockaddr_in recv_addr, node_t
*phead);
void quit_group_chat(int sock,Packet packet, struct sockaddr_in recv_addr,
node_t *phead);

int main(int argc, char* argv[]) {

    if(2 != argc)

  {

          printf("Usage:%s portnumber\n", argv[0]);

          return EXIT_FAILURE;

     }
    int serv_sock;   //返回值
    Packet packet;   //发的包
    struct sockaddr_in serv_addr,recv_addr;//定义套接字
    socklen_t recv_addr_size;
    char buf[BUF_size];
    //create套接字
    serv_sock = socket(PF_INET, SOCK_DGRAM, 0);
    if (serv_sock == -1)
        error_message("UDP创建失败");
    //connect  服务
    memset(&serv_addr, 0, sizeof(serv_addr));
    int listen_port = atoi(argv[1]);
    //设定发送窗口
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    serv_addr.sin_port = htons(listen_port);
    if (bind(serv_sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr)) == -1)
        error_message("bind error");

    //定义链表头节点
    node_t *phead = NULL;
     creat_link(&phead);
     phead->next = NULL;


    while (1)
    {

        memset(&recv_addr, 0, sizeof(recv_addr));//清空操作
```

```
        recv_addr_size=sizeof(recv_addr);
        //收到信息
        if (recvfrom(serv_sock, &packet, sizeof(packet), 0, (struct
sockaddr*)&recv_addr,
            &recv_addr_size) == -1)
            error_message("recvfrom error");
        switch (packet.type)
        {
        case 1://登录请求
        printf("\n%s 加入聊天\n", packet.username);
          do_register(serv_sock,packet,recv_addr,phead);
          break;
        case 2://聊天信息
        printf("\n%s : %s\n", packet.username, packet.message);
            do_group_chat(serv_sock,packet,recv_addr,phead);
            break;
        case 3:
            printf("\n%s退出聊天", packet.username);
            quit_group_chat(serv_sock,packet,recv_addr,phead);
        default:
            break;
        }

    }
    close(serv_sock);
    return EXIT_SUCCESS;
}
void creat_link(node_t **head)
{
    *head = (node_t *)malloc(sizeof(node_t));
}
void do_register(int sock,Packet packet, struct sockaddr_in recv_addr, node_t
*phead)
{
    //遍历链表将登录信息发送给所以人
    node_t *p = phead;
    while (p->next != NULL)
    {
        p = p->next;
        if (sendto(sock, &packet, sizeof(packet),0, (struct sockaddr *)&(p-
>c_addr), sizeof(p->c_addr)) == -1)
        {
            error_message("recvfrom error");
        }
    }
    //将登录的客户端信息插入保存在链表
    //头插法
    //定义一个新的指针保存客户端信息
    node_t *newp = NULL;
    creat_link(&newp);
    newp->c_addr = recv_addr;
    newp->next = phead->next;
    phead->next = newp;
}
void do_group_chat(int sock,Packet packet, struct sockaddr_in recv_addr, node_t
*phead)
{
    //遍历链表，将消息发给除自己之外的所有人
```

```
    node_t *p = phead;
    while (p->next != NULL)
    {
        p = p->next;
        //判断链表客户端信息是否是自己
        //是自己就不发送
        if (memcmp(&(p->c_addr), &recv_addr, sizeof(recv_addr)))
        {
            if (sendto(sock, &packet, sizeof(packet), 0, (struct sockaddr *)&(p-
>c_addr), sizeof(p->c_addr)) == -1)
            {
                error_message("recvfrom error");
            }
        }
    }

