# Socket Programming in C

Usman Wajid
usman.wajid88@gmail.com

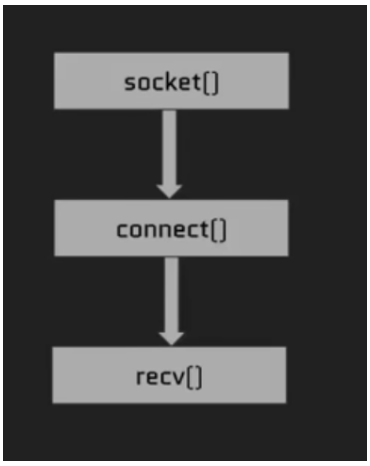
FAST, Peshawar

## 1 Socket

Sockets are the low-level endpoints used for processing information across a network. Some common protocols like HTTP, FTP rely on sockets to make connections. Socket Programming is the route of connecting two points on a network to communicate with each other.

## 2 Client Socket Workflow

The client socket is created with a socket() call, and then connected to a remote address with the connect() call, and then finally can retrieve data with the recv() call

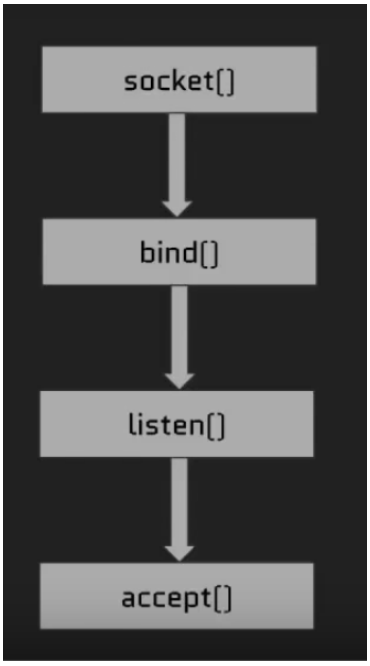


## 2.1 Client socket code example

```c
#include <stdio.h>
#include <stdlib.h>

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

#include <netinet/in.h>

void main(){

        printf("Hello world\n");


        /* creating a socket */
        int network_socket;
        network_socket = socket(AF_INET, SOCK_STREAM, 0 );
        /*
        int socket(int domain, int type, int protocol);
        network_socket = socket(AF_INET, SOCK_STREAM, 0 );

        1st argument:
                AF_INET -------> IPv4
                AF_INET6 ------> IPv6

        2nd argument:
                SOCK_STREAM ---> TCP, acknowledgment, full duplex
                SOCK_DGRAM ----> UDP, no-acknowledgment, connectionless,

        3rd argument:
                0 ---> in case when there exist a single protocol. For example TCP has a single protocol, UDP has a single protocol
        */

        // specify an address for the socket
        struct sockaddr_in server_address;
        server_address.sin_family = AF_INET;
        server_address.sin_port = htons(9002);
        server_address.sin_addr.s_addr = INADDR_ANY;

        //making a connection
        int connection_status = connect(network_socket, (struct sockaddr *) &server_address, sizeof(server_address));
        if (connection_status == -1)
                printf("Connection error!!!\n");

        // to receive data from the server
        char server_response[256];
        recv(network_socket, &server_response, sizeof(server_response), 0);

        // printing the msg/data
        printf("%s\n", server_response);

        // closing the socket
        //close(network_socket);

}
```

# 3 Server Socket workflow

On the "Server" end of the socket, we need to also create a socket with a socket() call, but then, we need to bind() that socket to an IP and port where it can then listen() for connections, and then finally accept() a connection and then send() or recv() data to the other sockets it has connected to



## 3.1 Server socket code example

```c
#include <stdio.h>
#include <stdlib.h>

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

#include <netinet/in.h>

void main(){

        //printf("Hello world\n");

        char server_message[256] = "You have reached the server";

        /* creating a socket */
        int network_socket;
        network_socket = socket(AF_INET, SOCK_STREAM, 0 );
        /*
        int socket(int domain, int type, int protocol);
        network_socket = socket(AF_INET, SOCK_STREAM, 0 );

        1st argument:
                AF_INET -------> IPv4
                AF_INET6 ------> IPv6

        2nd argument:
                SOCK_STREAM ---> TCP, acknowledgment, full duplex
                SOCK_DGRAM ----> UDP, no-acknowledgment, connectionless,

        3rd argument:
                0 ---> in case when there exist a single protocol. For example TCP has a single protocol, UDP has a single protocol
        */

        // define the server address
        struct sockaddr_in server_address;
        server_address.sin_family = AF_INET;
        server_address.sin_port = htons(9002);
        server_address.sin_addr.s_addr = INADDR_ANY; // destination IP address ; INADDR_ANY ---> ip address of the same pc ; or inet_addr
  (DEST_IP) ; or inet_addr("192.168.1.100")

        //bind the socket
        int bind_status = bind(network_socket, (struct sockaddr *) &server_address, sizeof(server_address));
        if (bind_status == -1)
                printf("Connection error!!!\n");

        listen(network_socket, 5);

        int client_socket;
        client_socket = accept(network_socket, NULL, NULL);

        // send the message
        send(client_socket, server_message, sizeof(server_message), 0);

        // closing the socket
        close(network_socket);

}
```