组播技术：

组播可以实现小范围内的互联，在发送者和每一个接受者之间时间一对多点的网络连接，是广播通信的一种变种。

根据IP地址的规定，D类地址为组播地址，其网络号为固定的1110,第4到31位定义了某一特殊的组播地址，范围为224.0.0.0～239.255.255.255。其中224.0.0.0～244.0.0.255的地址，它们大多是为了特殊的目的保留的，不建议使用。

套接字的基本属性：组播参数对应5个参数，通过setsockopt设置

```
//加入组播
int setsockopt(client_socket,IPPROTO_IP,IP_ADD_MEMBERSHIP,&multiaddress,sizeof(multiaddress))

//退出组播
int setsockopt(client_socket,IPPROTO_IP,IP_DROP_MEMBERSHIP,&multiaddress,sizeof(multiaddress))


//这里有一个重要的参数multiaddress，结构：
struct ip_mreq{
    struct in_addr imr_multiaddr;         //组播地址
    struct in_addr imr_interface;           //IPv4地址
}
```

接收数据流程：

1. 服务器端设置一个多播地址，创建一个多播组。
2. 客户端指定多播地址，加入多播。
3. 程序结束后，退出多播。

```
//memberServer.c

#include <stdio.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char **argv)
{
    int server_socket;
    struct sockaddr_in address;

    //创建UDP
    server_socket=socket(AF_INET,SOCK_DGRAM,0);
    if(server_socket<0){
```

```
        perror("socket");
        return 0;
    }

    //初始化多播地址
    memset(&address,0,sizeof(address));
    address.sin_family=AF_INET;
    address.sin_port=htons(5555);
    address.sin_addr.s_addr=inet_addr("224.0.1.100");

    //发送信息
    while(1){
        char buf[200];
        printf("input your word:");
        scanf("%s",buf);
        if(sendto(server_socket,buf,sizeof(buf),0,(struct sockaddr*)&address,sizeof(address))
<0){
            perror("sendto");
            return 0;
        }
    }

    return 0;
}
//memberClient.c

#include <stdio.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char **argv)
{
    struct ip_mreq mreq;
    int serveraddress_len;
    int client_socket;
    struct sockaddr_in serveraddress;

    //初始化地址
    memset(&serveraddress,0,sizeof(serveraddress));
    serveraddress.sin_family=AF_INET;
    serveraddress.sin_port=htons(5555);
    serveraddress.sin_addr.s_addr=htonl(INADDR_ANY);
```

```
    if((client_socket=socket(AF_INET,SOCK_DGRAM,0))<0){
        perror("client");
        return 0;
    }

    //绑定SOCKET
    if(bind(client_socket,(struct sockaddr*)&serveraddress,sizeof(serveraddress))<0){
        printf("bind");
        return 0;
    }

    int opt=1;
    if(setsockopt(client_socket,SOL_SOCKET,SO_REUSEADDR,&opt,sizeof(opt))<0){
        printf("setsockopt1");
        return 0;
    }

    //加入多播
    mreq.imr_multiaddr.s_addr=inet_addr("244.0.1.100");
    mreq.imr_interface.s_addr=htonl(INADDR_ANY);

    if(setsockopt(client_socket,IPPROTO_IP,IP_ADD_MEMBERSHIP,&mreq,sizeof(mreq))<0){
        perror("setsockopt2");
        return 0;
    }

    while(1){
        char buf[200];
        serveraddress_len=sizeof(serveraddress);
        if(recvfrom(client_socket,buf,200,0,(struct sockaddr*)&serveraddress,
(socklen_t *)serveraddress_len)<0){
            perror("recvfrom");
        }
        printf("msg from server: %s\n",buf);

        if(strcmp(buf,"quit")==0){
            if(setsockopt(client_socket,IPPROTO_IP,IP_DROP_MEMBERSHIP,&mreq,sizeof(mreq))<0){
                perror("setsokopt3");
            }
            close(client_socket);
            return 0;
        }
    }

    return 0;
}
```

11.3 多播

单播用于两个主机之间的端对端通信，广播用于一个主机对整个局域网上所有主机上的数据通信。单播和广播是两个极端，要么对一个主机进行通信，要么对整个局域网上的主机进行通信。实际情况下，经常需要对一组特定的主机进行通信，而不是整个局域网上的所有主机，这就是多播的用途。

