## 一、概述

UDP 是 User Datagram Protocol 的简称， 中文名是用户数据报协议，是一种[无连接的协议](http://blog.csdn.net/tennysonsky/article/details/44455565)。

## 二、**UDP 编程的 C/S**

UDP 客户端程序

ssize\_t sendto(int sockfd, const void \*buf, size\_t nbytes, int flags, const struct sockaddr \*to, socklen\_t addrlen );

功能：

向 to 结构体指针中指定的 ip，发送 UDP 数据，可以发送 0 长度的 UDP 数据包

参数：

sockfd：[套接字](http://blog.csdn.net/tennysonsky/article/details/45047209)

buf：发送数据缓冲区

nbytes：发送数据缓冲区的大小

flags：一般为 0

to：指向目的主机地址结构体的指针

addrlen：to 所指向内容的长度

返回值：

成功：发送数据的长度

失败： -1

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

int main()

{

    unsigned short port = 8080; //服务器端口

    const char \*server\_ip = "127.0.0.1";   //服务器ip地址

    int sockfd = socket(AF\_INET, SOCK\_DGRAM, 0);   //创建UDP套接字

    if(sockfd < 0)

    {

        // 创建socket失败

        perror("socket");

        exit(-1);

    }

    // 指定服务器套接字地址

    struct sockaddr\_in dest\_addr;

    bzero(&dest\_addr, sizeof(dest\_addr));   // 清空内容

    dest\_addr.sin\_family = AF\_INET;     // ipv4

    dest\_addr.sin\_port   = htons(port); // 端口转换

inet\_pton(AF\_INET, server\_ip, &dest\_addr.sin\_addr); // ip地址转换

    printf("send data to UDP server %s:%d!\n", server\_ip, port);

    while(1)

    {

        char send\_buf[512] = "";

        fgets(send\_buf, sizeof(send\_buf), stdin);   //获取输入

        send\_buf[strlen(send\_buf)-1] = '\0';            // 数组不会以\0结尾，需要手动添加

        //发送数据

        int len = sendto(sockfd, send\_buf, strlen(send\_buf), 0, (struct sockaddr\*)&dest\_addr, sizeof(dest\_addr));

        printf("len = %d\n", len);

    }

    close(sockfd);

    return 0;

}

UDP 客户端**注意点**：

1）本地IP、本地端口（我是谁）

2）目的IP、目的端口（发给谁）

3）在客户端的代码中，我们只设置了目的IP、目的端口

4）客户端的本地 ip、本地 port 是我们调用 sendto 的时候 linux 系统底层自动给客户端分配的；分配端口的方式为随机分配

## 三、UDP 服务器程序

UDP网络程序想要收取数据需什么条件？

1）确定的 ip 地址

2）确定的端口（port）

接收端使用 bind() 函数，来完成地址结构与 socket 套接字的绑定，这样 ip、port 就固定了，发送端在 sendto 函数中指定接收端的 ip、port，就可以发送数据了。

需要头文件：#include <sys/socket.h>

int bind( int sockfd, const struct sockaddr \*myaddr, socklen\_t addrlen );

功能：

将本地协议地址与 sockfd 绑定，这样 ip、port 就固定了

参数：

sockfd： [socket 套接字](http://blog.csdn.net/tennysonsky/article/details/45047209)

myaddr： 指向特定协议的地址结构指针

addrlen：该地址结构的长度

返回值：

成功：返回 0

失败：-1

ssize\_t recvfrom( int sockfd, void \*buf,  size\_t nbytes,int flags,struct sockaddr \*from, socklen\_t \*addrlen );

功能：

接收 UDP 数据，并将源地址信息保存在 from 指向的结构中

参数：

sockfd:套接字

buf：接收数据缓冲区

nbytes:接收数据缓冲区的大小

flags：套接字标志（常为 0）

from：源地址结构体指针，用来保存数据的来源

addrlen：from 所指内容的长度

返回值：

成功：接收到的长度

失败： -1

#include <iostream>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

int main()

{

unsigned short port = 8080;     // 本地端口

    int sockfd = socket(AF\_INET, SOCK\_DGRAM, 0);    // 创建套接字

    if(sockfd < 0)

    {

        // 创建套接字失败

        std::cout << "create socket fail" << std::endl;

        return -1;

    }

    // 本地网络地址

    struct sockaddr\_in my\_addr;

    bzero(&my\_addr,  sizeof(my\_addr));   // 清空结构体内容

    my\_addr.sin\_family = AF\_INET;   // ipv4

    my\_addr.sin\_port   = htons(port);   // 端口转换，端口需要转成网络字节序

    my\_addr.sin\_addr.s\_addr = htonl(INADDR\_ANY); // 绑定网卡所有ip地址，INADDR\_ANY为通配地址，值为0

    std::cout << "Binding server to port" << port << std::endl;

    // 绑定套接字

    int err\_log = bind(sockfd, (struct sockaddr\*)&my\_addr, sizeof(my\_addr));

    if(err\_log != 0)

    {

        // 绑定失败

        std::cout << "bind fail" << std::endl;

        close(sockfd);

        exit(-1);

    }

    std::cout << "receive data..." << std::endl;

    while(true)

    {

        char recv\_buf[512] = "";

        struct sockaddr\_in client\_addr;

        socklen\_t cliaddr\_len = sizeof(client\_addr);

        // 接受数据

        int recv\_len = recvfrom(sockfd, recv\_buf, sizeof(recv\_buf), 0, (struct sockaddr\*)&client\_addr, &cliaddr\_len);

        char cli\_ip[INET\_ADDRSTRLEN] = "";      //INET\_ADDRSTRLEN=16

        inet\_ntop(AF\_INET, &client\_addr.sin\_addr, cli\_ip, INET\_ADDRSTRLEN);

        std::cout << "ip: " << cli\_ip << ", port: " << ntohs(client\_addr.sin\_port) << std::endl;

        std::cout << "length:" << recv\_len << ", data:" << recv\_buf << std::endl;

    }

    close(sockfd);

    return 0;

}