Bits and Pieces of Code

My personal place to share projects and code tutorials.

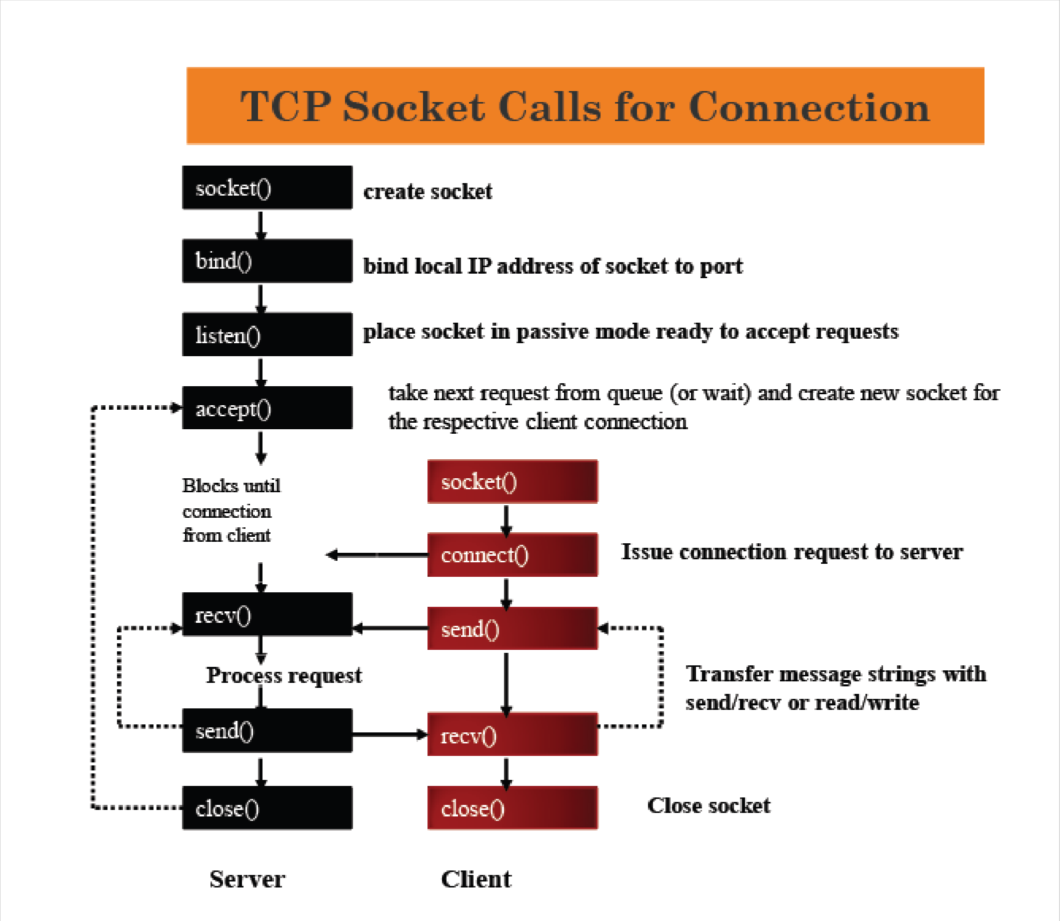
Client-Server chat in C++ using sockets

Karim Oumghar / June 16, 2016

4 Votes

In this tutorial, I'll demonstrate a simple Linux socket program that uses sockets to create a chat between a client and server. Before you read further into this, I recommend reading up on Linux socket programming and a bit on the layers of the internet for background. As this program uses Linux system calls, again, its best to have background on that as well.

Essentially, this program will be a mock instant messaging program that will communicate over TCP using sockets. The client will connect to the server through an IP address specified. The server will listen for up to 5 requests at a time. Afterwards, the server will accept the request to connect from a client and messages will be sent back and forth through a buffer. Should either the client or server decide to stop, at the end we close the sockets and terminate the program.


