# socket

# 辅助工具

[sockettest](1-doc/SocketTest3.zip)

用于模拟服务器端或客户端

# 基本类型

## 地址结构

### addrinfo

/usr/include/netdb.h

/\* Structure to contain information about address of a service provider. \*/

struct addrinfo

**{**

int ai\_flags**;** /\* Input flags. \*/

int ai\_family**;** /\* Protocol family for socket. \*/

int ai\_socktype**;** /\* Socket type. \*/

int ai\_protocol**;** /\* Protocol for socket. \*/

socklen\_t ai\_addrlen**;** /\* Length of socket address. \*/

struct sockaddr **\*ai\_addr;** /\* Socket address for socket. \*/

char **\***ai\_canonname**;** /\* Canonical name for service location. \*/

struct addrinfo **\***ai\_next**;** /\* Pointer to next in list. \*/

**};**

### ai\_family

/\* Address families. \*/

#define **AF\_UNSPEC** PF\_UNSPEC

#define AF\_LOCAL PF\_LOCAL

#define AF\_UNIX PF\_UNIX

#define AF\_FILE PF\_FILE

#define **AF\_INET** PF\_INET

#define AF\_AX25 PF\_AX25

#define AF\_IPX PF\_IPX

#define AF\_APPLETALK PF\_APPLETALK

#define AF\_NETROM PF\_NETROM

#define AF\_BRIDGE PF\_BRIDGE

#define AF\_ATMPVC PF\_ATMPVC

#define AF\_X25 PF\_X25

#define **AF\_INET6** PF\_INET6

#define AF\_ROSE PF\_ROSE

#define AF\_DECnet PF\_DECnet

#define AF\_NETBEUI PF\_NETBEUI

#define AF\_SECURITY PF\_SECURITY

#define AF\_KEY PF\_KEY

#define AF\_NETLINK PF\_NETLINK

#define AF\_ROUTE PF\_ROUTE

#define AF\_PACKET PF\_PACKET

#define AF\_ASH PF\_ASH

#define AF\_ECONET PF\_ECONET

#define AF\_ATMSVC PF\_ATMSVC

#define AF\_RDS PF\_RDS

#define AF\_SNA PF\_SNA

#define AF\_IRDA PF\_IRDA

#define AF\_PPPOX PF\_PPPOX

#define AF\_WANPIPE PF\_WANPIPE

#define AF\_LLC PF\_LLC

#define AF\_CAN PF\_CAN

#define AF\_TIPC PF\_TIPC

#define AF\_BLUETOOTH PF\_BLUETOOTH

#define AF\_IUCV PF\_IUCV

#define AF\_RXRPC PF\_RXRPC

#define AF\_ISDN PF\_ISDN

#define AF\_PHONET PF\_PHONET

#define AF\_IEEE802154 PF\_IEEE802154

#define AF\_CAIF PF\_CAIF

#define AF\_ALG PF\_ALG

#define AF\_NFC PF\_NFC

#define AF\_MAX PF\_MAX

### in\_addr

/usr/linux/in.h

/\* Internet address. \*/

struct in\_addr **{**

\_\_be32 s\_addr**;**

**};**

### in6\_addr--【为完成】

### in\_addr\_t

/\* Address to accept any incoming messages. \*/

#define INADDR\_ANY ((unsigned long int) 0x00000000)

/\* Address to send to all hosts. \*/

#define INADDR\_BROADCAST ((unsigned long int) 0xffffffff)

/\* Address indicating an error return. \*/

#define INADDR\_NONE ((unsigned long int) 0xffffffff)

/\* Network number for local host loopback. \*/

#define IN\_LOOPBACKNET 127

/\* Address to loopback in software to local host. \*/

#define INADDR\_LOOPBACK 0x7f000001 /\* 127.0.0.1 \*/

#define IN\_LOOPBACK(a) ((((long int) (a)) & 0xff000000) == 0x7f000000)