11.3.1 多播的概念

多播，也称为"组播"，将网络中同一业务类型主机进行了逻辑上的分组，进行数据收发的时候其数据仅仅在同一分组中进行，其他的主机没有加入此分组不能收发对应的 数据。

在广域网上广播的时候，其中的交换机和路由器只向需要获取数据的主机复制并转发数据。主机可以向路由器请求加入或退出某个组，网络中的路由器和交换机有选择地复制并传输数据，将数据仅仅传输给组内的主机。多播的这种功能，可以一次将数据发送到多个主机，又能保证不影响其他不需要（未加入组）的主机的其他通信。

相对于传统的一对一的单播，多播具有如下的优点：

具有同种业务的主机加入同一数据流，共享同一通道，节省了带宽和服务器的优点，具有广播的优点而又没有广播所需要的带宽。

服务器的总带宽不受客户端带宽的限制。由于组播协议由接收者的需求来确定是否进行数据流的转发，所以服务器端的带宽是常量，与客户端的数量无关。

与单播一样，多播是允许在广域网即Internet上进行传输的，而广播仅仅在同一局域网上才能进行。

组播的缺点：

多播与单播相比没有纠错机制，当发生错误的时候难以弥补，但是可以在应用层来实现此种功能。

多播的网络支持存在缺陷，需要路由器及网络协议栈的支持。

多播的应用主要有网上视频、网上会议等。

11.3.2 广域网的多播

多播的地址是特定的，D类地址用于多播。D类IP地址就是多播IP地址，即224.0.0.0至239.255.255.255之间的IP地址，并被划分为局部连接多播地址、预留多播地址和管理权限多播地址3类：

局部多播地址：在224.0.0.0～224.0.0.255之间，这是为路由协议和其他用途保留的地址，路由器并不转发属于此范围的IP包。

预留多播地址：在224.0.1.0～238.255.255.255之间，可用于全球范围（如Internet）或网络协议。

管理权限多播地址：在239.0.0.0～239.255.255.255之间，可供组织内部使用，类似于私有IP地址，不能用于Internet，可限制多播范围。

11.3.3 多播的编程

多播的程序设计使用setsockopt()函数和getsockopt()函数来实现，组播的选项是IP层的，其选项值和含义参见11.5所示。

表11.5 多播相关的选项

getsockopt()/setsock	含 义

opt()的选项	
IP_MULTICAST_TTL	设置多播组数据的TTL值
IP_ADD_MEMBERSHIP	在指定接口上加入组播组
IP_DROP_MEMBERSHIP	退出组播组
IP_MULTICAST_IF	获取默认接口或设置接口
IP_MULTICAST_LOOP	禁止组播数据回送

1．选项IP_MULTICASE_TTL

选项IP_MULTICAST_TTL允许设置超时TTL，范围为0～255之间的任何值，例如：

点击(此处)折叠或打开

```
1. unsigned char ttl=255;
2. setsockopt(s,IPPROTO_IP,IP_MULTICAST_TTL,&ttl,sizeof(ttl));
```

2．选项IP_MULTICAST_IF

选项IP_MULTICAST_IF用于设置组播的默认默认网络接口，会从给定的网络接口发送，另一个网络接口会忽略此数据。例如：

点击(此处)折叠或打开

```
1. struct  in_addr  addr;
2. setsockopt(s,IPPROTO_IP,IP_MULTICAST_IF,&addr,sizeof(addr))
```

参数addr是希望多播输出接口的IP地址，使用INADDR_ANY地址回送到默认接口。

默认情况下，当本机发送组播数据到某个网络接口时，在IP层，数据会回送到本地的回环接口，选项IP_MULTICAST_LOOP用于控制数据是否回送到本地的回环接口。例如：

点击(此处)折叠或打开

```
1. unsigned char  loop;
2. setsockopt(s,IPPROTO_IP,IP_MULTICAST_LOOP,&loop,sizeof(loop))
```

参数loop设置为0禁止回送，设置为1允许回送。

3．选项IP_ADD_MEMBERSHIP和IP_DROP_MEMBERSHIP

加入或者退出一个组播组，通过选项IP_ADD_MEMBERSHIP和IP_DROP_MEMBER- SHIP，对一个结构struct ip_mreq类型的变量进行控制，struct ip_mreq原型如下：