TCP Socket Calls for Connection
socket()
create socket
bind()
bind local IP address of socket to port
listen()
place socket in passive mode ready to accept requests
accept()
take next request from queue (or wait) and create new socket for the respective client connection
Blocks until connection from client
socket()
connect()
Issue connection request to server
recv()
send()
Process request
Transfer message strings with send/recv or read/write
send()
recv()
close()
close()
Close socket
Server
Client

tcp socket calls for connection

server.cpp

```
#include <iostream>
#include <string>
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <netdb.h>
#include <sys/uio.h>
#include <sys/time.h>
#include <sys/wait.h>
#include <fcntl.h>
#include <fstream>
using namespace std;
//Server side
int main(int argc, char *argv[])
{
    //for the server, we only need to specify a port number
    if(argc != 2)
    {
        cerr << "Usage: port" << endl;
        exit(0);
    }
    //grab the port number
```

```
    bzero((char*)&servAddr, sizeof(servAddr));
    servAddr.sin_family = AF_INET;
    servAddr.sin_addr.s_addr = htonl(INADDR_ANY);
    servAddr.sin_port = htons(port);

    //open stream oriented socket with internet address
    //also keep track of the socket descriptor
    int serverSd = socket(AF_INET, SOCK_STREAM, 0);
    if(serverSd < 0)
    {
        cerr << "Error establishing the server socket" << endl;
        exit(0);
    }
    //bind the socket to its local address
    int bindStatus = bind(serverSd, (struct sockaddr*) &servAddr,
        sizeof(servAddr));
    if(bindStatus < 0)
    {
        cerr << "Error binding socket to local address" << endl;
        exit(0);
    }
    cout << "Waiting for a client to connect..." << endl;
    //listen for up to 5 requests at a time
    listen(serverSd, 5);
    //receive a request from client using accept
    //we need a new address to connect with the client
    sockaddr_in newSockAddr;
    socklen_t newSockAddrSize = sizeof(newSockAddr);
    //accept, create a new socket descriptor to
    //handle the new connection with client
    int newSd = accept(serverSd, (sockaddr *)&newSockAddr, &newSoc
    if(newSd < 0)
    {
        cerr << "Error accepting request from client!" << endl;
        exit(1);
    }
    cout << "Connected with client!" << endl;
    //lets keep track of the session time
    struct timeval start1, end1;
    gettimeofday(&start1, NULL);
    //also keep track of the amount of data sent as well
    int bytesRead, bytesWritten = 0;
    while(1)
    {
        //receive a message from the client (listen)
        cout << "Awaiting client response..." << endl;
        memset(&msg, 0, sizeof(msg));//clear the buffer
        bytesRead += recv(newSd, (char*)&msg, sizeof(msg), 0);
        if(!strcmp(msg, "exit"))
        {
            cout << "Client has quit the session" << endl;
            break;
        }
        cout << "Client: " << msg << endl;
        cout << ">";
        string data;
        getline(cin, data);
        memset(&msg, 0, sizeof(msg)); //clear the buffer
        strcpy(msg, data.c_str());
        if(data == "exit")
        {
            //send to the client that server has closed the connec
            send(newSd, (char*)&msg, strlen(msg), 0);
            break;
        }
        //send the message to client
        bytesWritten += send(newSd, (char*)&msg, strlen(msg), 0);
    }
```

