





# 知识点1 【抓包工具的使用】（了解）

## 1、软件的安装

名称	修改日期
 wireshark-win32-1.6.2.exe	2014/9/11 9:43
 Wireshark-win64-2.6.2.exe	2018/8/30 20:45
 Wireshark使用.pdf	2019/3/22 16:34
 Wireshark抓包全集 (85种协议、类别的...)	2014/9/11 9:43

## 2、过滤规则

### 1、协议过滤: tcp udp http tftp

### 2、端口过滤:

udp.port==8000 udp协议的源端口或目的端口为8000

udp.srcport==8000 udp协议的源端口为8000

udp.dstport==8000 udp协议的目的端口为8000

tcp.port==8000 tcp协议的源端口或目的端口为8000

tcp.srcport==8000 tcp协议的源端口为8000

tcp.dstport==8000 tcp协议的目的端口为8000

### 3、ip地址过滤

ip.addr==10.9.21.201 源或目的IP为10.9.21.201

ip.src==10.9.21.201 源IP为10.9.21.201

ip.dst==10.9.21.201 目的IP为10.9.21.201

### 4、mac地址过滤

eth.dst == A0:00:00:04:C5:84 // 过滤目标mac

eth.src eq A0:00:00:04:C5:84 // 过滤源mac

eth.addr eq A0:00:00:04:C5:84 // 过滤来源 MAC 和目标 MAC

# 知识点2 【TFTP简单文件传送协议】（了解）

通信原理：协议的报文格式

通信流程：协议的通信步骤

## 1、协议的概述

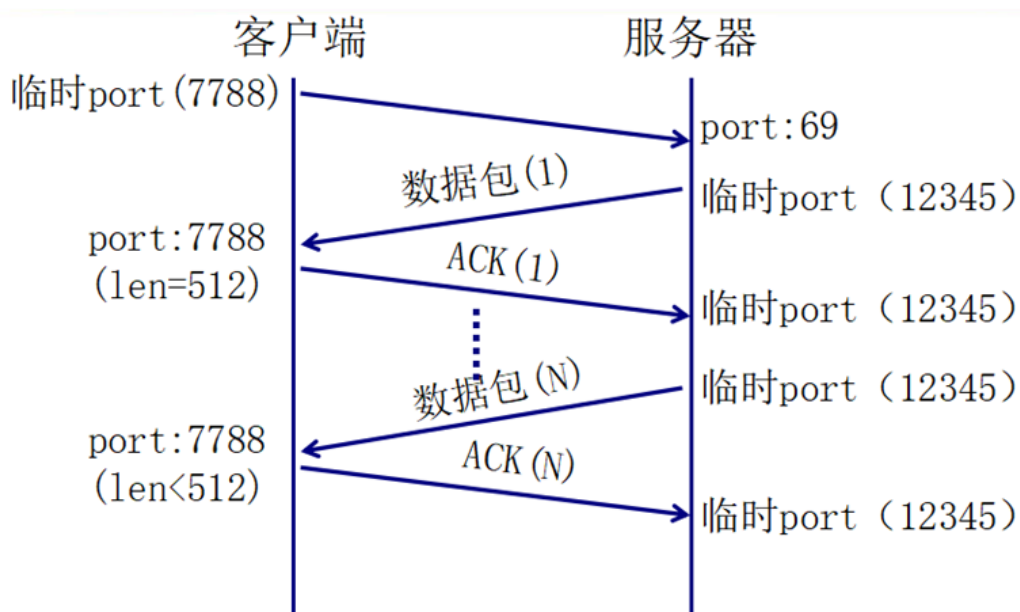
TFTP简单文件传送协议，基于UDP实现

数据传输模式：

octet：二进制模式

netascii：文本模式

## 2、通信流程



### 1、客户端的流程：下载文件(重要)

创建一个udp套接字sockfd

构建读请求报文 ----> 使用sendto (69号端口) ---> 发送读请求报文

open打开本地文件==fd

```
1
2 while(1)
3 {
4     int len = recvfrom接收服务器的应答
5     判断服务器的应答
6     出错：退出
7     成功：将文件数据写入本地文件--->write(fd,file_data, len);
8     给服务器回应ACK:
9     sendto--->ACK--->服务器的临时端口 (recvfrom倒数第二个参数)
10    if(len < 512)
11        break;
12 }
```

close(sockfd);

close(fd);