点击(此处)折叠或打开

```
1. struct  ip_mreq
2. {
3.             struct  in_addr imn_multiaddr;  /*加入或者退出的广播组IP地址*/
4.             struct  in_addr imr_interface;  /*加入或者退出的网络接口IP地址*/
5. }
```

选项IP_ADD_MEMBERSHIP用于加入某个广播组，之后就可以向这个广播组发送数据或者从广播组接收数据。此选项的值为mreq结构，成员imn_multiaddr是需要加入的广播组IP地址，成员imr_interface是本机需要加入广播组的网络接口IP地址。例如：

点击(此处)折叠或打开

```
1. struct  ip_mreq mreq;
2. setsockopt(s,IPPROTO_IP,IP_ADD_MEMBERSHIP,&mreq,sizeof(mreq))
```

使用IP_ADD_MEMBERSHIP选项每次只能加入一个网络接口的IP地址到多播组，但并不是一个多播组仅允许一个主机IP地址加入，可以多次调用IP_ADD_MEMBERSHIP选项来实现多个IP地址加入同一个广播组，或者同一个IP地址加入多个广播组。当imr_ interface为INADDR_ANY时，选择的是默认组播接口。

4．选项IP_DROP_MEMBERSHIP

选项IP_DROP_MEMBERSHIP用于从一个广播组中退出。例如：

点击(此处)折叠或打开

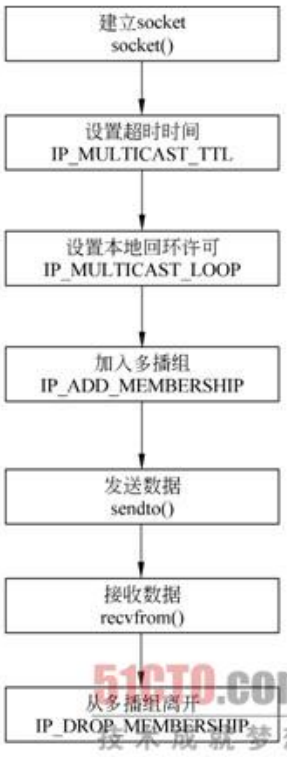
```
1. struct  ip_mreq mreq;
2. setsockopt(s,IPPROTP_IP,IP_DROP_MEMBERSHIP,&mreq,sizeof(sreq))
```