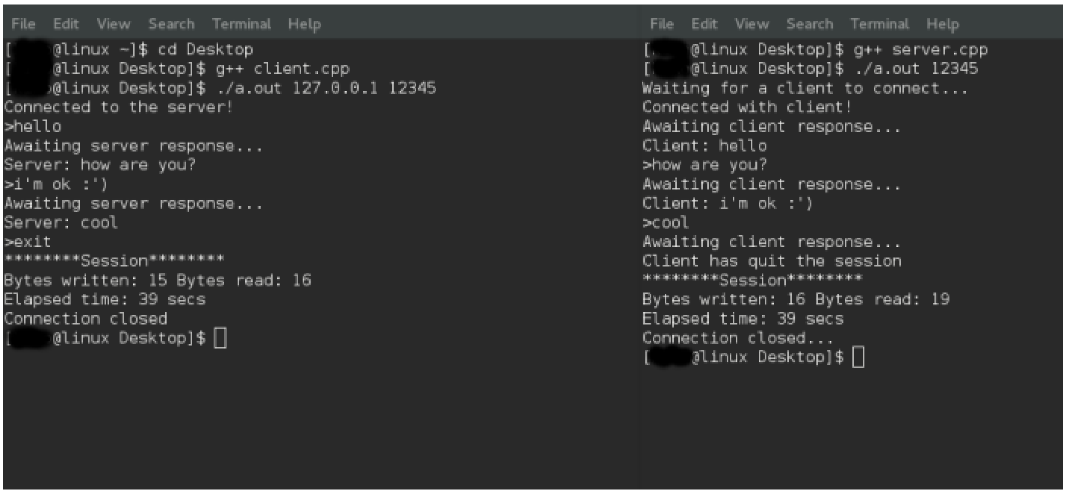
```
        << " secs" << endl;
    cout << "Connection closed..." << endl;
    return 0;
}
```

client.cpp

```
#include <iostream>
#include <string>
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <netdb.h>
#include <sys/uio.h>
#include <sys/time.h>
#include <sys/wait.h>
#include <fcntl.h>
#include <fstream>
using namespace std;
//Client side
int main(int argc, char *argv[])
{
    //we need 2 things: ip address and port number, in that order
    if(argc != 3)
    {
        cerr << "Usage: ip_address port" << endl; exit(0);
    } //grab the IP address and port number
    char *serverIp = argv[1]; int port = atoi(argv[2]);
    //create a message buffer
    char msg[1500];
    //setup a socket and connection tools
    struct hostent* host = gethostbyname(serverIp);
    sockaddr_in sendSockAddr;
    bzero((char*)&sendSockAddr, sizeof(sendSockAddr));
    sendSockAddr.sin_family = AF_INET;
    sendSockAddr.sin_addr.s_addr =
        inet_addr(inet_ntoa(*(struct in_addr*)*host->h_addr_list));
    sendSockAddr.sin_port = htons(port);
    int clientSd = socket(AF_INET, SOCK_STREAM, 0);
    //try to connect...
    int status = connect(clientSd,
                         (sockaddr*) &sendSockAddr, sizeof(sendSock
    if(status < 0)
    {
        cout<<"Error connecting to socket!"<<endl; break;
    }
    cout << "Connected to the server!" << endl;
    int bytesRead, bytesWritten = 0;
    struct timeval start1, end1;
    gettimeofday(&start1, NULL);
    while(1)
    {
        cout << ">";
        string data;
        getline(cin, data);
        memset(&msg, 0, sizeof(msg));//clear the buffer
        strcpy(msg, data.c_str());
        if(data == "exit")
        {
            send(clientSd, (char*)&msg, strlen(msg), 0);
```

```cpp
        {
            cout << "Server has quit the session" << endl;
            break;
        }
        cout << "Server: " << msg << endl;
    }
    gettimeofday(&end1, NULL);
    close(clientSd);
    cout << "********Session********" << endl;
    cout << "Bytes written: " << bytesWritten <<
    " Bytes read: " << bytesRead << endl;
    cout << "Elapsed time: " << (end1.tv_sec- start1.tv_sec)
      << " secs" << endl;
    cout << "Connection closed" << endl;
    return 0;
}
```

Output


File Edit View Search Terminal Help
File Edit View Search Terminal Help

```
[[redacted]@linux ~]$ cd Desktop
[@linux Desktop]$ g++ client.cpp
[@linux Desktop]$ ./a.out 127.0.0.1 12345
Connected to the server!
>hello
Awaiting server response...
Server: how are you?
>i'm ok :')
Awaiting server response...
Server: cool
>exit
********Session********
Bytes written: 15 Bytes read: 16
Elapsed time: 39 secs
Connection closed
[[redacted]@linux Desktop]$
```

```
[[redacted]@linux Desktop]$ g++ server.cpp
[@linux Desktop]$ ./a.out 12345
Waiting for a client to connect...
Connected with client!
Awaiting client response...
Client: hello
>how are you?
Awaiting client response...
Client: i'm ok :')
>cool
Awaiting client response...
Client has quit the session
********Session********
Bytes written: 16 Bytes read: 19
Elapsed time: 39 secs
Connection closed...
[[redacted]@linux Desktop]$
```

Share this:

Twitter Facebook Email Print LinkedIn Reddit

Tumblr Pinterest Pocket WhatsApp Skype Telegram

Like

Be the first to like this.