### 2、客户端的流程：上载文件

创建一个udp套接字sockfd

构建写请求报文 ----> 使用sendto (69号端口) ---> 发送写请求报文

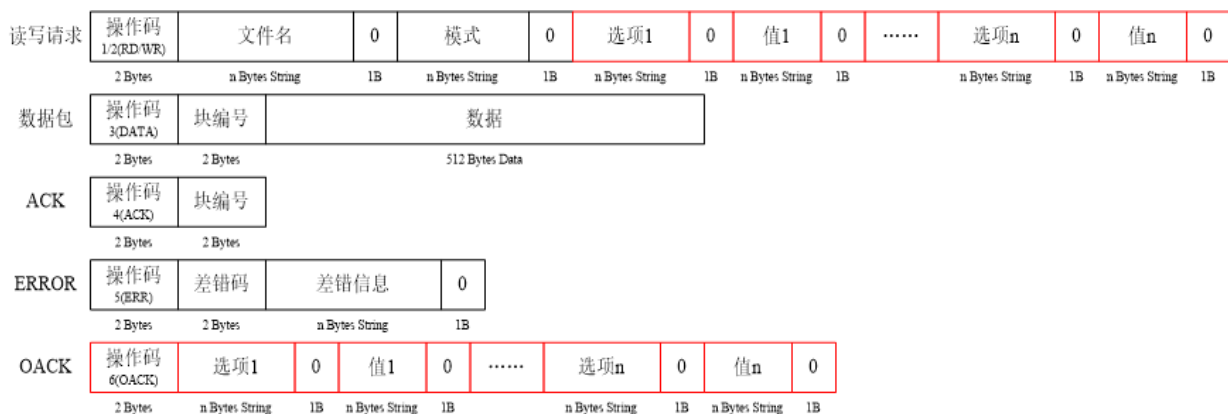
open打开本地文件 == fd

```
1
2 while(1)
3 {
4     read--->读取本地文件数据 int len = read(fd,buf)
5     sendto--->将文件数据报文 发送给服务器
6     recvfrom接收服务器的应答
7     if(len < 512)
8     break;
9 }
```

close(sockfd);

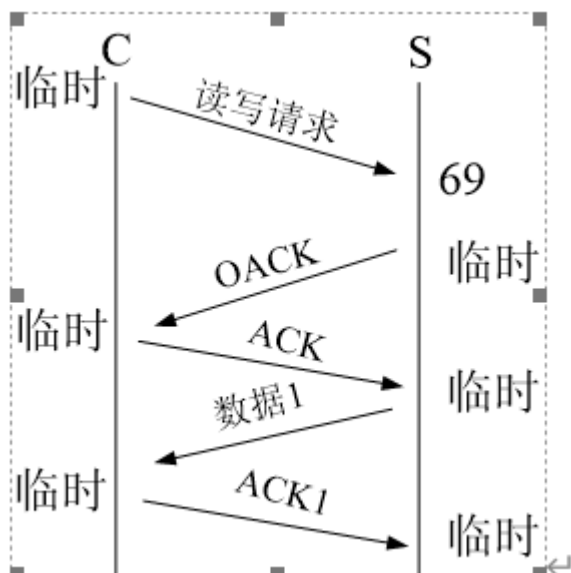
close(fd);

### 3、通信原理



### 4、选项





### tsize选项

当读操作时，**tsize**选项的参数必须为“0”，服务器会返回待读取的文件的大小  
 当写操作时，**tsize**选项参数应为待写入文件的大小，服务器会回显该选项