其中mreq包含了在IP_ADD_MEMBERSHIP中相同的值。

5．多播程序设计的框架

要进行多播的编程，需要遵从一定的编程框架，其基本顺序如图11.6所示。

（点击查看大图）
图11.6 多播的编程流程



多播程序框架主要包含套接字初始化、设置多播超时时间、加入多播组、发送数据、接收数据以及从多播组中离开几个方面。其步骤如下：

（1）建立一个socket。

（2）然后设置多播的参数，例如超时时间TTL、本地回环许可LOOP等。

（3）加入多播组。

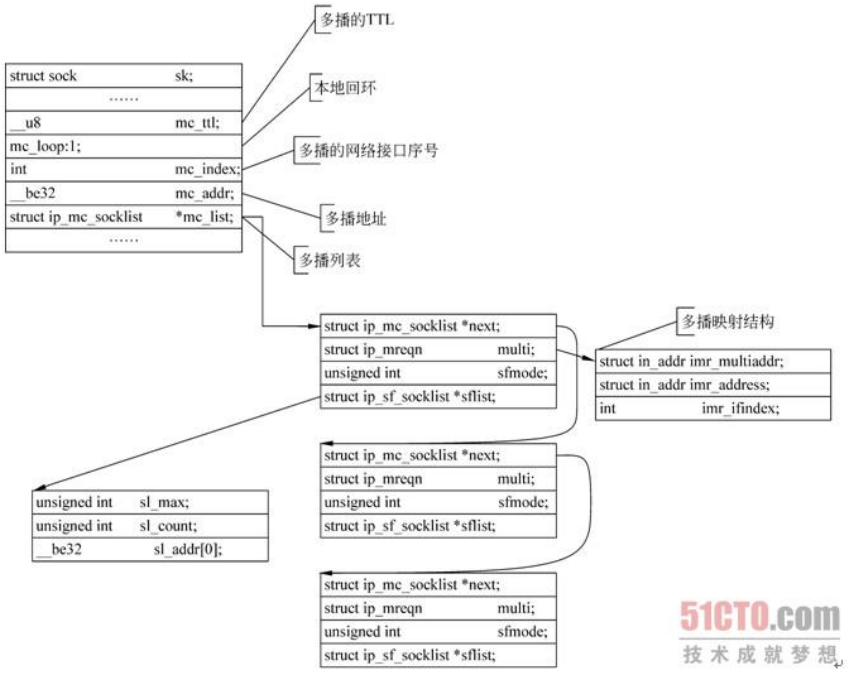
（4）发送和接收数据。

（5）从多播组离开。

11.3.4　内核中的多播

Linux内核中的多播是利用结构struct ip_mc_socklist来将多播的各个方面连接起来的，其示意图如图11.7所示。

（点击查看大图）
图11.7 多播的内核结构



点击(此处)折叠或打开

```
1. struct  inet_sock {
2.         ...
3.         __u8 mc_ttl;  /*多播TTL*/
4.         ...
5.         __u8  ...
6.                                                mc_loop:1;  /*多播回环设置*/
7.         int  mc_index;  /*多播设备序号*/
8.         __be32 mc_addr;  /*多播地址*/
9.         struct  ip_mc_socklist  *mc_list;  /*多播群数组*/
10.         ...
11. }
```

结构成员mc_ttl用于控制多播的TTL；

结构成员mc_loop表示是否回环有效，用于控制多播数据的本地发送；

结构成员mc_index用于表示网络设备的序号；

结构成员mc_addr用于保存多播的地址；

结构成员mc_list用于保存多播的群组。

1. 结构ip_mc_socklist

结构成员mc_list的原型为struct ip_mc_socklist，定义如下：

点击(此处)折叠或打开

```
1. struct  ip_mc_socklist
2. {
3.         struct  ip_mc_socklist  *next;
4.         struct  ip_mreqn multi;
5.         unsigned  int  sfmode;  /*MCAST_{INCLUDE,EXCLUDE}*/
6.         struct  ip_sf_socklist  *sflist;
7. }
```

成员参数next指向链表的下一个节点。

成员参数multi表示组信息，即在哪一个本地接口上，加入到哪一个多播组。

成员参数sfmode是过滤模式，取值为 MCAST_INCLUDE或MCAST_EXCLUDE，分别表示只接收sflist所列出的那些源的多播数据报，和不接收sflist所列出的那些源的多播数据报。

成员参数sflist是源列表。

2．结构ip_mreqn

multi成员的原型为结构struct ip_mreqn，定义如下：

点击(此处)折叠或打开

```
1. struct  ip_mreqn
2. {
3.         struct  in_addr imr_multiaddr;  /*多播组的IP地址*/
4.         struct  in_addr imr_address;  /*本地址网络接口的IP地址*/
5.         int  imr_ifindex;  /*网络接口序号*/
6. }
```

该结构体的两个成员分别用于指定所加入的多播组的组IP地址，和所要加入组的那个本地接口的IP地址。该命令字没有源过滤的功能，它相当于实现IGMPv1的多播加入服务接口。

3．结构ip_sf_socklist

成员sflist的原型为结构struct ip_sf_socklist，定义如下：

点击(此处)折叠或打开

```
1. struct  ip_sf_socklist
2. {
3.         unsigned  int  sl_max;  /*当前sl_addr数组的最大可容纳量*/
4.         unsigned  int  sl_count;  /*源地址列表中源地址的数量*/
5.         __u32 sl_addr[0];  /*源地址列表*/
6. }
```

