GeeksforGeeks

A computer science portal for geeks

Custom Search

Suggest a Topic

Login

Write an Article

- Socket Programming in C/C++: Handling multiple clients on server without multi threading
- Vector in C++ STL
- The C++ Standard Template Library (STL)
- Map in C++ Standard Template Library (STL)
- std::sort() in C++ STL
- Arrays in C/C++
- Set in C++ Standard Template Library (STL)

▲

- Priority Queue in C++ Standard Template Library (STL)
- Basic Concepts of Object Oriented Programming using C++
- Stack in C++ STL
- Writing first C++ program : Hello World example
- Basic Input / Output in C++
- C++ Data Types
- Bitwise Operators in C/C++
- Pointers in C and C++ | Set 1 (Introduction, Arithmetic and Array)
- std::string class in C++

- List in C++ Standard Template Library (STL)
- unordered_map in C++ STL
- Converting Strings to Numbers in C/C++
- Queue in Standard Template Library (STL)
- C++ Classes and Objects
- Inheritance in C++
- Different methods to reverse a string in C/C++
- Sort in C++ Standard Template Library (STL)
- Pair in C++ Standard Template Library (STL)
- Commonly

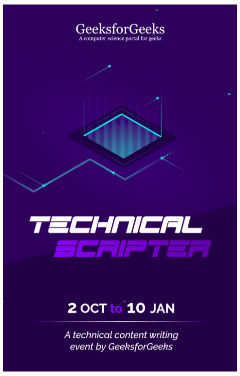
Asked OOP Interview Questions | Set 1

Sorting Vector of Pairs in C++ | Set 1 (Sort by first and second)

C++ string class and its applications

Polymorphism in C++

How to implement Min Heap using STL?

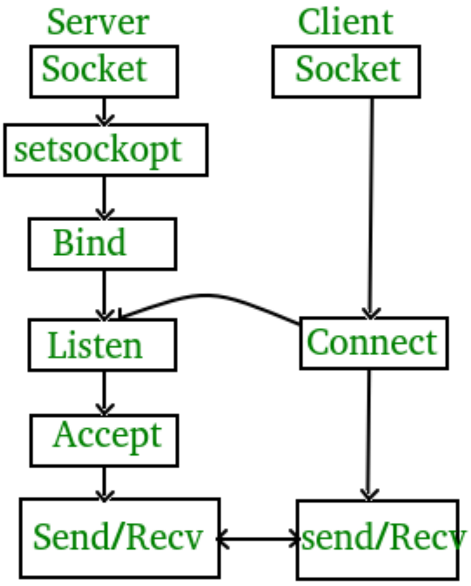

GeeksforGeeks
A computer science portal for geeks
TECHNICAL
SCRIPTER
2 OCT to 10 JAN
A technical content writing event by GeeksforGeeks

Socket Programming in C/C++

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);
```

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.

- **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 *a
```

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

```
// 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_REUS
                                                  &opt, sizeof(
    {
        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 *)&addr
                       (socklen_t*)&addrlen))<0)
    {
        perror("accept");
        exit(EXIT_FAILURE);
    }
    valread = read( new_socket , buffer, 1024);
```

client.c

```c
// Client side C/C++ program to demonstrate Socket programming
#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[])
{
    struct sockaddr_in address;
    int sock = 0, valread;
    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;
    }

    memset(&serv_addr, '0', sizeof(serv_addr));

    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(ser
    {
        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 page and help other Geeks.

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

Recommended Posts:

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

P : A Programming Language

C++ programming and STL facts

Introduction to Go Programming

Natural Language Programming

IDE for Python programming on Windows

Functional Programming Paradigm

Go Programming Language (Introduction)

Introduction to Programming Languages

Introduction of Programming Paradigms

Which Programming Language to Choose?

Fast I/O for Competitive Programming

Top 10 Programming Languages of 2015

C++ tricks for competitive programming (for C++ 11)

Comparing Ruby with other programming languages

Improved By : lcmgcd

Article Tags : C++ GBlog CPP-Library

Practice Tags : CPP

👍 1

3.3

□ To-do □ Done

Based on **10** vote(s)

Feedback Add Notes 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.

Load Comments Share this post!

▲

GeeksforGeeks
A computer science portal for geeks

710-B, Advant Navis Business Park,
Sector-142, Noida, Uttar Pradesh - 201305
feedback@geeksforgeeks.org

COMPANY

About Us
Careers
Privacy Policy
Contact Us

LEARN

Algorithms
Data Structures
Languages
CS Subjects
Video Tutorials

PRACTICE

Company-wise
Topic-wise
Contests
Subjective Questions

CONTRIBUTE

Write an Article
Write Interview Experience
Internships
Videos

▲