### blksize选项

修改传输文件时使用的数据块的大小（范围：8～65464）

### timeout选项

修改默认的数据传输超时时间（单位：秒）

## 5、TFTP客户端 不带选项 下载文件（案例）

```

1 #include <stdio.h>
2 #include <sys/socket.h>
3 #include <netinet/in.h>
4 #include <arpa/inet.h>
5 #include <sys/types.h>
6 #include <sys/stat.h>
7 #include <fcntl.h>
8 #include <string.h>
9 int main(int argc, char const *argv[])
10 {
11     if(argc != 3)
12     {
13         printf("./a.out server_ip fileName\n");
14         return 0;
15     }
16     //创建udp套接字
17     int sockfd = socket(AF_INET, SOCK_DGRAM, 0);
18
  
```

```
19 //构建下载报文
20 unsigned char cmd[512]="";
21 //需求: 以文本模式下载a.txt
22 int len = sprintf(cmd,"%c%c%s%c%s%c",0,1,argv[2], 0,"octet", 0);
23 //将cmd发送至 服务器的69号端口
24 struct sockaddr_in ser_addr;
25 bzero(&ser_addr, sizeof(ser_addr));
26 ser_addr.sin_family = AF_INET;
27 ser_addr.sin_port = htons(69);
28 ser_addr.sin_addr.s_addr = inet_addr(argv[1]);
29 //inet_pton(AF_INET, "10.9.21.211", &ser_addr.sin_addr.s_addr);
30 sendto(sockfd, cmd, len,0,\
31 (struct sockaddr *)&ser_addr, sizeof(ser_addr));
32
33 //打开文件文件 写的方式
34 int fd = open(argv[2], O_WRONLY|O_CREAT, 0666);
35 if(fd<0)
36 {
37     perror("open");
38     return 0;
39 }
40
41 //定义要接收的包编号
42 int pack_num=0;
43
44 //接收服务器的应答数据
45 while(1)
46 {
47     struct sockaddr_in from_addr;
48     socklen_t from_len = sizeof(from_addr);
49     unsigned char buf[1500]="";
50     int len = recvfrom(sockfd, buf,sizeof(buf), 0,\
51 (struct sockaddr *)&from_addr, &from_len);
52
53     //判断当前buf中的报文功能
54     if(buf[1] == 5)
55     {
56         printf("err");
57         break;
58     }
```

```

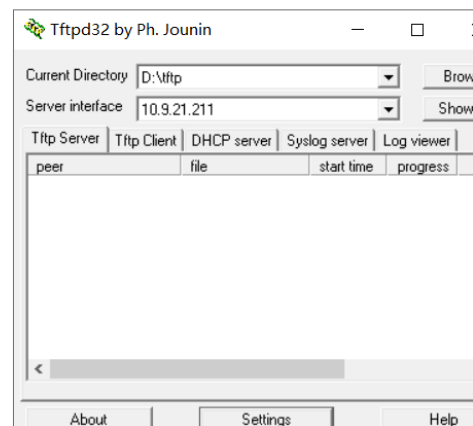
59     else if(buf[1] == 3)//文件数据
60     {
61         if((pack_num+1) == ntohs(*(unsigned short *)(buf+2)) )
62         {
63             pack_num = ntohs(*(unsigned short *)(buf+2));
64             printf("pack_num=%d\n", pack_num);
65             //将文件数据写入本地磁盘文件
66             write(fd, buf+4, len-4);
67         }
68         //回应ACK
69         buf[1]=4;
70         sendto(sockfd, buf, 4, 0, \
71             (struct sockaddr *)&from_addr, sizeof(from_addr));
72
73         //如果len<516接收文件完毕 退出
74         if(len < 516)
75             break;
76     }
77 }
78
79 //关闭文件
80 close(fd);
81 //关闭套接字
82 close(sockfd);
83
84 return 0;
85 }

