# 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);

bind函数成功返回0，否则为-1，常见error类型：

EACCES：被绑定的地址收到保护

EADDRINUSE：被绑定的地址正在使用中

## 监听socket

int listen(int sockfd,int backlog);

## 接收连接accept

int accept(int sockfd,struct sockaddr\* addr,socklen\_t\* addrlen);

从listen监听队列中接收一个连接

## 发起连接connect

int connect(int sockfd,const struct sockaddr\* serv\_addr,socklen\_t

addrlen);

connect函数成功返回0，否则为-1，常见error类型：

ECONNEFUSED：目标端口不存在

ETIMEOUT：连接超时

## 关闭连接close/shutdown

int close(int fd);

说明：close并不是真正关闭连接，而是将文件描述符fd的引用数-1，只有当fd引用次数为0时，才会关闭连接。如果想立即关闭连接，可以使用shutdown。

int shutdown(int sockfd,int howto);

shutdown函数中howto参数选项：

SHUT\_RE：关闭读这一半

SHUT\_WR：关闭写这一半

SHUT\_RDWR：同时关闭读和写

## 数据读写recv/send

对文件的读写，系统调用read、write同样适用于socket，但是这里提供了专用于socket的recv和send。

TCP数据读写API：

ssize\_t recv(int sockfd,void\* buf,size\_t len,int flags);

ssize\_t send(int sockfd,cons void\* buf,size\_t len,int flags);

recv函数和send函数的flag参数选项：

MSG\_WAITALL：读操作读取到一定数量字节后返回

MSG\_PEEK：查看读缓冲找那个的数据而不取出

UDP数据读写API：

ssize\_t recvfrom(int sockfd,void\* buf,size\_t len,int flags,struct sockaddr\* src\_addr,socklen\_t\* addrlen);

ssize\_t sendto(int sockfd,cons void\* buf,size\_t len,int flags,const struct sockaddr\* dest\_addr,sockelen\_t addrlen);

因为UDP通信没有连接的概念，所以每次读取数据都需要获取发送端数据的大小。

# TCP服务端/客户端开发

## 流程

## 示例

### 客户端

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <stdio.h>

#include <assert.h>

#include <unistd.h>

#include <string.h>

#include <stdlib.h>

int main(int argc,char\* argv[])

{

if(argc <= 2)

{

printf("please input server ip and port number\r\n");

return -1;

}

const char\* ip = argv[1];

int port = atoi(argv[2]);

struct sockaddr\_in server\_address;

bzero(&server\_address,sizeof(server\_address));

server\_address.sin\_family = AF\_INET;

inet\_pton(AF\_INET,ip,&server\_address.sin\_addr);

server\_address.sin\_port = htons(port);

int sockfd = socket(AF\_INET,SOCK\_STREAM,0);

if(sockfd < 0)

{

printf("socket create failed \r\n");

return -1;

}

if(connect(sockfd,(struct sockaddr\*)&server\_address,sizeof(server\_address))<0)

{

printf("connect failed \r\n");

return -1;

}

else

{

char data[64]="Where are you from ?";

char buffer[64];

int i,len;

for(i=0;i<10;i++)

{

memset(buffer,0x00,sizeof(buffer));

send(sockfd,data,strlen(data),0);

len = recv(sockfd,buffer,sizeof(buffer),0);

sleep(1);

printf("index is %d len is %d : %s\r\n",i,len,buffer);

}

}

close(sockfd);

return 0;

}

### 服务端

# UDP服务端/客户端开发