/\* Defines for Multicast INADDR \*/

#define INADDR\_UNSPEC\_GROUP 0xe0000000U /\* 224.0.0.0 \*/

#define INADDR\_ALLHOSTS\_GROUP 0xe0000001U /\* 224.0.0.1 \*/

#define INADDR\_ALLRTRS\_GROUP 0xe0000002U /\* 224.0.0.2 \*/

#define INADDR\_MAX\_LOCAL\_GROUP 0xe00000ffU /\* 224.0.0.255 \*/

### sa\_family\_t

/\* Protocol families. \*/

#define PF\_UNSPEC 0 /\* Unspecified. \*/

#define PF\_LOCAL 1 /\* Local to host (pipes and file-domain). \*/

#define PF\_UNIX PF\_LOCAL /\* POSIX name for PF\_LOCAL. \*/

#define PF\_FILE PF\_LOCAL /\* Another non-standard name for PF\_LOCAL. \*/

#define PF\_INET 2 /\* IP protocol family. \*/

#define PF\_AX25 3 /\* Amateur Radio AX.25. \*/

#define PF\_IPX 4 /\* Novell Internet Protocol. \*/

#define PF\_APPLETALK 5 /\* Appletalk DDP. \*/

#define PF\_NETROM 6 /\* Amateur radio NetROM. \*/

#define PF\_BRIDGE 7 /\* Multiprotocol bridge. \*/

#define PF\_ATMPVC 8 /\* ATM PVCs. \*/

#define PF\_X25 9 /\* Reserved for X.25 project. \*/

#define PF\_INET6 10 /\* IP version 6. \*/

#define PF\_ROSE 11 /\* Amateur Radio X.25 PLP. \*/

#define PF\_DECnet 12 /\* Reserved for DECnet project. \*/

#define PF\_NETBEUI 13 /\* Reserved for 802.2LLC project. \*/

#define PF\_SECURITY 14 /\* Security callback pseudo AF. \*/

#define PF\_KEY 15 /\* PF\_KEY key management API. \*/

#define PF\_NETLINK 16

#define PF\_ROUTE PF\_NETLINK /\* Alias to emulate 4.4BSD. \*/

#define PF\_PACKET 17 /\* Packet family. \*/

#define PF\_ASH 18 /\* Ash. \*/

#define PF\_ECONET 19 /\* Acorn Econet. \*/

#define PF\_ATMSVC 20 /\* ATM SVCs. \*/

#define PF\_RDS 21 /\* RDS sockets. \*/

#define PF\_SNA 22 /\* Linux SNA Project \*/

#define PF\_IRDA 23 /\* IRDA sockets. \*/

#define PF\_PPPOX 24 /\* PPPoX sockets. \*/

#define PF\_WANPIPE 25 /\* Wanpipe API sockets. \*/

#define PF\_LLC 26 /\* Linux LLC. \*/

#define PF\_CAN 29 /\* Controller Area Network. \*/

#define PF\_TIPC 30 /\* TIPC sockets. \*/

#define PF\_BLUETOOTH 31 /\* Bluetooth sockets. \*/

#define PF\_IUCV 32 /\* IUCV sockets. \*/

#define PF\_RXRPC 33 /\* RxRPC sockets. \*/

#define PF\_ISDN 34 /\* mISDN sockets. \*/

#define PF\_PHONET 35 /\* Phonet sockets. \*/

#define PF\_IEEE802154 36 /\* IEEE 802.15.4 sockets. \*/

#define PF\_CAIF 37 /\* CAIF sockets. \*/

#define PF\_ALG 38 /\* Algorithm sockets. \*/

#define PF\_NFC 39 /\* NFC sockets. \*/

#define PF\_MAX 40 /\* For now.. \*/

### sockaddr

16字节=2+14，一般不直接使用该类型

/\* Structure describing a generic socket address. \*/

struct sockaddr

**{**

unsinged short int sa\_**;**/\* Common data: address family and length.\*/

char sa\_data**[**14**];** /\* Address data. \*/

**};**

### sockaddr\_in

实际路径：/usr/linux/in.h

程序包含路径：netinet/in.h

/\* Structure describing an Internet (IP) socket address. \*/

#define \_\_SOCK\_SIZE\_\_ 16 /\* sizeof(struct sockaddr) \*/

struct sockaddr\_in **{**

\_\_kernel\_sa\_family\_t sin\_family**;** /\* Address family \*/

\_\_be16 sin\_port**;** /\* Port number \*/

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

/\* Pad to size of `struct sockaddr'. \*/

unsigned char \_\_pad**[**\_\_SOCK\_SIZE\_\_ **-** **sizeof(**short int**)** **-**

