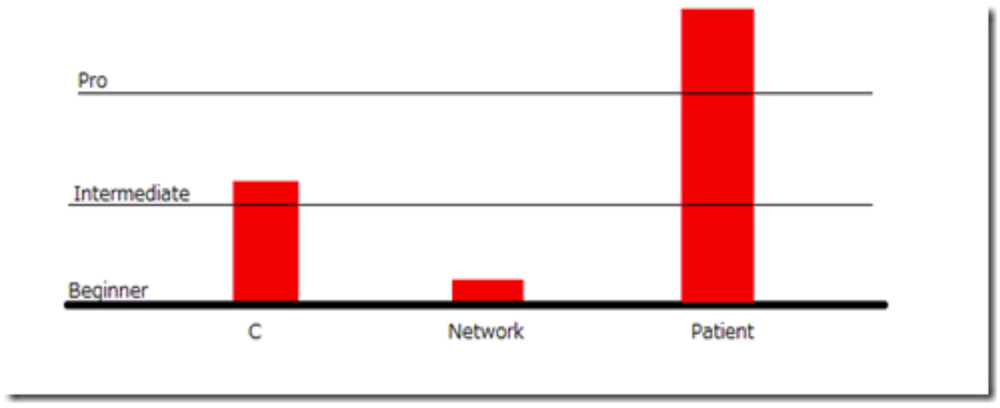
Networking and Socket Programming Tutorial in C

Edison Heng, 21 Aug 2014

★★★★★ 4.92 (28 votes) Rate this:

Networking and Socket programming tutorial in C.

This article is for programmers with the following requirements:



Before you start learning socket programming, make sure you already have a certain basic knowledge of network such as understanding what is IP address, TCP, UDP.

Before we start our tutorial, keep in mind that the following tutorial only works for **Linux OS** environment. If you are using Windows, I have to apologize to you because Windows has its own socket programming and it is different from Linux even though the connection concept is the same. Well, first copy and paste the following code and run it on server and client, respectively.

Both codes can be run on the same computer.

It is always easy to understand after getting the code to work.

Socket-server.c

Hide Shrink ▲ Copy Code

```
#include <sys/socket.h>
```

```
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>

int main(void)
{
  int listenfd = 0,connfd = 0;

  struct sockaddr_in serv_addr;

  char sendBuff[1025];
  int numrv;

  listenfd = socket(AF_INET, SOCK_STREAM, 0);
  printf("socket retrieve success\n");

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

  serv_addr.sin_family = AF_INET;
  serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
  serv_addr.sin_port = htons(5000);

  bind(listenfd, (struct sockaddr*)&serv_addr,sizeof(serv_addr));

  if(listen(listenfd, 10) == -1){
      printf("Failed to listen\n");
      return -1;
  }

  while(1)
    {
      connfd = accept(listenfd, (struct sockaddr*)NULL ,NULL); // accept awaiting
request

      strcpy(sendBuff, "Message from server");
      write(connfd, sendBuff, strlen(sendBuff));

      close(connfd);
      sleep(1);
    }

  return 0;
}
```

Socket-client.c

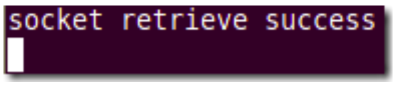
Hide Shrink Copy Code

```
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
```

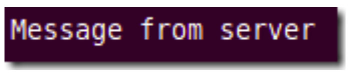
```
#include <errno.h>
#include <arpa/inet.h>

int main(void)
{
  int sockfd = 0,n = 0;
  char recvBuff[1024];
  struct sockaddr_in serv_addr;

  memset(recvBuff, '0' ,sizeof(recvBuff));
  if((sockfd = socket(AF_INET, SOCK_STREAM, 0))< 0)
    {
      printf("\n Error : Could not create socket \n");
      return 1;
    }

  serv_addr.sin_family = AF_INET;
  serv_addr.sin_port = htons(5000);
  serv_addr.sin_addr.s_addr = inet_addr("127.0.0.1");

  if(connect(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr))<0)
    {
      printf("\n Error : Connect Failed \n");
      return 1;
    }

  while((n = read(sockfd, recvBuff, sizeof(recvBuff)-1)) > 0)
    {
      recvBuff[n] = 0;
      if(fputs(recvBuff, stdout) == EOF)
    {
      printf("\n Error : Fputs error");
    }
      printf("\n");
    }

  if( n < 0)
    {
      printf("\n Read Error \n");
    }

  return 0;
}
```

After debugging both source files, run `Socket-server.out`, then run `Socket-client`. Attention here, never mess up with the order of executing `Socket-server.out` and `Socket-client`. `Socket-server` must be executed first, then execute `Socket-client.out` and never try to break `Socket-server` forever loop. It means, you need to open two terminals to run each of the outputs.

