How to Code a Server and Client in C with Sockets on Linux – Code Examples

By Silver Moon | August 11, 2020

64 Comments

In a previous example we learnt about the basics of socket programming in C. In this example we shall build a basic ECHO client and server. The server/client shown here use TCP sockets or SOCK_STREAM.

Tcp sockets are connection oriented, means that they have a concept of independent connection on a certain port which one application can use at a time.

The concept of connection makes TCP a "reliable" stream such that if errors occur, they can be detected and compensated for by resending the failed packets.

Server

Lets build a very simple web server. The steps to make a webserver are as follows :

1. Create socket
2. Bind to address and port
3. Put in listening mode
4. Accept connections and process there after.

Quick example

Code

```
/*
    C socket server example
*/

#include<stdio.h>
#include<string.h>  //strlen
#include<sys/socket.h>
#include<arpa/inet.h>   //inet_addr
#include<unistd.h>  //write

int main(int argc , char *argv[])
{
    int socket_desc , client_sock , c , read_size;
    struct sockaddr_in server , client;
    char client_message[2000];

    //Create socket
    socket_desc = socket(AF_INET , SOCK_STREAM , 0);
    if (socket_desc == -1)
    {
        printf("Could not create socket");
    }
    puts("Socket created");

    //Prepare the sockaddr_in structure
    server.sin_family = AF_INET;
    server.sin_addr.s_addr = INADDR_ANY;
    server.sin_port = htons( 8888 );

    //Bind
    if( bind(socket_desc,(struct sockaddr *)&server , sizeof(server)) < 0)
    {
        //print the error message
        perror("bind failed. Error");
        return 1;
    }
    puts("bind done");

    //Listen
    listen(socket_desc , 3);

    //Accept and incoming connection
    puts("Waiting for incoming connections...");
    c = sizeof(struct sockaddr_in);

    //accept connection from an incoming client
    client_sock = accept(socket_desc, (struct sockaddr *)&client, (socklen_t*)&c);
    if (client_sock < 0)
    {
        perror("accept failed");
        return 1;
    }
    puts("Connection accepted");

    //Receive a message from client
    while( (read_size = recv(client_sock , client_message , 2000 , 0)) > 0 )
```

```
    {
        //Send the message back to client
        write(client_sock , client_message , strlen(client_message));
    }

    if(read_size == 0)
    {
        puts("Client disconnected");
        fflush(stdout);
    }
    else if(read_size == -1)
    {
        perror("recv failed");
    }

    return 0;
}
```

The above code example will start a server on localhost (127.0.0.1) port 8888
Once it receives a connection, it will read some input from the client and reply back with the same message.
To test the server run the server and then connect from another terminal using the telnet command like this

```
$ telnet localhost 8888
```

Client

Now instead of using the telnet program as a client, why not write our own client program. Quite simple again

Code

```
/*
    C ECHO client example using sockets
*/
#include <stdio.h>  //printf
#include <string.h> //strlen
#include <sys/socket.h> //socket
#include <arpa/inet.h>  //inet_addr
#include <unistd.h>

int main(int argc , char *argv[])
{
    int sock;
    struct sockaddr_in server;
    char message[1000] , server_reply[2000];

    //Create socket
    sock = socket(AF_INET , SOCK_STREAM , 0);
    if (sock == -1)
    {
        printf("Could not create socket");
    }
    puts("Socket created");

    server.sin_addr.s_addr = inet_addr("127.0.0.1");
    server.sin_family = AF_INET;
    server.sin_port = htons( 8888 );

    //Connect to remote server
    if (connect(sock , (struct sockaddr *)&server , sizeof(server)) < 0)
    {
        perror("connect failed. Error");
        return 1;
    }

    puts("Connected\n");

    //keep communicating with server
    while(1)
    {
        printf("Enter message : ");
        scanf("%s" , message);

        //Send some data
        if( send(sock , message , strlen(message) , 0) < 0)
        {
            puts("Send failed");
            return 1;
        }

        //Receive a reply from the server
        if( recv(sock , server_reply , 2000 , 0) < 0)
        {
            puts("recv failed");
            break;
        }
```

```
        puts("Server reply :");
        puts(server_reply);
    }

    close(sock);
    return 0;
}
```

The above program will connect to localhost port 8888 and then ask for commands to send. Here is an example, how the output would look

```
$ gcc client.c && ./a.out
Socket created
Connected
Enter message : hi
Server reply :
hi
Enter message : how are you
```

Server to handle multiple connections

The server in the above example has a drawback. It can handle communication with only 1 client. Thats not very useful.

One way to work around this is by using threads. A thread can be assigned for each connected client which will handle communication with the client.

