<https://blog.csdn.net/zz709196484/article/details/54864770>

From: [http://www.it165.NET/pro/html/201211/4066.html](http://www.it165.net/pro/html/201211/4066.html)

在初学的时候对二者甚是迷茫一会就晕了   
在这里总结一句话：

struct sockaddr 这个结构体是linux的网络编程接口中用来表示IP地址的标准结构体，bind、connect等函数中都需要这个结构体，这个结构体是兼容IPV4和IPV6的。**在实际编程中这个结构体会被一个struct sockaddr\_in所填充。**

sockaddr 在bind的man手册中提到

struct sockaddr {

sa\_family\_t sa\_family; //所选协议族AF\_INET

char sa\_data[14]; //ip地址及端口号

}

而sockaddr\_in

{

\_\_SOCKADDR\_COMMON (sin\_);

in\_port\_t sin\_port; /\* Port number. \*/

struct in\_addr sin\_addr; /\* Internet address. \*/

unsigned char sin\_zero[sizeof (struct sockaddr) -

\_\_SOCKADDR\_COMMON\_SIZE -

sizeof (in\_port\_t) -

sizeof (struct in\_addr)];

};

虽然是两个结构体可是二者的占用的内存是一致的，因此可以互相转化。

sockaddr常用于bind、connect、recvfrom、sendto等函数的参数，指明地址信息。是一种通用的套接字地址。

而sockaddr\_in 一般是储存地址和端口的。用于信息的显示及存储使用

例如：

struct sockaddr\_in addr\_server;

addr\_server.sin\_family = AF\_INET;

addr\_server.sin\_port = htons(RPORT);

addr\_server.sin\_addr.s\_addr = inet\_addr(RHOST);

然而，在类似于bind accept的函数中

ret = bind(fd\_sock, (struct sockaddr \*)&addr\_server,

sizeof(addr\_server));

if(ret < 0){

perror("bind");

return -1;

}

之前只是这样的记下来了，可是知道一天，想显示所连接的客户端的ip地址的时候，就发现了问题所在

char \*inet\_ntoa(struct in\_addr in);

函数原型是这样的，可是在

struct in\_addr

{

in\_addr\_t s\_addr;

};

这个in\_addr是sockaddr\_in的一个mamber

fd\_connection = accept(fd\_sock,

(struct sockaddr \*)&addr\_client, &addr\_client\_len);

if(fd\_connection < 0){

perror("accept");

return -1;

}

printf("connected! : %d\n", fd\_connection);

printf("%s%s\n", "the client ip is :",

inet\_ntoa(addr\_client.sin\_addr));

版权声明：本文为博主原创文章，未经博主允许不得转载。 https://blog.csdn.net/zz709196484/article/details/54864770