```

```

edu@edu:~/work/net/day02$ gcc 00_tftp_download.c
edu@edu:~/work/net/day02$ ./a.out
./a.out server_ip fileName
edu@edu:~/work/net/day02$ ./a.out 10.9.21.211 a.txt
pack_num=1
pack_num=2
pack_num=3
pack_num=4
pack_num=5
pack_num=6
pack_num=7
edu@edu:~/work/net/day02$ ./a.out 10.9.21.211 a.txt

```



## 知识点3 【广播】（了解）

## 1、广播的概述

广播：由一台主机向该主机所在子网内的所有主机发送数据的方式

广播只能用**UDP**或**原始IP**实现，不能用**TCP**。

以下几个协议都用到广播

- 1、地址解析协议（ARP）
- 2、动态主机配置协议（DHCP）
- 3、网络时间协议（NTP）

## 2、广播的特点

- 1、处于同一子网的**所有主机**都必须**处理**数据
- 2、UDP数据包会沿协议栈向上一直到UDP层
- 3、运行音视频等较高速率工作的应用，会带来大负担
- 4、局限于**局域网**内使用。

## 3、广播的地址

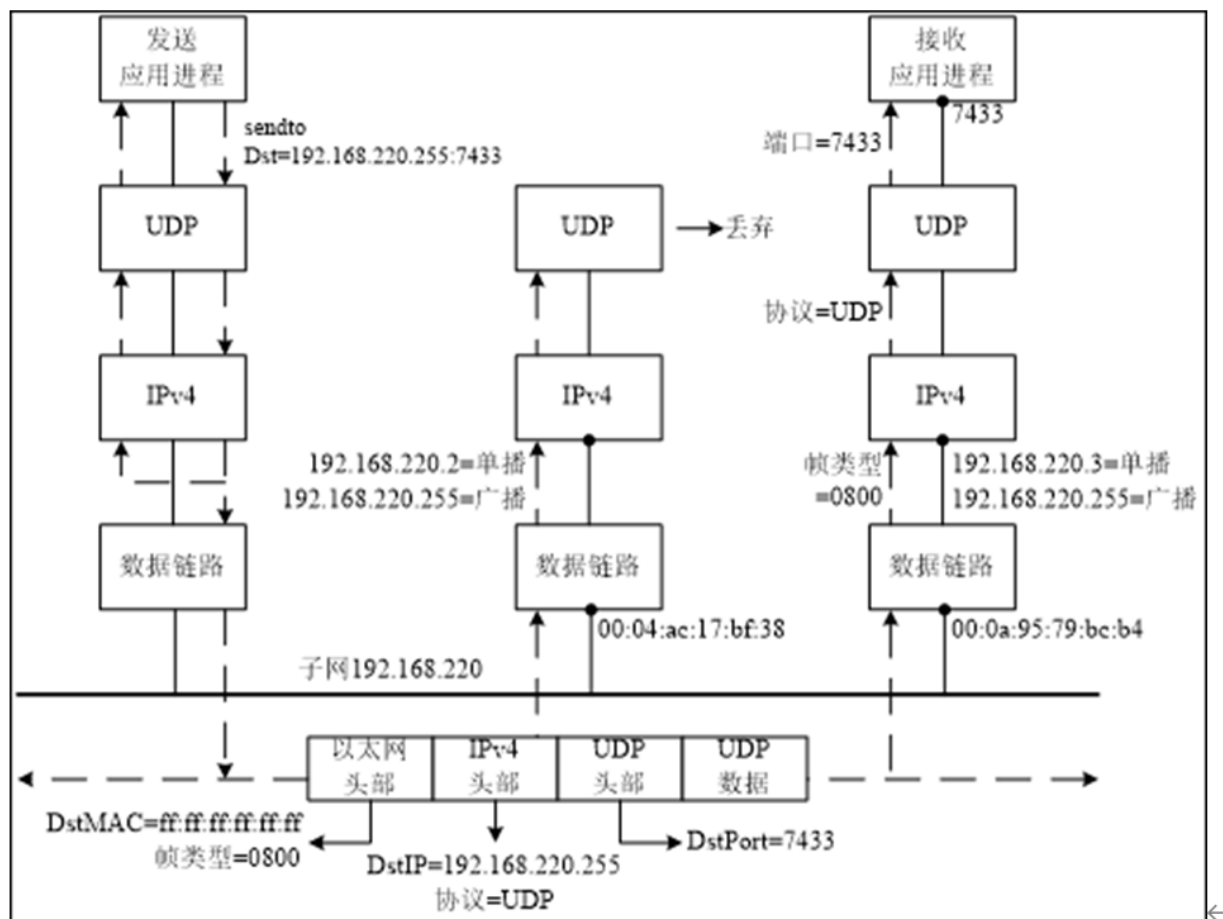
**定向广播地址**：主机ID全1

- 1、例：对于192.168.220.0/24，其定向广播地址为192.168.220.255
- 2、通常路由器不转发该广播

**受限广播地址**：255.255.255.255

路由器**从不转发**该广播

## 4、广播的流程



## 5、设置广播

```
1 int setsockopt(int sockfd, int level, int optname, const void *optval,
2               socklen_t optlen)
```

level	optname	说明	optval类型
