TCP/IP网络编程(三)

笔记本: 网络编程
创建时间: 2018/10/31 20:08 **更新时间:** 2018/11/9 20:27
作者: xiangkang94@outlook.com
标签: 第三章(地址族与序列数组)

---

1. 网络地址(IP)
2. 端口号
    - 用在同一操作系统内区分不同套接字设置的
    - 同一端口号不能分配给不同套接字
    - TCP/UDP套接字不共用端口号

数据传输目标地址同时包含IP地址和端口号，只有这样，数据才会被传输到最终的目的应用程序(应用程序套接字)。

---

1. 结构体sockaddr_in 和 sockaddr的联系以及区别
    - 调用bind函数的原型

```
struct sockaddr_in serv_addr;
if(bind(serv_sock,(struct sockaddr *) &serv_addr,sizeof(serv_addr))==-1)
    error_handling("bind() error");
```

由此可见，bind的第二个参数期望得到的是sockaddr的结构体的变量地址值，包括地址族，端口号，IP等，而二者的结构定义如下：

```
struct sockaddr_in
{
    sa_family_t    sin_family;
    uint16_t       sin_port;
    struct in_addr sin_addr;
    char           sin_zero;
}

struct sockaddr
{
    sa_family_t    sin_family;
    char           sa_data[14];
}
```

直接向sockaddr结构体填充这些信息很麻烦。sockaddr结构体中成员sa_data保存的地址信息中需要包含IP地址和端口号，剩余部分填0。所以才有sockaddr_in结构体，按照sockaddr_in的结构填写结构体，则生成符合bind函数要求的字节流，最后转换成sockaddr型的结构体变量，再传递给bind函数即可。

网络字节序与地址变换

1. 字节序与网络字节序
    - CPU向内存保存数据的方式有2种
        - 大端序：高位字节存放到地位地址
        - 小端序：高位字节存放到高位地址
    - 为了在通过网络传输数据时能够达成数据一致，**约定网络字节序，统一为大端序。**
2. 字节序转换
    - unsigned short htons(unsigned short); //把short型数据从主机字节序转换为网络字节序
    - unsigned long ntohl(unsigned long); //把long型数据从网络字节序转换为主机字节序
3. 将字符串形式的IP地址转换成32位整型数据
   #inclide <arpa/inet.h>
   in_addr_t inet_addr(const char * string)

   eg: unsign long conv_addr=inet_addr("1.2.3.4")

输出 0x4030201
不仅可以将IP地址转换成32位整数型，还可以检测无效IP，并转换为网络字节序(大端)

4. inet_ntoa: 将32位整型数据转换成字符串IP地址。