# 网络编程基础API

## socket地址API

### 主机字节和网络字节转换API

目前CPU的累加器一次能够装在4个字节，那么这4个字节在内存中的排列顺序影响累加器累加的值，这就是字节序的问题。目前，PC都采用小端字节序。但是，格式化的数据在两台不同字节序的主机之间传递时，接收端必然会错误地解释。解决的办法就是发送端总是将字节序转换为大端字节序，然后发送，接收端知道传过来的是大端字节序，因此接收端总是能正确地接收数据。

#include<netinet/in.h>

//主机字节转换成网络字节

unsigned short int htons(unsigned short int hostshrot);

//网络字节转换成主机字节

unsigned short int ntohs(unsigned short int netshrot);

专用socket地址结构体：

//TCP/IP协议族专用socket地址结构体

struct sockaddr\_in

{

sa\_family\_t sin\_family; //地址族：AF\_INET

u\_int16\_t sin\_port; //端口号，用于网络字节序表示

struct in\_addr sin\_addr; //IPv4地址结构体，是u\_int32\_t s\_addr

};

### IP转换函数

#include <arpa/inet.h>

//将字符串IP转换为整数IP

int inet\_pton(int af, const char\* src,void\* dst);

//将整数IP转换成字符串IP

const char\* inet\_ntop(int af, const void\* dst,socklen\_t cnt);

## 创建socket

#include<sys/types>

#include<sys/socket.h>

/\*

domain：底层协议族类型

type：指定服务类型

protocol：具体的协议

\*/

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

命名socket

//服务器绑定网络地址

int bind(int sockfd, const struct sockeaddr\* addr,socketlen\_t addrlen);

## 监听socket

## 接收连接

## 发起连接

## 关闭连接

## 数据读写

# TCP服务端/客户端开发

# UDP服务端/客户端开发