三、基于IP多播的网络会议程序

课程设计题目

参照附录3的局域网IP多播程序，设计一个网络会议程序（实现文本多播方式即可）。

课程设计内容

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。

多播，也称为“组播”，将局域网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。

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。

课程设计结果及结果分析 心得体会

多播主机分为三个级别：

```
0级：主机不能发送或接收I P多播。
```

这种主机应该自动丢弃它收到的具有D类目的地址的分组。

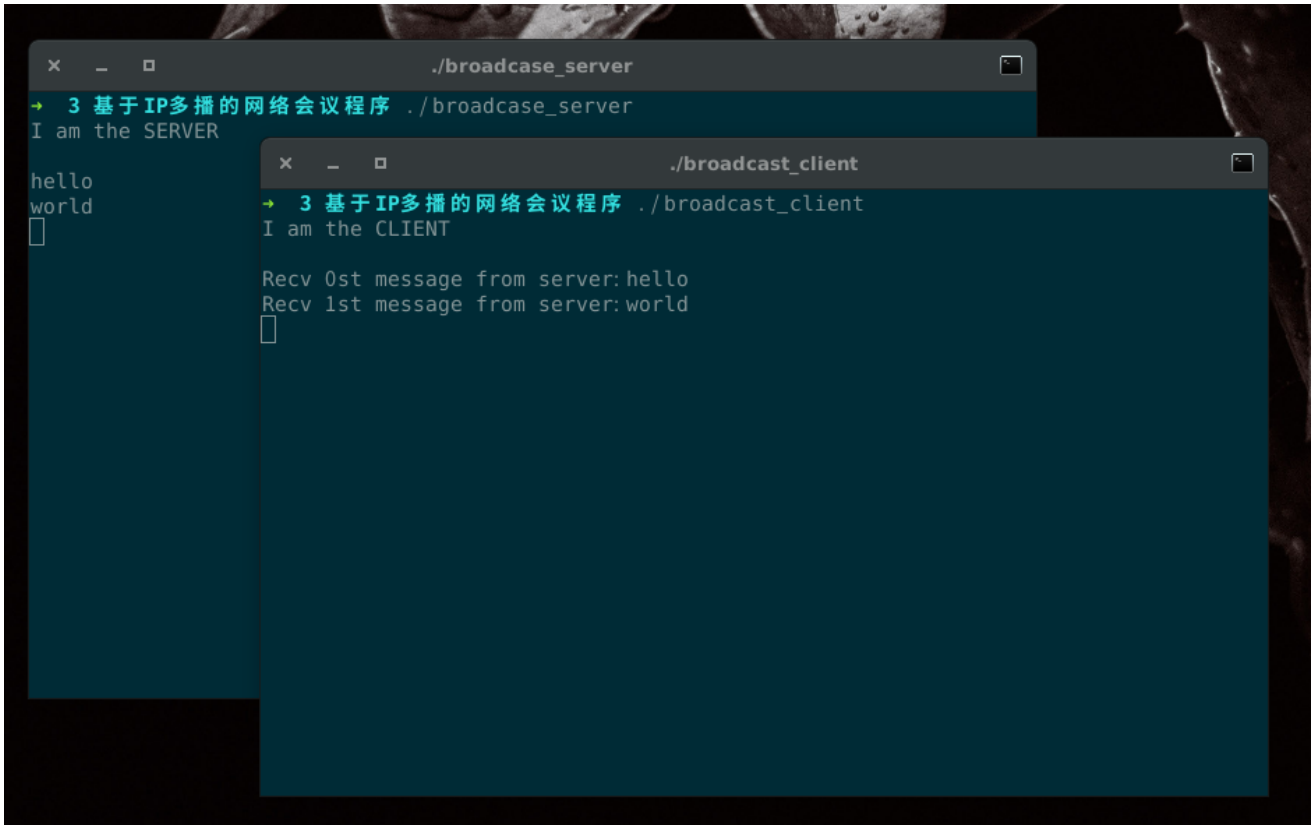
```
1级：主机能发送但不能接收I P多播。
```

在向某个I P多播组发送数据报之前，并不要求主机加入该组。多播数据报的发送方式与单播一样，除了多播数据报的目的地址是 I P多播组之外。网络驱动器必须能 够识别出这个地址，把在本地网络上多播数据报。

```
2级：主机能发送和接收I P多播。
```

