Pragma in C/C++

Modulo Operator (%) in C/C++ with Examples

Setting up a C++ Competitive Programming Environment

How to find the Entry with largest Value in a C++ Map

Learn C++ Programming Step by Step - A 20 Day Curriculum!

Writing code faster during Competitive Programming in C++

Role of SemiColon in various Programming Languages

Priority queue of pairs in C++ with ordering by first and second element

Default values

in linear time using C++ STL

Implementing upper_bound() and lower_bound() for Ordered Set in C++

C/C++ program for calling main() in main()

Important functions of STL Components in C++

Default value of Vector in C++ STL

How to find index of a given element in a Vector in C++

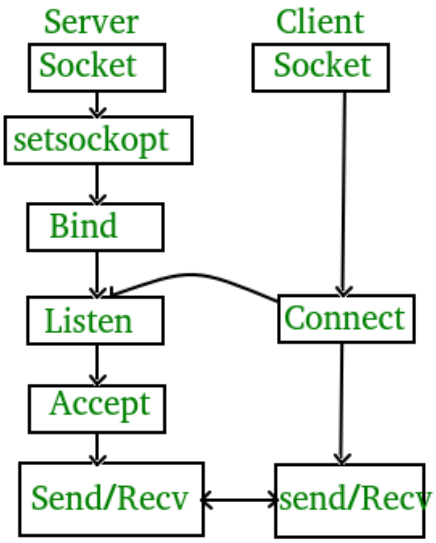
Socket Programming in C/C++

06-12-2016

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 other socket reaches out to the other to form a connection. Server forms the listener socket while client reaches out to the server.

State diagram for server and client model



Stages for server

- **Socket creation:**

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

sockfd: socket descriptor, an integer (like a file-handle)
domain: integer, communication domain e.g., AF_INET (IPv4 protocol) , AF_INET6 (IPv6 protocol)
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)

- **Setsockopt:**

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

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”.

- **Bind:**

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

ber 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.

- **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 when the queue is full, the client may receive an error with an indication of ECONNREFUSED.

- **Accept:**

```
int new_socket= accept(int sockfd, struct sockaddr *addr, socklen_t *
```

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:**

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

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.

Implementation

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

server.c

```c
// Server side C/C++ program to demonstrate Socket programming
#include <unistd.h>
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <string.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);
    }
```

```
    // 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*)&addrlen))<0)
    {
        perror("accept");
        exit(EXIT_FAILURE);
    }
    valread = read( new_socket , buffer, 1024);
    printf("%s\n",buffer );
    send(new_socket , hello , strlen(hello) , 0 );
    printf("Hello message sent\n");
    return 0;
}
```

client.c

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

```
    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/ Address not supported \n");
        return -1;
    }

    if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)
    {
        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 );
    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 contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main pa and help other Geeks.

Recommended Posts:

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

What is web socket and how it is different from the HTTP?

Web Programming in C++

P : A Programming Language

C++ programming and STL facts

Introduction to SAS programming

Best 5 Programming Languages For a Getting a Job

Why learning C Programming is a must?

C++ Programming Basics

How to Learn Programming?

I Can't Use Logic In Programming. What Should I Do?

Introduction of Programming Paradigms

Introduction to Programming Languages

Blog | Programming Guidelines

5 Programming Tips For Beginners

5 Best Programming Languages For Newbies

What is Competitive Programming and How to Prepare for It?

What Are The Best Resources For Competitive Programming?

Introduction to C++ Programming Language

Improved By : lcmgcd, MichaelThomasKloos

Article Tags : C++ GBlog CPP-Library

Practice Tags : CPP

26

3

To-do Done

Based on **20** vote(s)

Feedback/ Suggest Improvement Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

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

40 Comments GeeksforGeeks Disqus' Privacy Policy 1 Login

Recommend 13 Tweet Share Sort by Newest

Join the discussion…

LOG IN WITH OR SIGN UP WITH DISQUS ?

Name

benjamin c • 2 months ago

guys, a newbie here, in line,

```
char *hello = "Hello from server";
```

why does it use char pointer to hold message instead string ?

• Reply • Share ›

Abdulrahman Hatim benjamin c • 16 days ago

Because he's actually using C only and not C++

• Reply • Share ›

trent_r3zn0r • 8 months ago

and with the send() command, recv() is more suited. I was getting constant runtime error and couldn't figure out what the problem was, until I saw a hyperlink to recv() command in the man pages of send()

• Reply • Share ›

trent_r3zn0r • 8 months ago

for me, the read() didn't work out. recv() was what worked. and it makes sense too. isn't read() used for file i/o?

1 • Reply • Share ›

Pankaj Chowdary Peddi • a year ago

Header files used here like #include <arpa inet.h=""> , #include <sys socket.h=""> doesn't run in the windows but in linux , so you can do either of these two things...
1. winsock
2. install cygwin {don't forget to download the packages for socket programming as well as g++ while installing , or you can run the setup anytime to add the packages whenever you need} .
what cygwin does is ,allows Unix or Linux applications to be compiled and run on a Windows
operating system from within a Linux-like interface .
I hope it helps.

1 • Reply • Share ›

preetam • a year ago