Code example

Code

```
/*
    C socket server example, handles multiple clients using threads
*/

#include<stdio.h>
#include<string.h>  //strlen
#include<stdlib.h>  //strlen
#include<sys/socket.h>
#include<arpa/inet.h>   //inet_addr
#include<unistd.h>  //write
#include<pthread.h> //for threading , link with lpthread

//the thread function
void *connection_handler(void *);

int main(int argc , char *argv[])
{
    int socket_desc , client_sock , c , *new_sock;
```

```
struct sockaddr_in server , client;

//Create socket
socket_desc = socket(AF_INET , SOCK_STREAM , 0);
if (socket_desc == -1)
{
    printf("Could not create socket");
}
puts("Socket created");

//Prepare the sockaddr_in structure
server.sin_family = AF_INET;
server.sin_addr.s_addr = INADDR_ANY;
server.sin_port = htons( 8888 );

//Bind
if( bind(socket_desc,(struct sockaddr *)&server , sizeof(server)) < 0)
{
    //print the error message
    perror("bind failed. Error");
    return 1;
}
puts("bind done");

//Listen
listen(socket_desc , 3);

//Accept and incoming connection
puts("Waiting for incoming connections...");
c = sizeof(struct sockaddr_in);


//Accept and incoming connection
puts("Waiting for incoming connections...");
c = sizeof(struct sockaddr_in);
while( (client_sock = accept(socket_desc, (struct sockaddr *)&client, (socklen_t*)&c)) )
{
    puts("Connection accepted");

    pthread_t sniffer_thread;
    new_sock = malloc(1);
    *new_sock = client_sock;

    if( pthread_create( &sniffer_thread , NULL ,  connection_handler , (void*) new_sock) < 0)
    {
        perror("could not create thread");
        return 1;
    }

    //Now join the thread , so that we dont terminate before the thread
    //pthread_join( sniffer_thread , NULL);
    puts("Handler assigned");
}

if (client_sock < 0)
{
    perror("accept failed");
    return 1;
}
```

```
    return 0;
}

/*
 * This will handle connection for each client
 * */
void *connection_handler(void *socket_desc)
{
    //Get the socket descriptor
    int sock = *(int*)socket_desc;
    int read_size;
    char *message , client_message[2000];

    //Send some messages to the client
    message = "Greetings! I am your connection handler\n";
    write(sock , message , strlen(message));

    message = "Now type something and i shall repeat what you type \n";
    write(sock , message , strlen(message));

    //Receive a message from client
    while( (read_size = recv(sock , client_message , 2000 , 0)) > 0 )
    {
        //Send the message back to client
        write(sock , client_message , strlen(client_message));
    }

    if(read_size == 0)
    {
        puts("Client disconnected");
        fflush(stdout);
    }
    else if(read_size == -1)
    {
        perror("recv failed");
    }

    //Free the socket pointer
    free(socket_desc);

    return 0;
}
```

Run the above server and connect from multiple clients and it will handle all of them. There are other ways to handle multiple clients, like select, poll etc.

We shall talk about them in some other article. Till then practise the above code examples and enjoy.

CATEGORY: C

TAGS: SOCKET PROGRAMMING

About Silver Moon

A Tech Enthusiast, Blogger, Linux Fan and a Software Developer. Writes about Computer hardware, Linux and Open Source software and coding in Python, Php and Javascript. He can be reached at binarytides@gmail.com.

View all posts by Silver Moon →

64 thoughts on “ How to Code a Server and Client in C with Sockets on Linux – Code Examples ”

Pingback: Resolved: How to host a website in my network (localhost) using Java [closed] - Daily Developer Blog

yashar amirabadii

May 11, 2021 at 5:05 pm

hii, how to run it and send data

plz help me!

Minor

July 17, 2020 at 11:11 am

If any one looking for mac os sample like me, this will help you.
https://razibdeb.blogspot.com/2020/07/simple-bsd-socket-server-client-example.html

---

Silver Moon Post author

August 11, 2020 at 12:12 pm

thanks for sharing.

---

jknjk

November 29, 2017 at 6:06 pm

The multiple client server is all messed up and doesn't even work properly, trash and useless. If you're posting something to help, at least post it right.

Silver Moon Post author

May 17, 2020 at 4:04 pm

fixed it. try to compile the code now.
it should work.