程序源文件主要代码展示

程序运行截图



程序主要代码展示

`Server.cpp`

```
int main()
{
    printf("I am the SERVER\n\n");

    int s;
    struct sockaddr_in mcast_addr;
    s = socket(AF_INET, SOCK_DGRAM, 0);
    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;
    mcast_addr.sin_addr.s_addr = inet_addr(MCAST_ADDR);
    mcast_addr.sin_port = htons(MCAST_PORT);

    while (1)
    {
        char buff[BUFF_SIZE];
        gets(buff);
        // int n = sendto(s, MCAST_DATA, sizeof(MCAST_DATA), 0, (struct
sockaddr*)&mcast_addr, sizeof(mcast_addr));
        int n = sendto(s, buff, BUFF_SIZE, 0, (struct sockaddr*)&mcast_addr,
sizeof(mcast_addr));
        if (n < 0)
```

```
        {
            perror("sendto()");
            return -2;
        }
        memset(buff, '0', BUFF_SIZE);

    }

    return 0;
}
```

Client.cpp

```
int main()
{
    printf("I am the CLIENT\n\n");

    int s;
    struct sockaddr_in local_addr;
    int err = -1;

    // create socket
    s = socket(AF_INET, SOCK_DGRAM, 0);
    if (s == -1)
    {
        perror("socket()");
        return -1;
    }

    // fill with 0
    memset(&local_addr, 0, sizeof(local_addr));
    local_addr.sin_family = AF_INET;
    local_addr.sin_addr.s_addr = htonl(INADDR_ANY); // INADDR_ANY:回送到默认接口
    local_addr.sin_port = htons(MCAST_PORT);

    err = bind(s, (struct sockaddr*)&local_addr, sizeof(local_addr));
    if (err < 0)
    {
        perror("bind()");
        return -2;
    }

    // 设置回环许可
    int loop = 1; // 1: 允许数据回送到本地的回环接口
    err = setsockopt(s, IPPROTO_IP, IP_MULTICAST_LOOP, &loop, sizeof(loop));
    if (err < 0)
    {
        perror("setsockopt():IP_MULTICAST_LOOP");
        return -3;
    }

    // 加入多播祖
    struct ip_mreq mreq;
```

```
    mreq.imr_multiaddr.s_addr = inet_addr(MCAST_ADDR);
    mreq.imr_interface.s_addr = htonl(INADDR_ANY); // 网络接口为默认

    err = setsockopt(s, IPPROTO_IP, IP_ADD_MEMBERSHIP, &mreq, sizeof(mreq));
    if (err < 0)
    {
        perror("setsockopt():IP_ADD_MEMBErSHIP");
        return -4;
    }

    int times = 0;
    int addr_len = 0;
    char buff[BUFF_SIZE];
    int n = 0;
    for (times = 0; times < 5; times++)
    {
        addr_len = sizeof(local_addr);
        memset(buff, 0, BUFF_SIZE); // 清空接收缓冲区
        n = recvfrom(s, buff, BUFF_SIZE, 0, (struct sockaddr*)&local_addr, &addr_len);
        if (n == -1)
        {
            perror("recvfrom()");
        }

        printf("Recv %dst message from server:%s\n", times, buff);
        // sleep(MCAST_INTERVAL);
    }

    err = setsockopt(s, IPPROTO_IP, IP_DROP_MEMBERSHIP, &mreq, sizeof(mreq));

    close(s);
    return 0;
}
```