**sizeof(**unsigned short int**)** **-** **sizeof(**struct in\_addr**)];**

**};**

直观的看，在32位机器上

/\* Structure describing an Internet (IP) socket address. \*/

struct sockaddr\_in **{**

unsigned short sin\_family**;** /\* Address family \*/

unsigned short sin\_port**;** /\* Port number \*/

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

/\* Pad to size of `struct sockaddr'. \*/

unsigned char \_\_pad**[**8**];**

**};**

### sockaddr\_in6--【未完成】

## [fd\_set](2-fd_set/fd_set.docx)

## 其它

### ai\_flags

/usr/include/netdb.h

/\* Possible values for `ai\_flags' field in `addrinfo' structure. \*/

/\* Socket address is intended for `bind'. \*/

套接字将用于被动打开

# define AI\_PASSIVE 0x0001

/\* Request for canonical name. \*/

返回主机的规范名字

# define AI\_CANONNAME 0x0002

/\* Don't use name resolution. \*/

hostname参数必须是一个地址串

# define AI\_NUMERICHOST 0x0004

/\* IPv4 mapped addresses are acceptable. \*/

如果ai\_family=AF\_INET6，如果没有可用的IPv6地址，那么就返回IPv4映射的IPv6地址。

# define AI\_V4MAPPED 0x0008

/\* Return IPv4 mapped and IPv6 addresses. \*/

如果同时指定了ai\_family=AF\_INET6，那么除了返回AAAA记录对应的IPv6地址，还返回IPv4映射的IPv6地址。

# define AI\_ALL 0x0010

/\* Use configuration of this host to choose returned address type.. \*/

根据所在主机的配置选择返回地址类型。

# define AI\_ADDRCONFIG 0x0020

# ifdef \_\_USE\_GNU

/\* IDN encode input (assuming it is encoded in the current locale's character set)

before looking it up. \*/

# define AI\_IDN 0x0040

/\* Translate canonical name from IDN format. \*/

# define AI\_CANONIDN 0x0080

/\* Don't reject unassigned Unicode code points. \*/

# define AI\_IDN\_ALLOW\_UNASSIGNED 0x0100

/\* Validate strings according to STD3 rules. \*/

# define AI\_IDN\_USE\_STD3\_ASCII\_RULES 0x0200

# endif

/\* Don't use name resolution. \*/

service参数必须是一个十进制端口号数串

# define AI\_NUMERICSERV 0x0400

# 基本方法

## demo

### 原型

### 功能

### 参数

#### xx

### 返回值

### 注意

## fcntl

### 原型

#include <fcntl.h>

int fcntl(int fd, int cmd, … /\* arg \*/);

### 功能

设置非阻塞IO

### 参数

#### fd

#### cmd

* F\_SETFL

FL表示flags。设置标志位。

* F\_GETFL

获取标志位。

#### arg

* O\_NONBLOCK

非阻塞式I/O。

* F\_GETFL

获取标志位。

### 返回值

* 若成功取决于cmd
* 若出错返回-1

### 注意

**设置某个文件状态标志位的唯一正确的方法是**：**先取得当前标志，与新标志逻辑或后再设置标志**。

int **flags**;

if (**flags** = fcntl(fd, F\_GETFL, 0) < 0)

出错

**flags** |= O\_NONBLOCK;

if (fcntl(fd, F\_SETFL,**flags**) < 0)

出错

## freeaddrinfo

### 原型

/usr/include/netdb.h

void freeaddrinfo(struct addrinfo \*ai);

### 功能

由getaddrinfo返回的所有存储空间都是动态获取的，需要freeaddrinfo来释放。

### 相关方法

getaddrinfo

gai\_strerror

### 参数

#### ai

### 返回值

### 注意

## gai\_strerror

### 原型

/usr/include/netdb.h

const char\* gai\_strerror(int error);

### 功能

返回错误信息

### 相关方法

getaddrinfo

freeaddrinfo

### 参数

#### error

### 返回值

### 注意

## getaddrinfo

### 原型

#include <netdb.h>

int getaddrinfo( const char\* hostname,

const char\* service,

const struct addrinfo \*hints,

struct addrinfo \*\*result);

### 功能

两种转换

* 名字🡨🡪地址
* 服务🡨🡪端口

返回的sockaddr结构可直接在套接字函数中使用。

### 相关方法

gai\_strerror

freeaddrinfo

### 参数

#### hostname

主机名或地址串（IPv4的点分十进制数串或IPv6的十六进制串）

#### service

服务名或十进制端口号数串（实例）

#### hints

hints：英文释义为“提示”

调用者在这个结果中填入期望返回的暗示。

* 指向addrinfo的指针
  + [ai\_flags](#_ai_flags)

零个或多个在一起的AI\_xxx值。

* + ai\_family

某个AF\_xxx值

* + ai\_socketype

某个SOCK\_xxx值

* + ai\_protocol
* 空指针
  + ai\_flags = 0
  + ai\_socketype = 0
  + ai\_protocol = 0
  + ai\_family = AF\_UNSPEC