Zamer Chaudhary

April 16, 2022 at 2:02 pm

Hey sir i need you help related to Develop a client/server application using Linux TCP sockets and the C programming language.
I will share the more information on mail. Please respond me.
aaliyahnza@gmail.com

Tim

May 7, 2017 at 8:08 pm

I found two problem with the threaded server code:

1. The file descriptor is never closed when the thread is freed. After a while this eats up all of the file descriptors in the operating system. To fix this, add close(sock); just before free(sock_desc), like so:
//Free the socket pointer
close(sock)
free(socket_desc);
return 0;

2. There is a memory leak that can be fixed by uncommenting //pthread_join(sniffer_thread , NULL);

When these two issues are corrected it works perfectly for me.

Alex

June 8, 2017 at 4:04 pm

Yes, it fixes the bug with memory leakage but disables the multiple clients functionality, so this example doesn't really work :(I cann't solve this problem yet.

Beau Carlson

April 17, 2021 at 11:11 pm

Were you able to? I have a project coming up that this could be very useful for

---

proxy998inchains

April 27, 2017 at 4:04 pm

why do i get i connection error…..should i change port or there is another solution…..

David

November 5, 2021 at 1:01 am

I'm wondering that, too. As soon as I change from the localhost address, I start getting message refused on the client.

Artem

November 30, 2016 at 10:10 pm

Hello. I cannot compile client example. Got that message: client.c:60:5: warning: implict declaration of function ‘close’ (-Wimplict-function-declaration) close(sock);

Adithiya

June 29, 2017 at 11:11 am

Just add the header file in the program “#include”

---

bit-pressure

July 25, 2017 at 2:02 am

You have to include this: #include

---

saqib

October 6, 2016 at 11:11 pm

what should i do to make the client ask for only one message and server prints it completely but just once. i am using single client code.

Ersan Altun

May 5, 2016 at 3:03 pm

hello Sir

i want to make a process , such as a reverse the message. which has sent by clients and server goes to reverse it ,repost it back. How could i do this function ? please help me . thanks

---

Fabrizio Guespe

July 13, 2015 at 9:09 am

Sometimes i get error:invalid operands to binary expression in this line

if(bind(socket_desc,(struct sockaddr *)&server , sizeof(server)) < 0)

Sometimes i do and sometimes not, very wierd

Arun Kumar

June 30, 2015 at 5:05 pm

when i excute the socket program.using switch case function. i am getting my input in server.. itself when i excute…plz help it

---

Aja

March 18, 2014 at 6:06 pm

Can someone explain the pthread_join? Why is it commented out? I assume we need it, but where should it go in the code?

ChrisX

March 10, 2014 at 4:04 am

http://stackoverflow.com/questions/22289163/socket-server-hangs

This code itself has some serious issues in it. Do NOT use it, on the long term if you try to build a server on this it will hang after a time.

Bruno

November 9, 2017 at 4:04 pm

Not if you understand what the code does and how to correct the mistakes in it, it hangs after a while due to zombie threads and not closing the file descriptor after use!

A lot of people just copy an paste without actually understanding what each line does, there is no wonder it doesn't work for people who do this! How many times have you found code on the web that work out of the box without having to modify it? For me, never!

Henry

September 8, 2019 at 4:04 am

And whats the real good code?

---

keen123

January 4, 2014 at 11:11 pm

Nice tutorial. I do have a question:
In this program we're using INADDR_ANY as the server address, which means listen on all the available IP address.
How to re-write a this program if I want to achieve this:
1. Server should listen on two specific sockets lets say for example 192.168.1.1:8888 & 192.168.1.2:8888?

Can you please demonstrate the same, it would be really helpful for me?

Murad Ali

December 18, 2013 at 11:11 pm

Kindly any one tell me how we can make this code so that client can communicate with client in multi clients

---

3bood

November 16, 2013 at 4:04 am

Hey every one; i tried to compiler the multithraded server code but i got this message:

/tmp/ccTHnrln.o: In function `main':
thread.c:(.text+0x142): undefined reference to `pthread_create'
collect2: ld returned 1 exit status

Brennan

November 17, 2013 at 7:07 am

add -lpthread flag when you compile

---

shubham

April 18, 2014 at 10:10 am

how it can be implemented…? please

Nayank

September 12, 2014 at 6:06 pm

Hey you got a solution? I am also stuck with this problem.

Cephas

April 22, 2016 at 3:03 pm

how exactly is it added cuz i hav same prob

Dervine

July 2, 2016 at 5:05 am

Hey Cephas….while compiling type,

cc -pthread -o server server.c

Prabhat Kumar Suman

July 2, 2016 at 6:06 pm

Compile it like gcc -lthread client.c . Do similarly for server program.

Prabhat Kumar Suman

July 2, 2016 at 6:06 pm

-pthread

---

lindy

July 1, 2016 at 6:06 pm

hii, did you get a solution to that error because I'm having the same issue and i don't know how to go about

---

Javier

October 16, 2013 at 2:02 am

Silver,
Excellent code. Just what I needed. Thank you.

