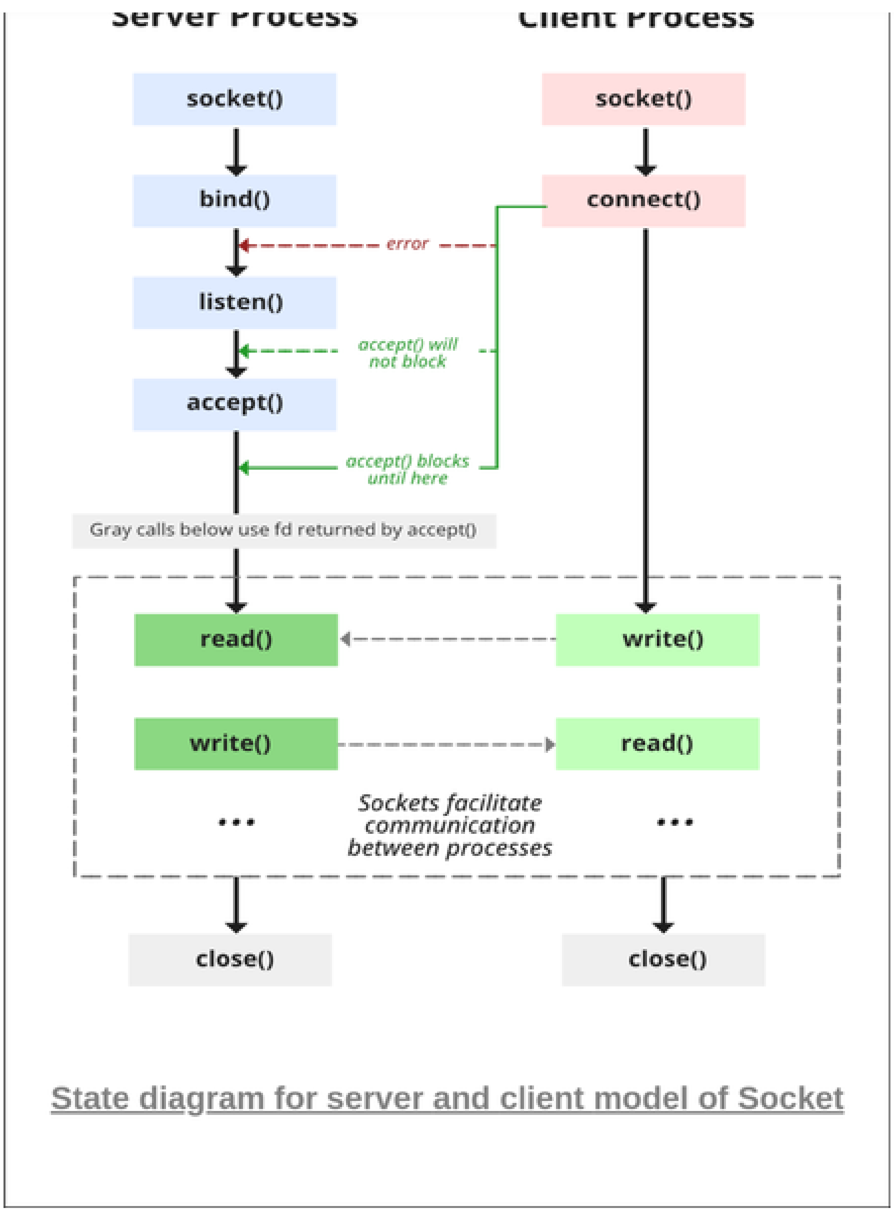
Socket Programming in C/C++

Difficulty Level : Medium • Last Updated : 08 Jul, 2022

What is socket programming?

Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while the other socket reaches out to the other to form a connection. The server forms the listener socket while the client reaches out to the server.

State diagram for server and client model


Server Process
Client Process
socket()
bind()
listen()
accept()
socket()
connect()
error
accept() will not block
accept() blocks until here
Gray calls below use fd returned by accept()
read()
write()
write()
read()
...
Sockets facilitate communication between processes
...
close()
close()
State diagram for server and client model of Socket

State diagram for server and client model of Socket

Stages for server

1. Socket creation:

```
int sockfd = socket(domain, type, protocol)
```

- **sockfd:** socket descriptor, an integer (like a file-handle)
  domain: integer, specifies communication domain. We use AF_ LOCAL as defined in the POSIX standard for communication between processes on the same host. For

- **type:** communication type
  SOCK_STREAM: TCP(reliable, connection oriented)
  SOCK_DGRAM: UDP(unreliable, connectionless)
- **protocol:** Protocol value for Internet Protocol(IP), which is 0. This is the same number which appears on protocol field in the IP header of a packet.(man protocols for more details)

2. Setsockopt: This helps in manipulating options for the socket referred by the file descriptor sockfd. This is completely optional, but it helps in reuse of address and port. Prevents error such as: "address already in use".

> *int setsockopt(int sockfd, int level, int optname, const void *optval, socklen_t optlen);*

3. Bind:

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

After creation of the socket, bind function binds the socket to the address and port number specified in addr(custom data structure). In the example code, we bind the server to the localhost, hence we use INADDR_ANY to specify the IP address.

