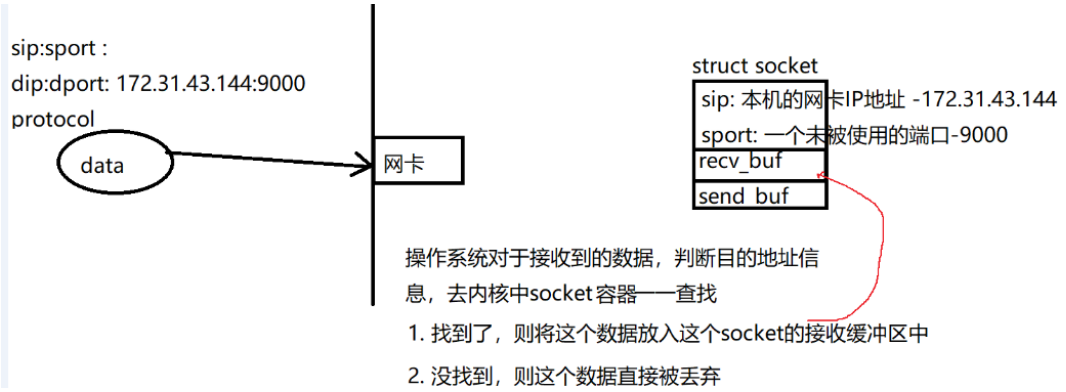
socket套接字编程:
socket是一套网络编程接口，类似于中间件;.上层用户可以通过这些接口简单的完成网络通信传输; 而不需要过于关心内部的实现过程套接字编程讲的就是使用socket接口实现网络通信
socket编程: tcp/udp
传输层有两个协议: tcp/udp; 这两个协议特性各有不同,因此实现流程也稍有差别;因此需要分开来讲
udp:用户数据报协议:无连接，不可靠，面向数据报;应用场景就是数据实时性大于安全性的场景--视频传输
tcp:传输控制协议:面向连接，可靠传输，面向字节流;应用场景就是数据安全性大于实时性的场景--文件传输
面向数据报:无连接的， 不可靠的，无序的，有最大长度限制的数据传输服务
面向字节流:基于连接的，可靠的，有序的，双向的字节流传输服务　不限制上层传输数据大小的传输方式

网络通信，是网络中的两端主机上的进程之间的通信; 这两端有个名称:客户端/服务器端
客户端:是主动发出请求的一方主机
服务器端:是被动接收请求的一方主机
永远都是客户端主机先向服务器端发送请求
udp网络通信程序编程流程:
client：
1.创建套接字
2.为套接字绑定地址信息
3.发送数据
4.接收数据
5.关闭套接字
server：
1.创建套接字--在内核中创建socket结构体;向进程返回一个操作句柄;通过这个内核中的socket结构体与网卡建立联系
2.为套接字绑定地址信息--向内核中创建的socket结构体添加各种地址描述信息(ip地址和端口)
绑定地址就是为了告诉操作系统，我使用了哪个地址和端口你接收到了数据，若目的地址信息和我绑定的地址信息相同，则将数据交给我处理
并且发送数据的时候，源端地址信息就是绑定的地址信息
3.接收数据--socket结构体中的接收缓冲区中取出数据
每个数据中都包含源地址和目的地址，因此获取数据也就获悉了对端是谁
4.发送数据--把数据拷贝到内核中的socket结构体的发送缓冲区中
操作系统这时候会在合适的时候从发送缓冲区中取出数据，然后进行数据的层层封装,最终通过网卡发送出去
5.不通信了，关闭套接字,释放资源



服务端socket只能绑定的是服务端主机上的IP地址;客户端也绑定的是自己主机上的IP地址

客户端永远都是主动发送数据的一方,意味着客户端必须知道服务端的地址信息才可以在发送数据的时候，将数据能够层层数据封装完成(网络传输的数据都应该包含:源IP地址/目的IP地址源端口/目的端口/协议)

客户端所知道的服务端地址，都是服务端告诉它的---服务器的地址通常都是永久不变的

socket接口的介绍:

1.创建套接字:

int socket(int domain, int type, int protocol);

domain:地址域---不同的网络地址结构　　AF_ INET - IPv4地址域

type:套接字类型-流式套接字/数据报套接字

　流式套接字: 一种有序的，可靠的，双向的，基于连接的字节流传输　　SOCK_ STREAM

　数据报套接字:无连接的，不可靠的，有最大长度限制的传输　　SOCK_ DGRAM

protocol:使用的协议　0--不同套接字类型下的默认协议;流式套接字默认是tcp/数据报套接字默认是udp

IPPROTO _TCP - tcp协议　　IPPROTO_ UDP - udp协议

返回值:返回套接字的操作句柄---文件描述符

2.为套接字绑定地址信息

int bind(int sockfd, struct sockaddr *addr,socklen_t len);

sockfd:创建套接字返回的操作句柄

addr:要绑定的地址信息结构

len :地址信息的长度

返回值:成功返回0;失败返回-1

用户先定义sockaddr_in的IPV4地址结构,强转之后传入bind之中

```
bind(sockaddr*) {
    if (sockaddr->sin_family == AF_INET){
        这是ipv4地址结构的解析
    }else if (sockaddr->sin_family == AF_INET6)
}
```

bind((struct sockaddr*)&addr)

3.发送数据.

int sendto(int sockfd,char *data,int data_ len,int flag,struct sockaddr *dest_addr,socklen_t addr_ len);

sockfd:套接字操作句柄，发送数据就是将数据拷贝到内核的socket发送缓冲区中

data:要发送的数据的首地址

data_ len:要发送的数据的长度
flag:选项参数　　默认为0----表示当前操作是阻塞操作　MSG_DONTWAIT--设置为非阻塞
若发送数据的时候，socket发送缓冲区已经满了,则0默认阻塞等待; MSG_DONTWAIT就是立即报错返回了
dest_ addr:目的端地址信息结构----表示数据要发送给谁
每一条数据都要描述源端信息(绑定的地址信息)和对端信息(当前赋予的信息)
addr_ len: 地址信息结构长度
返回值:成功返回实际发送的数据字节数;失败返回-1
IP地址和端口的时候说过:网络中的每条数据都需要包含源端信息和目的端信息

4.接收数据
int recvfrom(int sockfd, char *buf, int len, int flag, struct sockaddr *src_ addr, socklen _t *addr len);
sockfd:套接字操作句柄
buf:缓冲区的首地址， 用于存放接收到的数据，从内核socket接收缓冲区中取出数据放入这个buf用户态缓冲区中
len:用户想要读取的数据长度， 但是不能大于buf缓冲区的长度
flag: 0-默认阻塞操作 ---- 若缓冲区中没有数据则一直等待　　MSG_ DONTWAIT　非阻塞
src_ addr:接收到的数据的发送端地址--表示这个数据是谁发的,从哪来的--回复的时候就是对这个地址进行回复
addr_ len:输入输出型参数，用于指定想要获取多长的地址信息;获取地址之后，用于返回地址信息的实际长度
返回值:成功返回实际接收到的数据字节长度:失败返回-1
5.关闭套接字
int close(int fd);

通过所学接口编写
udp客户端程序:使用c++封装一个udpsocket类,向外提供简单接口就能实现一个客户端/服务端
udp服务端程序:使用C语言编写