成员参数sl_addr表示是源地址列表；

成员参数sl_count表示是源地址列表中源地址的数量；

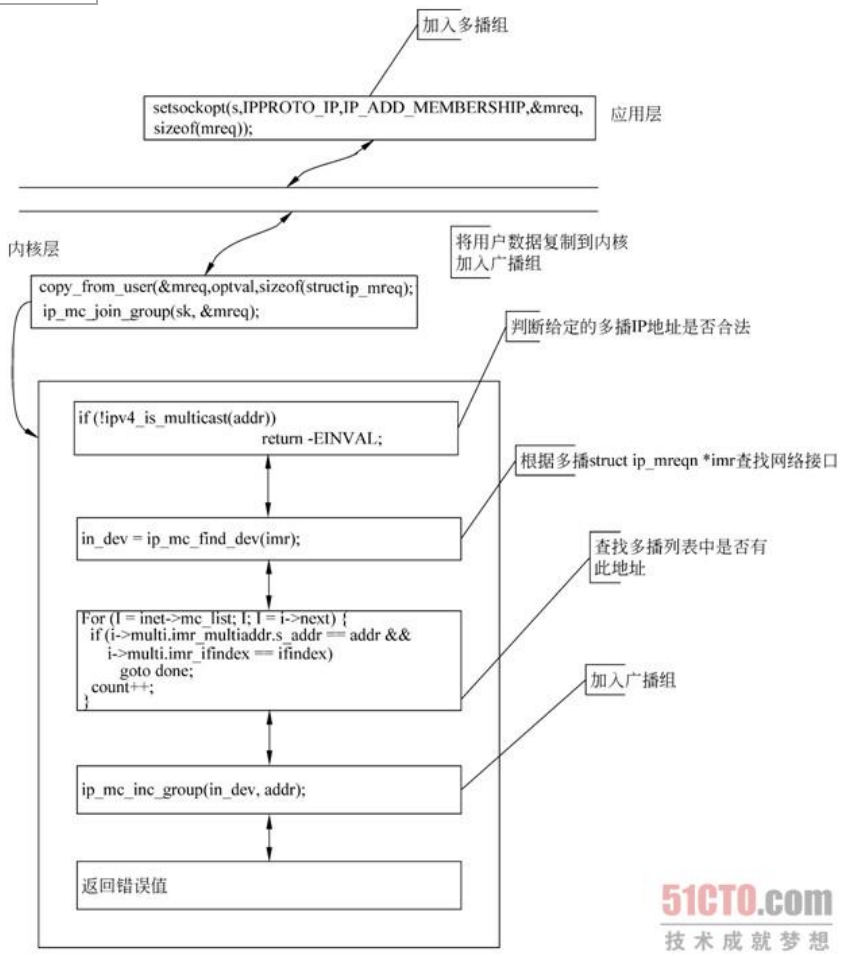
成员参数sl_max表示是当前sl_addr数组的最大可容纳量（不确定）。

4．选项IP_ADD_MEMBERSHIP

选项IP_ADD_MEMBERSHIP用于把一个本地的IP地址加入到一个多播组，在内核中其处理过程如图11.8所示，在应用层调用函数setsockopt()函数的选项IP_ADD_MEMBE- RSHIP后，内核的处理过程如下，主要调用了函数ip_mc_join_group()。

