IPv4 Socket Address Structure

The Internet socket address structure is named `sockaddr_in` and is defined by including `<netinet/in.h>` header.

```c
struct in_addr {
    in_addr_t s_addr;  /* 32-bit IP address */
} /* network byte ordered */

struct sockaddr_in {
    uint8_t sin_len;  /* length of structure (16) */
    sa_family_t sin_family; /* AF_INET */
    in_port_t sin_port;   /* 16-bit TCP or UDP port number */
                           /* network byte ordered */
    struct in_addr sin_addr; /* 32-bit IPv4 address */
                           /* network byte ordered */
    char sin_zero[8];    /* unused */
};
```
Figure 2.16

Leon-Garcia & Widjaja: Communication Networks
TCP socket calls

Server
- `socket()`
- `bind()`
- `listen()`
- `accept()` (blocks until server receives a connect request from client)

Client
- `socket()`
- `connect()` (connect negotiation)
- `read()`
- `write()`
- `read()`
- `close()`

Networks: TCP/IP Socket Calls

Copyright ©2000 The McGraw Hill Companies  Leon-Garcia & Widjaja: Communication Networks  Figure 2.17
UDP socket calls

Server

socket()

bind()

recvfrom()

blocks until server receives data from client

sendto()

close()

Client

socket()

bind()

Not needed

sendto()

recvfrom()

close()
System Calls for Elementary TCP Sockets

#include <sys/types.h>
#include <sys/socket.h>

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int socket(...)</td>
<td>int family, int type, int protocol</td>
</tr>
</tbody>
</table>

**family**: specifies the protocol family   {AF_INET for TCP/IP}

**type**: indicates communications semantics

- SOCK_STREAM   stream socket   TCP
- SOCK_DGRAM     datagram socket  UDP
- SOCK_RAW       raw socket

**protocol**: set to 0 except for raw sockets

**returns** on success:    socket descriptor  {a small nonnegative integer}

on error:    -1

Example:

```c
if ((sd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    err_sys(“socket call error”);
```
connect Function

int connect (int sockfd, const struct sockaddr *servaddr, socklen_t addrlen);

sockfd: a socket descriptor returned by the socket function
*servaddr: a pointer to a socket address structure
addrlen: the size of the socket address structure

The socket address structure must contain the **IP address** and the **port number** for the connection wanted.

In TCP `connect` initiates a three-way handshake. `connect` returns only when the connection is established or when an error occurs.

returns on success: 0
on error: -1

Example:

```c
if (connect (sd, (struct sockaddr *) &servaddr, sizeof (servaddr)) != 0)
  err_sys("connect call error");
```
TCP socket calls

Server
- `socket()`
- `bind()`
- `listen()`
- `accept()`

blocks until server receives a connect request from client

Client
- `socket()`
- `connect()`

connect negotiation

- `read()`
- `write()`
- `close()`

- `read()`
- `write()`
- `close()`
bind Function

int bind (int sockfd, const struct sockaddr *myaddr, socklen_t addrlen);

bind assigns a local protocol address to a socket.

protocol address: a 32 bit IPv4 address and a 16 bit TCP or UDP port number.

sockfd: a socket descriptor returned by the socket function.

*myaddr: a pointer to a protocol-specific address.

addrlen: the size of the socket address structure.

Servers bind their “well-known port” when they start.

returns on success: 0

on error: -1

Example:

if (bind (sd, (struct sockaddr *) &servaddr, sizeof (servaddr)) != 0)
errsyst (“bind call error”);
listen Function

int listen (int sockfd, int backlog);

listen is called only by a TCP server and performs two actions:
1. Converts an unconnected socket (sockfd) into a passive socket.
2. Specifies the maximum number of connections (backlog) that the kernel should queue for this socket.

listen is normally called before the accept function.

returns on success: 0
on error: -1

Example:

if (listen (sd, 2) != 0)
    errsyst ("listen call error");
accept Function

int accept (int sockfd, struct sockaddr *cliaddr, socklen_t *addrlen);

accept is called by the TCP server to return the next completed connection from the front of the completed connection queue.

sockfd: This is the same socket descriptor as in listen call.

*cliaddr: used to return the protocol address of the connected peer process (i.e., the client process).

*addrlen: {this is a value-result argument}

before the accept call: We set the integer value pointed to by *addrlen to the size of the socket address structure pointed to by *cliaddr;

on return from the accept call: This integer value contains the actual number of bytes stored in the socket address structure.

returns on success: a new socket descriptor

on error: -1
int accept (int sockfd, struct sockaddr *cliaddr, socklen_t addrlen);

For `accept` the first argument `sockfd` is the listening socket and the returned value is the connected socket.

The server will have one connected socket for each client connection accepted.

When the server is finished with a client, the connected socket must be closed.

Example:

```c
sfd = accept (sd, NULL, NULL);
if (sfd == -1) err_sys ("accept error");
```
**close Function**

```c
int close (int sockfd);
```

*close* marks the socket as closed and returns to the process immediately.

*sockfd:* This socket descriptor is no longer useable.

Note – TCP will try to send any data already queued to the other end before the normal connection termination sequence.

Returns on success: 0

on error: -1

Example:

```
close (sd);
```