Greetings from Argentina,
Javier

Lokoko

October 14, 2013 at 12:12 am

Hi Silver, thanks for every tutorial you wrote here, they have been very useful :). I do have a question, though: I need to make a server write a client's message into every other client connected to simulate a chatroom. So if client 1 writes "hello", client 2 and 3 see in their own screen "Client 1 wrote: hello". Any ideas to do this? Thanks for everything!.

---

oladunk123

October 9, 2013 at 2:02 pm

Thanks for writing tutorials, but as far as the multi client versions goes, both this version and the revised version posted below are prone to errors – for different reasons. The original version containing a buffer overflow/possible access violation and the revised version containing a race condition.

Irfan Doank

September 26, 2013 at 10:10 am

great code…i have use this code to receive data from gps thanks…but when i try receive data from vt310 gps meitrack, i just receive $$, can you help me?sorry for my bad english…

Teem

August 12, 2013 at 5:05 pm

how to kick client if send wrong message?

Marc

August 8, 2013 at 2:02 pm

Very nice well explained piece of work. Got it to build and run on a PC with eclipse and Cygwin compiler in a few minutes :-) , think I needed to add one stupid “include ” for the definition of “close”.

Hadri Rahman

June 19, 2013 at 2:02 pm

I got the multithreaded code to work, but when I send a message using the client, I get this:

$ ./echoclient.out
Socket created

Connected

Enter message: Hello
Server reply:

Hi There! I'm the connection handler.

Type the message and the server shall repeat it.

Enter message: Hello
Server reply:
Hellohere! I'm the connection handler.

Type the message and the server shall repeat it.

Enter message: Help
Server reply:
Hellohere! I'm the connection handler.

Type the message and the server shall repeat it.

Enter message: this is stupid
Server reply:
Helpohere! I'm the connection handler.

Type the message and the server shall repeat it.

Enter message: Server reply:
thisisere! I'm the connection handler.

Type the message and the server shall repeat it.

Enter message: Server reply:
stupidere! I'm the connection handler.

Type the message and the server shall repeat it.

Any idea what to do?

Lokoko

October 13, 2013 at 3:03 am

Same thing happened to me. No idea why :/

---

serg

October 21, 2013 at 12:12 am

The problem is in reusind “client_message” buffer. Replace in server code:

```
while( (read_size = recv(client_sock , client_message , 2000 , 0)) > 0 )
{
//Send the message back to client
write(client_sock , client_message , strlen(client_message));
}
```

to:

```
while( (read_size = recv(client_sock , client_message , 2000 , 0)) > 0 )
{
//Send the message back to client
write(client_sock , client_message , read_size);
}
```

This will send exactly received number of bytes, not whole string.

Pranav

November 3, 2014 at 1:01 pm

sorry to say but this suggestion did not help in solving the problem…
the same thing continues
:(

---

saqib

October 6, 2016 at 11:11 pm

pranav i used the single client code but had the same problem. so what you can do in every code of server and client after printing initialize the char array to null. worked for me.

Alexander Scoutov

May 28, 2014 at 1:01 pm

I just put
memset(client_message,”,sizeof(client_message));

into the end of the block
while((read_size = recv(client_sock , client_message , 2000 , 0)) > 0)

Loca Iino

March 15, 2015 at 1:01 pm

thanks!

Martin

March 19, 2016 at 3:03 am

I solved that issue adding a “+1" after every use of the function strlen(), i.e.

write(client_sock , client_message , strlen(client_message) + 1);

strlen returns the length of the c string omitting the the terminating null character, so write sends the string without it. you receive it fine in the receive buffer, but when you try to read from it you read all the characters from the beginning of buffer until it finds a random occurring null character. You could null out your buffer before reading or writing like so:

memset(client_message,”\n”,sizeof(client_message));

I think sending the null character is simpler.

Hadri Rahman

June 19, 2013 at 1:01 pm

Hey, I tried running the multithreaded server code, but when I compiled it I received this error message:

echoserver.c:88:1: error: expected declaration or statement at end of input

Where line 88 is the closing curly bracket just after :
free(socket_desc);
return 0;
}

I’m not sure what went wrong, any suggestions?

jimmy

June 18, 2013 at 9:09 pm

Hi Silver,

first of all thanks a lot for sharing this. I really appreciate it, so thanks again.

Here come my questions, just in order to clarify:

1. You declare new_sock as a pointer and dynamically allocate e new byte each time a new client connects. Why didn't you just declared it as a normal int, assigning the pointer to client_sock as you already did?

