NAME

inet – Internet protocol family

SYNOPSIS

#include <sys/types.h>
#include <netinet/in.h>

DESCRIPTION

The Internet protocol family implements a collection of protocols which are centered around the *Internet Protocol* (IP) and which share a common address format. The Internet family protocols can be accessed using the socket interface, where they support the **SOCK_STREAM**, **SOCK_DGRAM**, and **SOCK_RAW** socket types, or the Transport Level Interface (TLI), where they support the connectionless (**T_CLTS**) and connection oriented (**T_COTS_ORD**) service types.

PROTOCOLS

The Internet protocol family comprises the Internet Protocol (IP), the Address Resolution Protocol (ARP), the Internet Control Message Protocol (ICMP), the Transmission Control Protocol (TCP), and the User Datagram Protocol (UDP).

TCP supports the socket interface's **SOCK_STREAM** abstraction and TLI's **T_COTS_ORD** service type. UDP supports the **SOCK_DGRAM** socket abstraction and the TLI **T_CLTS** service type. See **tcp**(7P) and **udp**(7P). A direct interface to IP is available using both TLI and the socket interface; See **ip**(7P). ICMP is used by the kernel to handle and report errors in protocol processing. It is also accessible to user programs; see **icmp**(7P). ARP is used to translate 32-bit IP addresses into 48-bit Ethernet addresses; see **arp**(7P).

The 32-bit IP address is divided into network number and host number parts. It is frequency-encoded; The most-significant bit is zero in Class A addresses, in which the high-order 8 bits represent the network number. Class B addresses have their high order two bits set to 10 and use the high-order 16 bits as the network number field. Class C addresses have a 24-bit network number part of which the high order three bits are 110. Sites with a cluster of IP networks may chose to use a single network number for the cluster; This is done by using subnet addressing. The host number portion of the address is further subdivided into subnet number and host number parts. Within a subnet, each subnet appears to be an individual network; Externally, the entire cluster appears to be a single, uniform network requiring only a single routing entry. Subnet addressing is enabled and examined by the following **ioctl**(2) commands; They have the same form as the **SIOCSIFADDR** command

SIOCSIFNETMASK Set interface network mask. The network mask defines the network part of the address; If it contains more of the address than the address type would indicate, then subnets are in use.

SIOCGIFNETMASK Get interface network mask.

ADDRESSING

IP addresses are four byte quantities, stored in network byte order. IP addresses should be manipulated using the byte order conversion routines (see **byteorder**(3N)).

Addresses in the Internet protocol family use the **sockaddr_in** structure, which has that following members:

```
short      sin_family;
u_short    sin_port;
struct     in_addr     sin_addr;
char       sin_zero[8];
```

Library routines are provided to manipulate structures of this form; See **inet**(3N).

The **sin_addr** field of the **sockaddr_in** structure specifies a local or remote IP address. Each network interface has its own unique IP address. The special value **INADDR_ANY** may be used in this field to effect "wildcard" matching. Given in a **bind**(3N) call, this value leaves the local IP address of the socket unspecified, so that the socket will receive connections or messages directed at any of the valid IP addresses of the system. This can prove useful when a process neither knows nor cares what the local IP address is or when a process wishes to receive requests using all of its network interfaces. The **sockaddr_in** structure given in the **bind**(3N) call must specify an **in_addr** value of either **IPADDR_ANY** or one of the system's valid IP addresses. Requests to bind any other address will elicit the error **EADDRNOTAVAI**. When a **connect**(3N) call is made for a socket that has a wildcard local address, the system sets the **sin_addr** field of the socket to the IP address of the network interface that the packets for that connection are routed via.

The **sin_port** field of the **sockaddr_in** structure specifies a port number used by TCP or UDP. The local port address specified in a **bind**(3N) call is restricted to be greater than **IPPORT_RESERVED** (defined in **<netinet/in.h>**) unless the creating process is running as the super-user, providing a space of protected port numbers. In addition, the local port address must not be in use by any socket of same address family and type. Requests to bind sockets to port numbers being used by other sockets return the error **EADDRINUSE**. If the local port address is specified as 0, then the system picks a unique port address greater than **IPPORT_RESERVED**. A unique local port address is also picked when a socket which is not bound is used in a **connect**(3N) or **sendto** (see **send**(3N)) call. This allows programs which do not care which local port number is used to set up TCP connections by simply calling **socket**(3N) and then **connect**(3N), and to send UDP datagrams with a **socket**(3N) call followed by a **sendto( )** call.

Although this implementation restricts sockets to unique local port numbers, TCP allows multiple simultaneous connections involving the same local port number so long as the remote IP addresses or port numbers are different for each connection. Programs may explicitly override the socket restriction by setting the **SO_REUSEADDR** socket option with **setsockopt** (see **getsockopt**(3N)).

TLI applies somewhat different semantics to the binding of local port numbers. These semantics apply when Internet family protocols are used using the TLI.

SEE ALSO

ioctl(2), **bind**(3N), **byteorder**(3N), **connect**(3N), **gethostbyname**(3N), **getnetbyname**(3N), **getprotobyname**(3N), **getservbyname**(3N), **getsockopt**(3N), **send**(3N), **socket**(3N), **arp**(7P), **icmp**(7P), **ip**(7P), **tcp**(7P), **udp**(7P)

Network Information Center, *DDN Protocol Handbook* (3 vols.), Network Information Center, SRI International, Menlo Park, Calif., 1985.

NOTES

The Internet protocol support is subject to change as the Internet protocols develop. Users should not depend on details of the current implementation, but rather the services exported.