# Linux平台Socket基础应用-UDP通信程序

实验名称：Linux平台Socket基础应用-UDP通信程序

学时安排：2课时 指导老师：李赞

实验类别：验证型、设计型 实验要求：1人1组

学号： 姓名： 班级：

## 一、实验目的和任务

1. 本实验要求复习ubuntu的IP配置知识。

2. 了解的掌握使用socket编程的基本方法。

## 二、实验设备介绍

1.软件需求： win10操作系统，VMware workstation，ubuntu18。

## 三、注意事项和要求

1.启动虚拟机前应首先使用"虚拟网络编辑器"配置要桥接的网卡。

2.测试虚拟机是否联网正常。

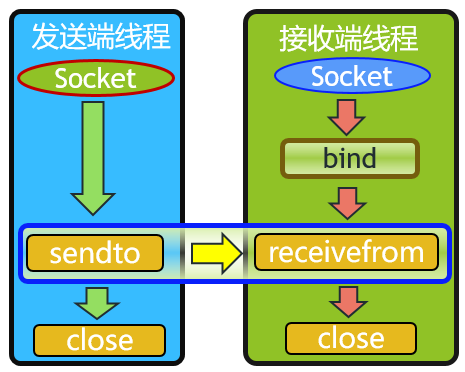
3.调试程序时需要两个程序同时运行进行联调。

## 四、实验内容和步骤

本次实验使用UDP协议实现socket网络通讯。

### 4.1 UDP通讯简介

UDP通讯模式如下图所示，只要实现绑定操作的socket资源就可以执行接收任务。UDP没有连接的概念，也没有断开连接的需要，直接调用sendto方法即可实现信息发送，接收端使用receivefrom方法接收数据。UDP通讯中使用的方法是socket, bind, sendto, receivefrom, close。UDP是常用的无连接的网络协议，SNMP网络管理的协议就是基于UDP的应用层协议。



UDP数据的发送和接收

### 4.2发送端代码uecho\_client.c

UDP数据发送端，程序从键盘接收字符输入，再通过UDP方法发送到目标机器。

//023.c udp数据发送端

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>//close方法

#include <arpa/inet.h>

#include <sys/socket.h>

#define BUF\_SIZE 30

void error\_handling(char \* message);

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

{

int sock;

char message[BUF\_SIZE];

int str\_len;

socklen\_t adr\_sz;

struct sockaddr\_in serv\_adr,from\_adr;

//命令 IP PORT

if(argc!=3)

{

printf("Usage :%s <IP> <PORT>\n",argv[0]);

exit(1);

}

sock=socket(PF\_INET,SOCK\_DGRAM,0);

if(sock==-1)

error\_handling("UDP socket() create failed.");

memset(&serv\_adr,0,sizeof(serv\_adr));

serv\_adr.sin\_family=AF\_INET;

//IP地址

serv\_adr.sin\_addr.s\_addr=inet\_addr(argv[1]);

//PORT

serv\_adr.sin\_port=htons(atoi(argv[2]));

while(1)

{

fputs("Insert message(q to quit):",stdout);

memset(message,0, BUF\_SIZE);

fgets(message,sizeof(message),stdin);

if(!strcmp(message,"q\n")||!strcmp(message,"Q\n"))

break;

sendto(sock,message,strlen(message),0,

(struct sockaddr\*)&serv\_adr,sizeof(serv\_adr));

adr\_sz=sizeof(from\_adr);

str\_len=recvfrom(sock,message,BUF\_SIZE,0,

(struct sockaddr\*)&from\_adr,&adr\_sz);

message[str\_len]=0;

printf("Message from server: %s",message);

}

close(sock);

return 0;

}

//错误处理是将信息输出到标准输出，然后退出进程。

void error\_handling(char \* message)

{

fputs(message,stderr);

fputc('\n',stderr);

exit(1);

}

### 4.3接收端代码uecho\_server.c

本小节示例代码的UDP服务端是接收发送端的数据，并直接原样sendto返还，该程序将无法退出。

//024.c udp数据接收，并进行回显

//头文件与uecho\_ client.c一样

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

{

int serv\_sock;

char message[BUF\_SIZE];

int str\_len;

socklen\_t clnt\_adr\_sz;

struct sockaddr\_in serv\_adr,clnt\_adr;

//命令 PORT

if(argc!=2)

{

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

exit(1);

}

serv\_sock=socket(PF\_INET,SOCK\_DGRAM,0);

if(serv\_sock==-1)

error\_handling("UDP socket() create failed.");

memset(&serv\_adr,0,sizeof(serv\_adr));

serv\_adr.sin\_family=AF\_INET;

//IP地址

serv\_adr.sin\_addr.s\_addr=htonl(INADDR\_ANY);

//PORT

serv\_adr.sin\_port=htons(atoi(argv[1]));

if(bind(serv\_sock,(struct sockaddr \*)&serv\_adr,sizeof(serv\_adr))==-1)

error\_handling("UDP bind() failed.");

while(1)

{

clnt\_adr\_sz=sizeof(clnt\_adr);

str\_len=recvfrom(serv\_sock,message,BUF\_SIZE,0,

(struct sockaddr\*)&clnt\_adr,&clnt\_adr\_sz);

printf("%s",message);

memset(message,0, BUF\_SIZE);

scanf("%s",message);

sendto(serv\_sock,message,str\_len,0,

(struct sockaddr\*)&clnt\_adr,clnt\_adr\_sz);

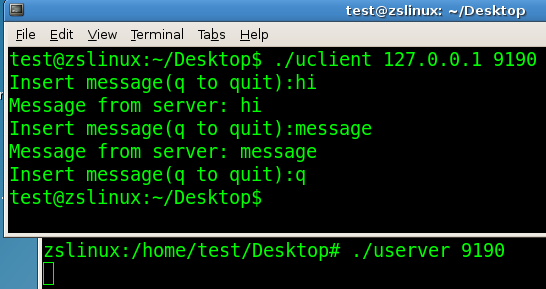
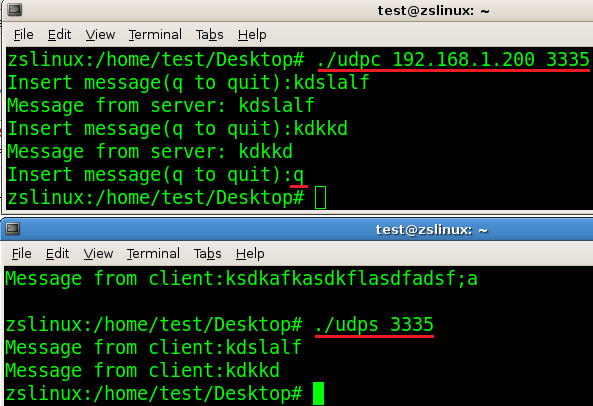
}

close(serv\_sock);

return 0;

}

使用gcc工具编译程序后运行UDP客户端与服务端在Debian系统的效果如下图，注意服务端程序不会有输出，也不终止。试修改服务端代码使服务端程序可以正常结束[非kill]。

左面是使服务端正常结束的截图，客户端输入q字符后，两个进程都结束。

## 五、程序调试中遇到的问题和解决过程及运行结果

1.将上述程序使用gcc 编译并运行，分析过程中遇到问题，对程序进行排错。

2.启动两个命令窗口分别运行发送端进程与接收端进程，将两个程序正常运行的结果截图及源代码截屏上传到学习通平台的作业中。

3.测试两台不同主机间程序的UDP通信程序的运行。