2. Sniffer_thread is only declared once and no dynamic allocation happens after a client connects, so I assume more than one thread (or thread_handler) can refer to the same pthread_t variable without any influence on the previously created/started threads. Am I right?

3. When passing arguments to the thread handler function (I see you pass the new socket variable as a pointer…), these become thread variables, thus are not shared by the different threads, am I right?

4. This would also be valid if passing a client address to the thread handler, thus it would not affect the client address of any other connection handled by the other threads, right?
Again, thanks a lot for the code snippet.

Silver Moon

June 19, 2013 at 12:12 pm

well, the thread handling shown in the code, may not be fully correct, I just wrote it as an experiment and it worked.

1. I created new_sock as a pointer to allocate memory separately for each thread, otherwise it would get overwritten across threads.

2. The sniffer_thread (should have named it something better) variable is created everytime in the while loop, so a new one gets created for every client. Its address is the first parameter to pthread_create function.

3. Not exactly, the new_sock is created by doing a malloc(1); in the main thread, so a new one is created for every thread.

4. You can pass any custom structure variable to the thread function and store anything in it. Like create a struct, fill it in the main thread with socket pointer, address variables, and then pass to thread. Just

make sure that in the main thread it does not get over-written (by using malloc for example)

---

jimmy

June 19, 2013 at 5:05 pm

Hi Silver,

thanks once again for the prompt answer. You are right for what concerns point 4: as long as the thread copies the passed struct in an own local struct, so it gets not over-written by the next thread operation, everything should be fine.

Still, for what concerns points 1 to 3, I'm not quite sure the dynamic allocation was the best choice and also the constant declaration of a new pthread_t var in the while loop sounds a little weird to me. I compared a couple of other threaded server sources and many of them declare the new_sock variable as a global one, passing it as an argument and not caring anymore about it (as it gets copied in a local thread variable) and only declare the pthread_t var once (in the main function variables). It then gets handled by the OS (?) in order to create all the needed and independent threads…as far as I got. Am I missing something or did I get it right?
Thanks again for the clarifications.

Silver Moon

June 20, 2013 at 4:04 pm

I gave the code a little thought. Your arguments are probably right.

Since we are accept and create a thread for only 1 incoming connection at a time, it is okay to use the same new_sock and pthread_t variable. The thread would create its local copy of new_sock and work fine.

I guess its all right to declare pthread_t variable only once and reuse it.

According to http://man7.org/linux/man-pages/man3/pthread_create.3.html
###

Before returning, a successful call to pthread_create() stores the ID
of the new thread in the buffer pointed to by thread; this identifier
is used to refer to the thread in subsequent calls to other pthreads
functions.

###

Therefore you can use the same variable again and again.

I wrote another version of the server following your ideas and it works fine.

https://gist.github.com/silv3rm00n/5821760

---

tibalt

July 29, 2014 at 8:08 am

see line 62 in https://gist.github.com/silv3rm00n/5821760
I have met such issue that the “client_sock” is modified before the “connection_handler” handle the connection. That means one client’s request will be ignored forever!

Nicolas Martin Toriggia

September 18, 2016 at 10:10 pm

Thanks for the example!!!!! Just what I am looking for.
Could you post the code again? the link is down.
thanks a lot!!!

Forvaine

April 14, 2013 at 11:11 pm

Perfect example, thanks a lot!
Will be checking out the select version as well. :)

Sravan Reddy

March 1, 2013 at 1:01 pm

Thanks for the example! Just what I am looking for.
I am wondering whether a similar threaded C-server will be sufficient when the client is a Python Web-handler issuing large number of simultaneous requests to the C-program.

Silver Moon

March 1, 2013 at 2:02 pm

the server and client can be in different languages, it does not matter.

However, how many requests the server can handle will depend on how it is coded. Instead of threads, select function can be used to handle multiple connections.
Check the following article for details on how to use the select function.

https://www.binarytides.com/multiple-socket-connections-fdset-select-linux/

Hari Shankar

February 23, 2013 at 1:01 am

very well laid out and crisp ! Thankyou!

Shai

February 3, 2013 at 1:01 pm

Thank you,

Great examples. it work for me

---

shruthi

January 28, 2013 at 1:01 pm

iam getting connection refused error

Monica

December 3, 2012 at 1:01 pm

I want to include one functionality which is not working here.
If we disconnect the server, all the client processes connecte dto it, will automatically be disconnected.

How to add that functionality in the above code?

Mehmet

December 7, 2021 at 5:05 pm

look at the signals that may help you.

---

leonardo

November 13, 2012 at 11:11 pm

thank you very much for your explanation!, it help me a lot!!