（点击查看大图）
图11.8 选项IP_ADD_MEMBERSHIP的内

核处理过程



（1）将用户数据复制如内核。

（2）判断广播IP地址是否合法。

（3）查找IP地址对应的网络接口。

（4）查找多播列表中是否已经存在多播地址。

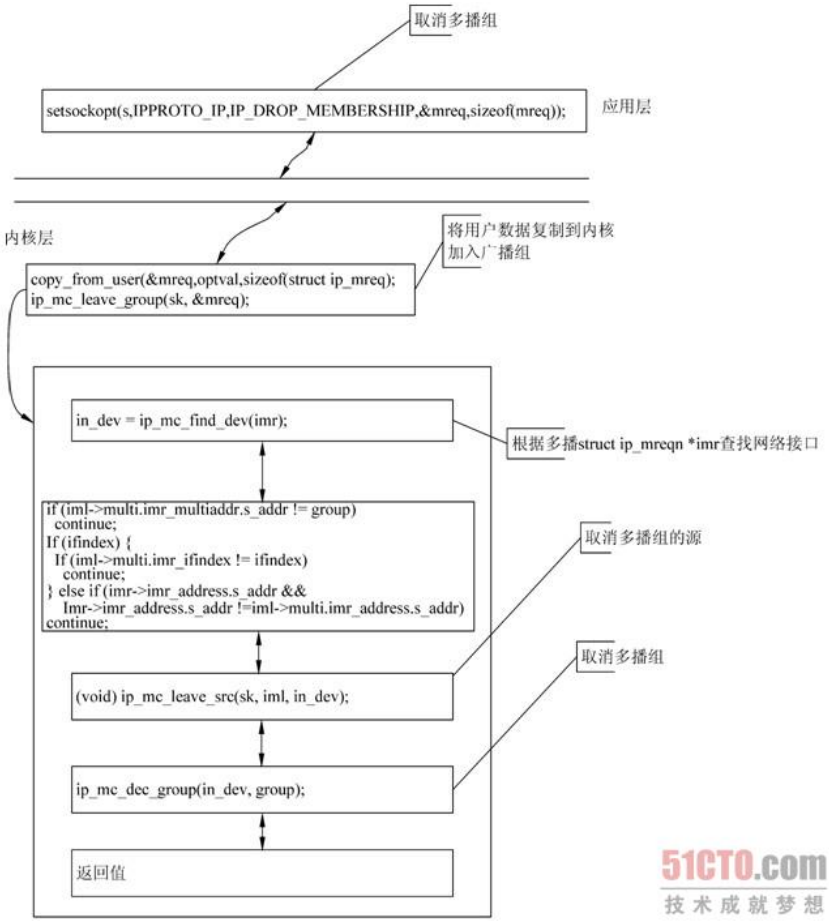
（5）将此多播地址加入列表。

（6）返回处理值。

5．选项IP_DROP_MEMBERSHIP

选项IP_DROP_MEMBERSHIP用于把一个本地的IP地址从一个多播组中取出，在内核中其处理过程如图11.9所示，在应用层调用setsockopt()函数的选项IP_DROP_ MEMBERSHIP后，内核的处理过程如下，主要调用了函数ip_mc_leave_group()。

（点击查看大图）
图11.9 选项IP_DROP_MEMBERSHIP的内核处理过程



（1）将用户数据复制入内核。

（2）查找IP地址对应的网络接口。

（3）查找多播列表中是否已经存在多播地址。

（4）将此多播地址从源地址中取出。

（5）将此地址结构从多播列表中取出。

（6）返回处理值。

11.3.5　一个多播例子的服务器端

下面是一个多播服务器的例子。多播服务器的程序设计很简单，建立一个数据包套接字，选定多播的IP地址和端口，直接向此多播地址发送数据就可以了。多播服务器的程序设计，不需要服务器加入多播组，可以直接向某个多播组发送数据。

下面的例子持续向多播IP地址"224.0.0.88"的8888端口发送数据"BROADCAST TEST DATA"，每发送一次间隔5s。

点击(此处)折叠或打开

```
1. /*
2. *broadcast_server.c  -  多播服务程序
3. */
4. #define MCAST_PORT 8888;
5. #define MCAST_ADDR  "224.0.0.88"/  /*一个局部连接多播地址，路由器不进行转发*/
6. #define MCAST_DATA  "BROADCAST TEST DATA"  /*多播发送的数据*
7. #define MCAST_INTERVAL 5 /*发送间隔时间*/
8. int  main(int  argc,  char*argv)
```

```
{
        int  s;
        struct  sockaddr_in mcast_addr;
        s  =  socket(AF_INET,  SOCK_DGRAM,  0);  /*建立套接字*/
        if  (s  ==  -1)
        {
                perror("socket()");
                return  -1;
        }

        memset(&mcast_addr,  0,  sizeof(mcast_addr));/*初始化IP多播地址为0*/
        mcast_addr.sin_family  =  AF_INET;  /*设置协议族类行为AF*/
        mcast_addr.sin_addr.s_addr  =  inet_addr(MCAST_ADDR);/*设置多播IP地址*/
        mcast_addr.sin_port  =  htons(MCAST_PORT);  /*设置多播端口*/


    /*向多播地址发送数据*/
        while(1)  {
                int  n  =  sendto(s,  /*套接字描述符*/
                                                                MCAST_DATA,  /*数据*/
                                                                sizeof(MCAST_DATA),  /*长
度*/
                                                                0,
                                                        (struct  sockaddr*)&mcast_add
r,
                                                                sizeof(mcast_addr))  ;
                if(  n  <  0)
                {
                        perror("sendto()");
                        return  -2;
                }

                sleep(MCAST_INTERVAL);  /*等待一段时间*/
        }

        return 0;
}
```