}
void quit_group_chat(int sock,Packet packet, struct sockaddr_in recv_addr,
node_t *phead)
{
    node_t *p = phead;

    while (p->next != NULL)
    {
        //判断链表客户端信息是否是自己
        //是自己就不发送并且将自己的客户端信息在链表内删除
        if (memcmp(&(p->next->c_addr), &recv_addr, sizeof(recv_addr)))
        {
            p = p->next;
            if (sendto(sock, &packet, sizeof(packet), 0, (struct sockaddr *)&(p-
>c_addr), sizeof(p->c_addr)) == -1)
            {
                error_message("recvfrom error");
            }
        }
        else
        {
            node_t *pnew;
            pnew = p->next;
            p->next = pnew->next;
            pnew->next = NULL;
            free(pnew);
            pnew = NULL;
        }
    }
}
```

client.c(附加)

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<sys/socket.h>
#include<arpa/inet.h>
#include<unistd.h>
#include<pthread.h>
#include<errno.h>
```

```c
#include<netinet/in.h>
//#include<time.h>
#define BUF_size 1024


typedef struct{
    int type;         //信息请求类型
    char username[20];//用户名
    char message[BUF_size];  //信息
}Packet;

char staticusername[20];

void* Send_Message(void *arg);
void* Recv_Message(void *arg);

void error_message(char* message) {
    fputs(message, stderr);
    fputc('\n', stderr);
    exit(1);
}

int sock;   //返回值
Packet packet;   //发的包
struct sockaddr_in serv_addr;//定义套接字

int main(int argc,char *argv[]) {
    //socklen_t clnt_addr_size;
    pthread_t send_thread, recv_thread;//定义线程

    if (argc != 3) {//ip转为10进制
        printf("Usage:%s hostname portnumber\n", argv[0]);
        return EXIT_FAILURE;
    }
    //char buf[BUF_size];
    //create套接字
    sock = socket(PF_INET, SOCK_DGRAM, 0);
    if (sock == -1)
        error_message("UDP创建失败");

    //connect  服务
    memset(&serv_addr, 0, sizeof(serv_addr));
    int port = atoi(argv[2]);
    //用于发送信息
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    serv_addr.sin_port = htons(port);
    printf("输入名称：");
    scanf("%s", packet.username);
    strcpy(staticusername, packet.username);//复制到全局中
    packet.type = 1;  //当前信息为登录信息
    if (sendto(sock, &packet, sizeof(packet), 0, (struct sockaddr*)&serv_addr,
        sizeof(serv_addr)) == -1)//发送
        error_message("发送失败");
        // int pt1, pt2;

    //建立发送信息线程
    pthread_create(&send_thread, NULL, Send_Message, (void*)&sock);
```

```
    //建立收到信息线程
    pthread_create(&recv_thread, NULL, Recv_Message, (void*)&sock);
    //阻塞线程，让主进程等待
    pthread_join(send_thread, NULL);
    //pthread_join(recv_thread, NULL);
    close(sock);
    return EXIT_SUCCESS;
}

void* Send_Message(void* arg) {
    while (1) {
        printf("输入信息(输入q退出)：");
        scanf("%s", packet.message);
        if (strcmp(packet.message, "q") == 0)
            break;
        strcpy(packet.username, staticusername);
        packet.type = 2;//发送请求
        if (sendto(sock, &packet, sizeof(packet), 0, (struct
sockaddr*)&serv_addr,
        sizeof(serv_addr)) == -1)//发送
        error_message("发送失败");
    }
    packet.type = 3;//登出请求
    sprintf(packet.message, "%s 离开了聊天", packet.username);
        if (sendto(sock, &packet, sizeof(packet), 0, (struct
sockaddr*)&serv_addr,
        sizeof(serv_addr)) == -1)//发送
        error_message("发送失败");
    pthread_exit(NULL);
}
void* Recv_Message(void* arg){
    socklen_t serv_addr_size;  //一种数据类型，它其实和int差不多，记录长度
    while (1) {
        serv_addr_size = sizeof(serv_addr);
        //接收
        if (recvfrom(sock, &packet, sizeof(packet), 0, (struct
sockaddr*)&serv_addr,
            &serv_addr_size) == -1)
            error_message("recvfrom error");
        switch (packet.type)
        {
        case 1://登录请求
            printf("\n%s加入聊天\n", packet.username);//打到字符串中
            break;
        case 2://聊天信息
            printf("\n%s : %s\n", packet.username, packet.message);
            break;
        case 3://退出信息
        printf("\n%s退出聊天\n", packet.username);//打到字符串中
            break;
    }
}
    pthread_exit(NULL);//关闭线程
}
```