The Socket Function

To perform network I/O, the first thing a process must do is, call the socket function, specifying the type of communication protocol desired and protocol family, etc.

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

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

This call returns a socket descriptor that you can use in later system calls or -1 on error.

Parameters

family − It specifies the protocol family and is one of the constants shown below −

Family	Description
AF_INET	IPv4 protocols
AF_INET6	IPv6 protocols
AF_LOCAL	Unix domain protocols
AF_ROUTE	Routing Sockets
AF_KEY	Ket socket

type − It specifies the kind of socket you want. It can take one of the following values −

Type	Description
SOCK_STREAM	Stream socket
SOCK_DGRAM	Datagram socket
SOCK_SEQPACKET	Sequenced packet socket
SOCK_RAW	Raw socket

protocol − The argument should be set to the specific protocol type given below, or 0 to select the system's default for the given combination of family and type −

Protocol	Description
IPPROTO_TCP	TCP transport protocol
IPPROTO_UDP	UDP transport protocol
IPPROTO_SCTP	SCTP transport protocol

By: Pragya Singh Tomar
ICS, Vikram University, Ujjain

The *connect* Function

The *connect* function is used by a TCP client to establish a connection with a TCP server.

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

int connect(int sockfd, struct sockaddr *serv_addr, int addrlen);
```

This call returns 0 if it successfully connects to the server, otherwise it returns -1 on error.

Parameters

- **sockfd** − It is a socket descriptor returned by the socket function.
- **serv_addr** − It is a pointer to struct sockaddr that contains destination IP address and port.
- **addrlen** − Set it to sizeof(struct sockaddr).

The *bind* Function

The *bind* function assigns a local protocol address to a socket. With the Internet protocols, the protocol address is the combination of either a 32-bit IPv4 address or a 128-bit IPv6 address, along with a 16-bit TCP or UDP port number. This function is called by TCP server only.

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

int bind(int sockfd, struct sockaddr *my_addr,int addrlen);
```

This call returns 0 if it successfully binds to the address, otherwise it returns -1 on error.

Parameters

- **sockfd** − It is a socket descriptor returned by the socket function.
- **my_addr** − It is a pointer to struct sockaddr that contains the local IP address and port.
- **addrlen** − Set it to sizeof(struct sockaddr).

The *listen* Function

The *listen* function is called only by a TCP server and it performs two actions −

- The listen function converts an unconnected socket into a passive socket, indicating that the kernel should accept incoming connection requests directed to this socket.
- The second argument to this function specifies the maximum number of connections the kernel should queue for this socket.

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


By: Pragya Singh Tomar
ICS, Vikram University, Ujjain

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

int listen(int sockfd,int backlog);
```

This call returns 0 on success, otherwise it returns -1 on error.

Parameters

- **sockfd** − It is a socket descriptor returned by the socket function.
- **backlog** − It is the number of allowed connections.

The *accept* Function

The *accept* function is called by a TCP server to return the next completed connection from the front of the completed connection queue. The signature of the call is as follows −

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

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

This call returns a non-negative descriptor on success, otherwise it returns -1 on error. The returned descriptor is assumed to be a client socket descriptor and all read-write operations will be done on this descriptor to communicate with the client.

Parameters

- **sockfd** − It is a socket descriptor returned by the socket function.
- **cliaddr** − It is a pointer to struct sockaddr that contains client IP address and port.
- **addrlen** − Set it to sizeof(struct sockaddr).

The *send* Function

The *send* function is used to send data over stream sockets or CONNECTED datagram sockets. If you want to send data over UNCONNECTED datagram sockets, you must use sendto() function.

You can use *write()* system call to send data. Its signature is as follows −

```
int send(int sockfd, const void *msg, int len, int flags);
```

This call returns the number of bytes sent out, otherwise it will return -1 on error.

Parameters

- **sockfd** − It is a socket descriptor returned by the socket function.
- **msg** − It is a pointer to the data you want to send.

By: Pragya Singh Tomar
ICS, Vikram University, Ujjain

- **len** − It is the length of the data you want to send (in bytes).
- **flags** − It is set to 0.

The *recv* Function

The *recv* function is used to receive data over stream sockets or CONNECTED datagram sockets. If you want to receive data over UNCONNECTED datagram sockets you must use recvfrom().

You can use *read()* system call to read the data. This call is explained in helper functions chapter.

```
int recv(int sockfd, void *buf, int len, unsigned int flags);
```

This call returns the number of bytes read into the buffer, otherwise it will return -1 on error.

Parameters

- **sockfd** − It is a socket descriptor returned by the socket function.
- **buf** − It is the buffer to read the information into.
- **len** − It is the maximum length of the buffer.
- **flags** − It is set to 0.

The *sendto* Function

The *sendto* function is used to send data over UNCONNECTED datagram sockets. Its signature is as follows −

```
int sendto(int sockfd, const void *msg, int len, unsigned int flags, const
struct sockaddr *to, int tolen);
```

This call returns the number of bytes sent, otherwise it returns -1 on error.

Parameters

- **sockfd** − It is a socket descriptor returned by the socket function.
- **msg** − It is a pointer to the data you want to send.
- **len** − It is the length of the data you want to send (in bytes).
- **flags** − It is set to 0.
- **to** − It is a pointer to struct sockaddr for the host where data has to be sent.
- **tolen** − It is set it to sizeof(struct sockaddr).

The *recvfrom* Function

The *recvfrom* function is used to receive data from UNCONNECTED datagram sockets.

```
int recvfrom(int sockfd, void *buf, int len, unsigned int flags struct
sockaddr *from, int *fromlen);
```

This call returns the number of bytes read into the buffer, otherwise it returns -1 on error.

Parameters

- **sockfd** − It is a socket descriptor returned by the socket function.
- **buf** − It is the buffer to read the information into.
- **len** − It is the maximum length of the buffer.
- **flags** − It is set to 0.
- **from** − It is a pointer to struct sockaddr for the host where data has to be read.
- **fromlen** − It is set it to sizeof(struct sockaddr).

The *close* Function

The *close* function is used to close the communication between the client and the server. Its syntax is as follows −

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

This call returns 0 on success, otherwise it returns -1 on error.

Parameters

- **sockfd** − It is a socket descriptor returned by the socket function.

The *shutdown* Function

The *shutdown* function is used to gracefully close the communication between the client and the server. This function gives more control in comparison to the *close* function. Given below is the syntax of *shutdown* −

```
int shutdown(int sockfd, int how);
```

This call returns 0 on success, otherwise it returns -1 on error.

Parameters

- **sockfd** − It is a socket descriptor returned by the socket function.
- **how** − Put one of the numbers −
    - **0** − indicates that receiving is not allowed,
    - **1** − indicates that sending is not allowed, and
    - **2** − indicates that both sending and receiving are not allowed. When *how* is set to 2, it's the same thing as close().