June 16, 2016 in C++. Tags: C++, chat, client, instant messaging, internet, linux, model, network, networking, programming, server, sockets, tcp

Related posts

Working with a Service for Android

← A Sample Application in Angular.JS

Using the ListBox control in .Net (C#) →

12 thoughts on "Client-Server chat in C++ using sockets"

Mr Anonymous March 5, 2018 at 1:10 pm

Hi, nice guide. But why do you run 12345 in the console?

★ Like

Reply

Karim Oumghar March 5, 2018 at 1:34 pm

Its the port number. You can make your port number any value you wish.

★ Like

Reply

Mr Anonymous March 5, 2018 at 2:16 pm

That's the port! I got it now.

★ Like

Reply

password using cin… then use compare strings in server to see if password is correct. But it doesn't work. I guess my skills are not that good ☹

★ Like

Reply

ki ca September 14, 2018 at 6:09 am

hey how to do multiple clients chatting with single server in linux using c++

★ Liked by 1 person

Reply

Karim Oumghar September 18, 2018 at 6:25 pm

https://www.geeksforgeeks.org/socket-programming-in-cc-handling-multiple-clients-on-server-without-multi-threading/

★ Liked by 1 person

Reply

MegloDon April 10, 2019 at 8:38 am

i faced problem while running.
then i removed the "break;" of line 43 which was in "client.cpp"
now it working well ☺

★ Liked by 1 person

Reply

Hareii December 1, 2019 at 6:42 am

Fazza December 1, 2019 at 6:52 am

i have a solution btw, you should add '#include ' and replace all 'brake' statement to 'return EINVAL'. Hope this help (sorry for my english ☺

★ Like

MUKESH October 19, 2019 at 10:50 am

can i use it for a pc outside local area network

★ Like

Reply

yerikojutsu December 10, 2019 at 8:21 pm

hi. i've tried and face problem with #include .
where does this library goes?

★ Like

Reply

Monika March 25, 2020 at 5:10 am

What if I can do this on Windows? What should I do?

★ Like

Reply

Enter your comment here...

Search ...

Recent Posts

[15 App and Project Ideas]

[Trie data structure from scratch]

[How to handle a failed push when the remote contains work you don't have locally]

[Call upon a REST API online using C#]

[AsyncTask in Java and doing HTTP calls]

Categories

Articles

C#

C++

Data Structures

Database

GDI+

HTML/CSS

Java

Javascript

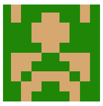
OOP

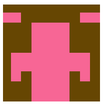
Searching & Sorting

Uncategorized

Karim Oumghar

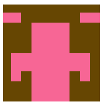
Monika on Client-Server chat in C++ usin...





David on Drawing by mouse on a PictureB…

Karim Oumghar on Drawing by mouse on a PictureB…



David on Drawing by mouse on a PictureB…

Interesantni Blog-ov… on Graphs: Depth First Traversal…

GitHub

My GitHub Profile

LinkedIn

LinkedIn

Follow Blog via Email

Enter your email address to follow this blog and receive notifications of new posts by email.

Enter your email address

Follow

Follow Bits and Pieces of Code

Visitors

Visitors

33,146	1,688
13,201	1,644
7,286	1,593
6,413	1,585
5,137	1,562
5,039	1,498
4,152	1,415
3,951	1,400
3,730	1,339
3,088	1,241
2,922	1,067
2,689	1,031
2,682	1,029
2,299	965
2,297	883
2,282	817
2,199	751
1,788	733
1,724	678
1,723	636

FLAG counter

Blog at WordPress.com.