SOL_SOCKET	SO_BROADCAST	允许发送广播数据包	int
	SO_RCVBUF	接收缓冲区大小	int
	SO_SNDBUF	发送缓冲区大小	int

```
1 #include <stdio.h>
2 #include <sys/socket.h>
3 #include <netinet/in.h>
4 #include <string.h>
5 #include <arpa/inet.h>
6 int main(int argc, char const *argv[])
7 {
8     //创建一个udp套接字
9     int sockfd= socket(AF_INET, SOCK_DGRAM, 0);
10
```

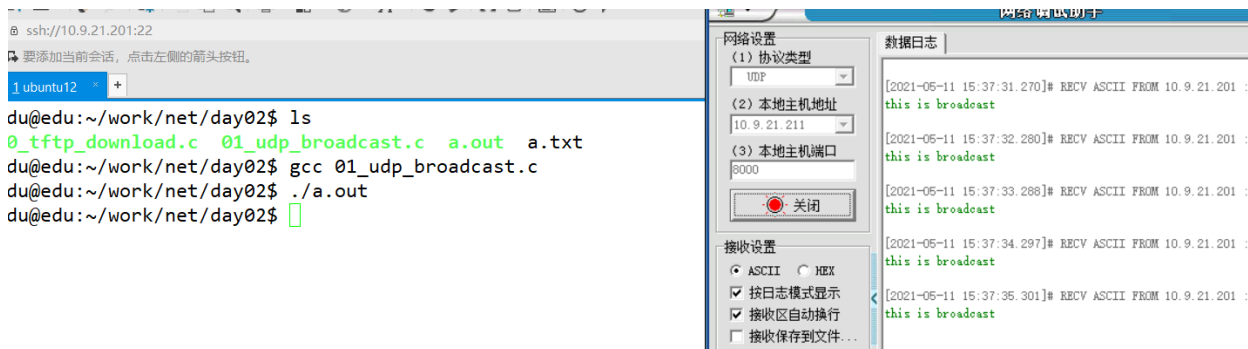


```

11 //设置套接字允许广播
12 int yes = 1;
13 setsockopt(sockfd, SOL_SOCKET, SO_BROADCAST, &yes, sizeof(int));
14
15 //定义地址结构存放广播地址信息
16 struct sockaddr_in dst_addr;
17 bzero(&dst_addr, sizeof(dst_addr));
18 dst_addr.sin_family = AF_INET;
19 dst_addr.sin_port = htons(8000);
20 dst_addr.sin_addr.s_addr = inet_addr("10.9.21.255");//广播地址
21 //广播
22 int i=0;
23 for ( i = 0; i < 5; i++)
24 {
25     sendto(sockfd, "this is broadcast", strlen("this is broadcast"), 0,
26         (struct sockaddr *)&dst_addr, sizeof(dst_addr));
27     sleep(1);
28 }
29
30 close(sockfd);
31 return 0;
32 }