11.3.6　一个多播例子的客户端

多播组的IP地址为224.0.0.88，端口为8888，当客户端接收到多播的数据后将打印出来。

客户端只有在加入多播组后才能接受多播组的数据，因此多播客户端在接收多播组的数据之前需要先加入多播组，当接收完毕后要退出多播组。

点击(此处)折叠或打开

```
/*
*broadcast_client.c  -  多播的客户端
*/
#define MCAST_PORT 8888;
#define MCAST_ADDR  "224.0.0.88"  /*一个局部连接多播地址，路由器不进行转发*/
#define MCAST_INTERVAL 5 /*发送间隔时间*/
#define BUFF_SIZE 256 /*接收缓冲区大小*/
int  main(int  argc,  char*argv[])
{
        int  s;  /*套接字文件描述符*/
        struct  sockaddr_in local_addr;  /*本地地址*/
```

```
12.         int  err  =  -1;
13.
14.         s  =  socket(AF_INET,  SOCK_DGRAM,  0);  /*建立套接字*/
15.         if  (s  ==  -1)
16.         {
17.                 perror("socket()");
18.                 return  -1;
19.         }
20.
21.                                                                                 /*初
始化地址*/
22.         memset(&local_addr,  0,  sizeof(local_addr));
23.         local_addr.sin_family  =  AF_INET;
24.         local_addr.sin_addr.s_addr  =  htonl(INADDR_ANY);
25.         local_addr.sin_port  =  htons(MCAST_PORT);
26.
27.                                                                                 /*绑
定socket*/
28.         err  =  bind(s,(struct  sockaddr*)&local_addr,  sizeof(local_addr))  ;
29.         if(err  <  0)
30.         {
31.                 perror("bind()");
32.                 return  -2;
33.         }
34.
35.                                                                                 /*设
置回环许可*/
36.         int  loop  =  1;
37.         err  =  setsockopt(s,IPPROTO_IP,  IP_MULTICAST_LOOP,&loop,  sizeof(loop));
38.         if(err  <  0)
39.         {
40.                 perror("setsockopt():IP_MULTICAST_LOOP");
41.                 return  -3;
42.         }
43.
44.         struct  ip_mreq mreq;  /*加入广播组*/
45.         mreq.imr_multiaddr.s_addr  =  inet_addr(MCAST_ADDR);  /*广播地址*/
46.         mreq.imr_interface.s_addr  =  htonl(INADDR_ANY);  /*网络接口为默认*/
47.
            /*将本机加入广播组*/
48.         err  =  setsockopt(s,  IPPROTO_IP,  IP_ADD_MEMBERSHIP,&mreq,  sizeof
49.         (mreq));
50.         if  (err  <  0)
51.         {
52.                 perror("setsockopt():IP_ADD_MEMBERSHIP");
53.                 return  -4;
54.         }
55.
56.         int  times  =  0;
57.         int  addr_len  =  0;
58.         char buff[BUFF_SIZE];
59.         int  n  =  0;
60.                                                                                 /*循环接收广播组
的消息，5次后退出*/
61.         for(times  =  0;times<5;times++)
62.         {
63.                 addr_len  =  sizeof(local_addr);
64.                 memset(buff,  0,  BUFF_SIZE);  /*清空接收缓冲区*/
65.
    /*接收数据*/
66.                 n  =  recvfrom(s,  buff,  BUFF_SIZE,  0,(struct  sockaddr*)&local_addr,
```

```
67.                 &addr_len);
68.                 if(  n==  -1)
69.                 {
70.                         perror("recvfrom()");
71.                 }
72.
```

/*打印信息*/

```
73.                 printf("Recv %dst message from server:%s\n",  times,  buff);
74.                 sleep(MCAST_INTERVAL);
75.         }
76.
77.
```

/*退出广播组*/

```
78.         err  =  setsockopt(s,  IPPROTO_IP,  IP_DROP_MEMBERSHIP,&mreq,  sizeof
79.         (mreq));
80.
81.         close(s);
82.         return 0;
83. }
```