4. Listen:

> *int listen(int sockfd, int backlog);*

It puts the server socket in a passive mode, where it waits for the client to approach the server to make a connection. The backlog, defines the maximum length to which the queue of pending connections for sockfd may grow. If a connection request arrives ıen the queue is full, the client may receive an error with an indication of ECONNREFUSED.

*int new_socket= accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen);*

It extracts the first connection request on the queue of pending connections for the listening socket, sockfd, creates a new connected socket, and returns a new file descriptor referring to that socket. At this point, connection is established between client and server, and they are ready to transfer data.

Stages for Client

- **Socket connection:** Exactly same as that of server's socket creation
- **Connect:** The connect() system call connects the socket referred to by the file descriptor sockfd to the address specified by addr. Server's address and port is specified in addr.

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

Implementation

Here we are exchanging one hello message between server and client to demonstrate the client/server model.

Server.c

```
// Server side C/C++ program to demonstrate Socket
// programming
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <unistd.h>
#define PORT 8080
int main(int argc, char const* argv[])
{
    int server_fd, new_socket, valread;
    struct sockaddr_in address;
    int opt = 1;
    int addrlen = sizeof(address);
    char buffer[1024] = { 0 };
    char* hello = "Hello from server";

    // Creating socket file descriptor
    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0))
        == 0) {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }

    // Forcefully attaching socket to the port 8080
    if (setsockopt(server_fd, SOL_SOCKET,
                   SO_REUSEADDR | SO_REUSEPORT, &opt,
                   sizeof(opt))) {
        perror("setsockopt");
        exit(EXIT_FAILURE);
    }
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(PORT);

    // Forcefully attaching socket to the port 8080
    if (bind(server_fd, (struct sockaddr*)&address,
             sizeof(address))
        < 0) {
        perror("bind failed");
        exit(EXIT_FAILURE);
    }
    if (listen(server_fd, 3) < 0) {
        perror("listen");
        exit(EXIT_FAILURE);
    }
    if ((new_socket
         = accept(server_fd, (struct sockaddr*)&address,
                  (socklen_t*)&addr    )
    }
    valread = read(new_socket, buffer, 1024);
    printf("%s\n", buffer);
    send(new_socket, hello, strlen(hello), 0);
    printf("Hello message sent\n");

  // closing the connected socket
    close(new_socket);
  // closing the listening socket
    shutdown(server_fd, SHUT_RDWR);
    return 0;
}
```

- **client.c**

C

```
// Client side C/C++ program to demonstrate Socket
// programming
#include <arpa/inet.h>
#include <stdio.h>
#include <string.h>
#include <sys/socket.h>
#include <unistd.h>
#define PORT 8080

int main(int argc, char const* argv[])
{
    int sock = 0, valread, client_fd;
    struct sockaddr_in serv_addr;
    char* hello = "Hello from client";
    char buffer[1024] = { 0 };
    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        printf("\n Socket creation error \n");
        return -1;
    }

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT);

    // Convert IPv4 and IPv6 addresses from text to binary
    // form
    if (inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr)
        <= 0) {
        printf(
            "\nInvalid address/ Add      not supported \n");
        return -1;
```

```
           = connect(sock, (struct sockaddr*)&serv_addr,
                      sizeof(serv_addr)))
          < 0) {
          printf("\nConnection Failed \n");
          return -1;
      }
      send(sock, hello, strlen(hello), 0);
      printf("Hello message sent\n");
      valread = read(sock, buffer, 1024);
      printf("%s\n", buffer);

      // closing the connected socket
      close(client_fd);
      return 0;
  }
```

Compiling:

```
gcc client.c -o client
gcc server.c -o server
```

Output:

```
Client:Hello message sent
Hello from server
Server:Hello from client
Hello message sent
```

Next: Socket Programming in C/C++: Handling multiple clients on server without multi threading

This article is contributed by **Akshat Sinha**. If you like GeeksforGeeks and would like to contribute, you can also write an article using write.geeksforgeeks.org or mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more formation about the topic discussed above.

RECOMMENDED ARTICLES Page : 1 2 3

01 Socket Programming in C/C++: Handling multiple clients on server without multi threading
29, Aug 16

02 getchar_unlocked() – Faster Input in C/C++ For Competitive Programming
18, Apr 16

05 C++: Methods of code shortening in competitive programming
04, Sep 18

06 Comparison of Java with other programming languages
16, Jan 19

Beginners
13, Jun 17

04 C++ Programming and STL Facts
20, Mar 18

08 Hello World Program : First program while learning Programming
25, Nov 19

Article Contributed By :

GeeksforGeeks

Vote for difficulty

Current difficulty : Medium

Easy Normal Medium Hard Expert

Improved By : lcmgcd, MichaelThomasKloos, introdynelco1, harendrakumar123, nehalnimkar

Article Tags : CPP-Library, C++

Practice Tags : CPP

Improve Article Report Issue

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

Start Your Coding Journey Now! Login Register

ReactJS Video Internship

NodeJS