```

运行结果：



## 知识点4【多播】（了解）

### 1、多播的概述

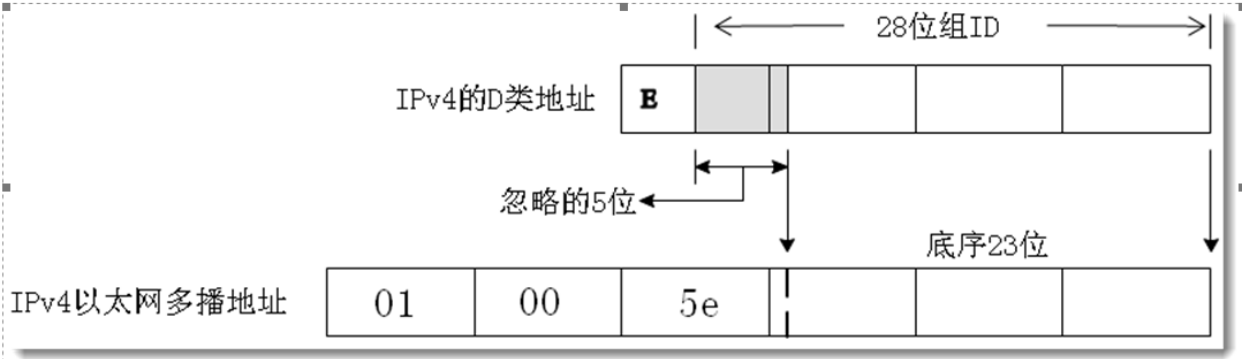
数据的收发仅仅在**同一分组**中进行

### 2、多播的特点：

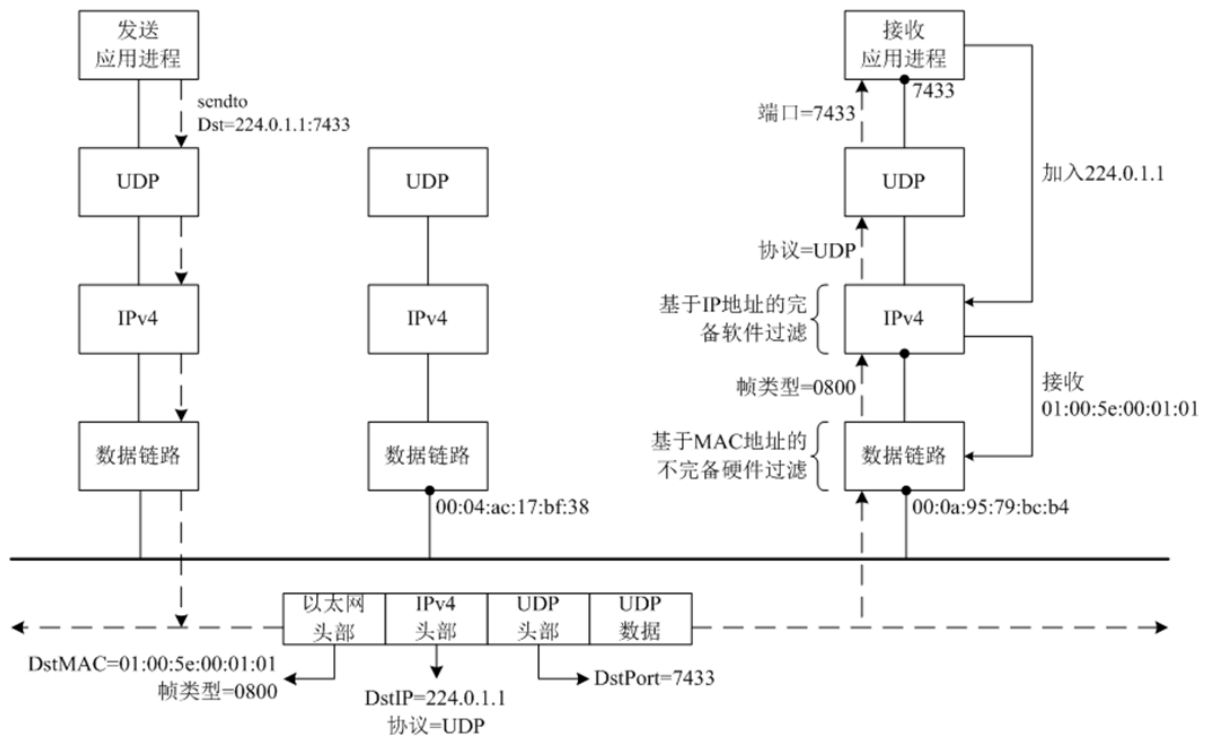
- 1、**多播地址**标示一组接口
- 2、多播可以用于**广域网**使用
- 3、在IPv4中，**多播**是可选的