#### result

如果本函数返回成功0，那么result参数指向的变量已被填入一个指针，它指向的是有ai\_next成员串接起来的[addrinfo](#_addrinfo)结构链表。

可导致返回多个addrinfo结构的情形。

1. hostname参数关联多个地址
2. service参数指定的服务支持多个套接字类型

### 返回值

成功：返回0

失败：返回非0

### 注意

## htons/htonl

### 原型

#include <netinet/in.h>

uint\_16 htons(uint16\_t host16bitvalue);

uint\_32 htonl(uint32\_t host32bitvalue);

### 功能

主机字节序🡪网络字节序

例如：linux下就是小端🡪大端

### 参数

#### host16bitvalue/ host32bitvalue

uint16\_t或uint32\_t

### 返回值

返回网络字节序

### 注意

## inetpton

### 原型

#include <arpa/inet.h>

int inetpton( int family,

const char \*strptr,

void \*addrptr);

### 功能

p: presentation

n: numeric

将“192.168.1.1”转换成0xC0A80101。

支持IPv4和IPv6。

### 参数

#### [family](#_ai_family)

#### strptr

IPv4：

点分十进制: “192.168.4.12”

IPv6：

“2001:0DB8:0000:0000:0000:0000:1428:0000”

#### addrptr

in\_addr或in6\_addr

### 返回值

成功返回1

输入不是有效的表达格式则为0

失败返回-1

### 注意

## ioctl—[未完成]

### 原型

#include <sys/ioctl.h>

int ioctl(int fd, int request, … /\* void \*arg \*/);

### 功能

获取接口信息、访问路由表、访问ARP高速缓存

### 参数

#### fd

#### request

好多

#### arg

与request对应

### 返回值

### 注意

## ntohs/ntohl

### 原型

#include <netinet/in.h>

uint16\_t ntohs(uint16\_t net16bitvalue);

uint32\_t ntohl(uint32\_t net32bitvalue);

### 功能

网络字节序🡪主机字节序

例如，linux下就是大端转小端。

### 参数

#### net16bitvalue/ net32bitvalue

uint16\_t

uint32\_t

### 返回值

主机字节序

### 注意

## select

### 原型

#include <sys/select.h>

#include <sys/time.h>

int select(int maxfdp1,

fd\_set \*readset,

fd\_set \*writeset,

fd\_set \*exceptset,

const struct timeval \*timeout);

### 参数

#### maxfdp1

#### readset，writeset，exceptset

这三个参数类型为：[fd\_set](2-fd_set/fd_set.docx)

##### 注意

该函数返回时，结果将指示那些描述符已就绪，可以使用FD\_ISSET宏来测试fd\_set数据类型中的描述符，描述符中集内任何与未就绪描述符对应的位返回时均清成0，所以**每次重新调用select函数时，都得再次把所有描述符集内所关心的位均置为1**。

#### timeout

* NULL

永远等待，直到有描述符准备好I/O时，才返回。

* 0

立即返回，检查完描述符后立即返回。

* 非0，非NULL

等待一段时间，如果发现有描述符准备好I/O时，就立即返回；否则一直等到超时时间后才返回。

### 返回值

* N>0

返回所有描述符集已就绪的总位数，超时之前。

* 0

没有描述符准备好，且时间已到

* -1

出错，例如select方法被信号中断

## socket

### 原型

#include <sys/socket.h>

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

### 参数

#### family

#### type

#### protocol

##### 头文件

/usr/include/netinet/in.h

/\* Standard well-defined IP protocols. \*/

enum

{

IPPROTO\_IP = 0, /\* Dummy protocol for TCP. \*/

#define IPPROTO\_IP IPPROTO\_IP

IPPROTO\_ICMP = 1, /\* Internet Control Message Protocol. \*/

#define IPPROTO\_ICMP IPPROTO\_ICMP

IPPROTO\_IGMP = 2, /\* Internet Group Management Protocol. \*/

#define IPPROTO\_IGMP IPPROTO\_IGMP

IPPROTO\_IPIP = 4, /\* IPIP tunnels (older KA9Q tunnels use 94). \*/

#define IPPROTO\_IPIP IPPROTO\_IPIP

IPPROTO\_TCP = 6, /\* Transmission Control Protocol. \*/

#define IPPROTO\_TCP IPPROTO\_TCP

IPPROTO\_UDP = 17, /\* User Datagram Protocol. \*/

#define IPPROTO\_UDP IPPROTO\_UDP

};