3、多播地址

IPv4的D类地址是多播地址  
十进制：224.0.0.1~ 239.255.255.254  
多播地址映射：



4、多播的流程



只能将自己主机的IP加入多播组

基于mac地址不完备硬件过滤：

基于IP地址的完备软件过滤：

5、加入或退出多播组

```
1 int setsockopt(int sockfd, int level,int optname, const void *optval,
2 socklen_t optlen);
```

level	optname	说明	optval类型
IPPROTO_IP	IP_ADD_MEMBERSHIP	加入多播组	ip_mreq{}
	IP_DROP_MEMBERSHIP	离开多播组	

IP_DROP_MEMBER SHIP	ip_mreq{
------------------------	----------

```
1 struct in_addr
2 {
3     in_addr_t s_addr;
4 };
5 struct ip_mreq
6 {
7     struct in_addr imr_multiaddr; //多播组IP
8     struct in_addr imr_interface; //将要添加到多播组的IP
9 };
```

```
1 #include <stdio.h>
2 #include <sys/socket.h>
3 #include <netinet/in.h>
4 #include <string.h>
5 #include <arpa/inet.h>
6 int main(int argc, char const *argv[])
7 {
8     //创建一个udp套接字
9     int sockfd= socket(AF_INET, SOCK_DGRAM, 0);
10
11     //bind绑定固定的IP 端口
12     struct sockaddr_in my_addr;
13     bzero(&my_addr, sizeof(my_addr));
14     my_addr.sin_family = AF_INET;
15     my_addr.sin_port = htons(8000);
16     my_addr.sin_addr.s_addr = htonl(INADDR_ANY);
17     bind(sockfd, (struct sockaddr *)&my_addr, sizeof(my_addr));
18
19     //将主机加入多播组224.0.0.1
20     struct ip_mreq mreq;
21     mreq.imr_multiaddr.s_addr = inet_addr("224.0.0.1"); //多播组IP
22     mreq.imr_interface.s_addr = htonl(INADDR_ANY); //主机的所有IP
23     //将主机的所有IP 加入到多播组中
24     setsockopt(sockfd, IPPROTO_IP, IP_ADD_MEMBERSHIP, &mreq, sizeof(mreq));
25
26     //测试接收多播组的消息
27     while(1)
```

```

28     {
29         unsigned char buf[1500]="";
30         recvfrom(sockfd, buf, sizeof(buf),0,NULL, NULL);
31         printf("buf=%s\n", buf);
32     }
33
34     close(sockfd);
35     return 0;
36 }

```