When you execute `Socket-cli`, I guess you will get the following result:



```
socket retrieve success
```



```
Message from server
```

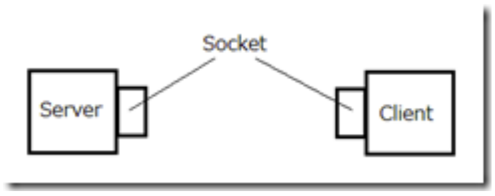
If you see the message above, congratulations, you have success with your first step to networking programming. Otherwise, do some checking on your development environment or try to run some simple code for instance hello world.

Why Both Server and Client on the Same Computer?

The answer is the server and client both are **software** but not hardware. It means what is happening on the top is there are two different software executed. To be more precise, the server and client are two different processes with different jobs. If you are experienced with constructing a server, you might find out that a server can be built on a home computer by installing a server OS. It is because server is a kind of software.

Understand Sockets

Imagine a socket as a seaport that allows a ship to unload and gather shipping, whereas socket is the place where a computer gathers and puts data into the internet.



Configure Socket

Things that need to be initialized are listed as follows:

1. Using TCP or UDP
2. Additional protocol
3. Permit the incoming IP address
4. Assign the port used

At the beginning, a socket function needs to be declared to get the socket descriptor.

Hide Copy Code

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

Domain	AF_UNIX - connect inside same machine AF_INET – connect with different machine
Type	SOCK_STREAM – TCP connection SOCK_DGRAM – UDP connection
Protocol	Define here when there is any additional protocol. Otherwise, define it as 0

Next, decide which struct needs to be used based on what domain is used above.

AF_UNIX	AF_INET
Hide Copy Code `struct sockaddr_un { sa_family_t sun_family ; char sun_path[]; };`	Hide Copy Code `struct sockaddr_in { short int sin_family ; int sin_port; struct in_addr sin_addr; };`
Use struct sockaddr_un if you are using AF_UNIX on your domain. It is required to include <sys/un.h>	Use struct sockaddr_in if you are using AF_INT on your domain.

In this article, I will explain sockadd_in that showed in the code above.

Hide Copy Code	Define the domain used

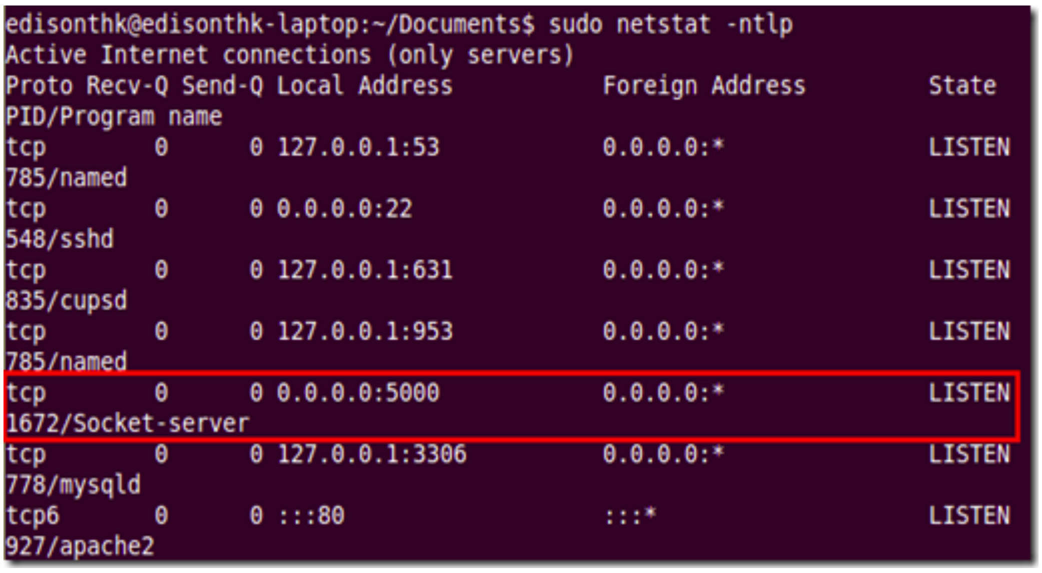
`serv_addr.sin_family = AF_INET;`	
Hide Copy Code `serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);`	Permit any incoming IP address by declaring `INADDR_ANY`
Hide Copy Code `serv_addr.sin_port = htons(5000);`	Declare port 5000 to be used.

Based on the example above, server is using port 5000. You can check it by the following command:

Hide Copy Code

```
sudo netstat -ntlp
```

Then, you will see the following list:

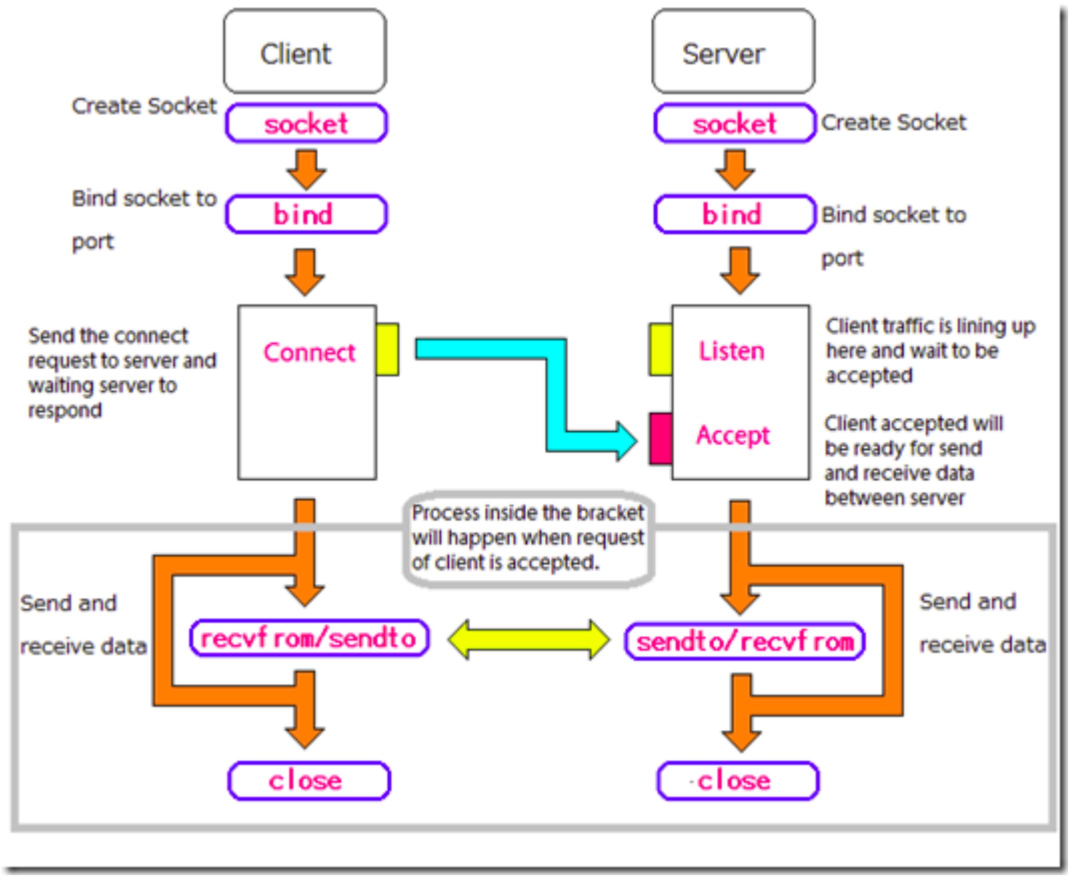


```
edisonthk@edisonthk-laptop:~/Documents$ sudo netstat -ntlp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
tcp        0      0 127.0.0.1:53            0.0.0.0:*               LISTEN
785/named
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
548/sshd
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
835/cupsd
tcp        0      0 127.0.0.1:953           0.0.0.0:*               LISTEN
785/named
tcp        0      0 0.0.0.0:5000            0.0.0.0:*               LISTEN
1672/Socket-server
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
778/mysqld
tcp6       0      0 :::80                   :::*                    LISTEN
927/apache2
```

Inside red bracket, you will find 0.0.0.0:5000 and `Socket-server`, it means port 5000 is used and listen to any valid incoming address.

On client side, `serv_addr.sin_port = htons(127.0.0.1)` is declared in order to listen to the internal network.

The flow chart below shows the interaction between client and server. The flow chart might look complicated but make sure you don't lose your patience due to the following flow chart. Because every process on the flow chart is needed and it acts as a very important role on network connection.


Client
Create Socket
socket
Bind socket to port
bind
Send the connect request to server and waiting server to respond
Connect
Server
socket
Create Socket
bind
Bind socket to port
Listen
Accept
Client traffic is lining up here and wait to be accepted
Client accepted will be ready for send and receive data between server
Process inside the bracket will happen when request of client is accepted.
Send and receive data
recvfrom/sendto
sendto/recvfrom
Send and receive data
close
close

After all setup on `struct sockaddr_in` is done, declare `bind` function. As flow chart, `bind` function must be declared on both server and client.

bind function

`server_socket` & `client_socket`	Put socket description retrieved on the top
`address`	Put `struct sockaddr_in` into it as domain is `AF_INET`. If your domain is `AF_UNIX`, try and put `struct sockaddr_un` here.
`address_len`	Put the length of the address

Server and client will start interacting with each other after the `bind` function and it is the most important session. From what flow chart shows, `listen`, `accept`, `connect`, three functions play very important roles.

Imagine that server looks like an ATM, and only one person can be used the ATM. So, what happens if there are 2 or more people that come at one time? The answer is simple, lining up and wait for the front people to finish using with ATM. It is exactly the same as what is happening in the server.

`Listen` function acts as a waiting room, asking the traffic wait on the waiting room. `Accept` function acts as the person who is asking the traffic waiting inside the waiting room to be ready for the meeting between server. Last, `connect` function acts as the person who wants to carry out some work with the server.

listen function

`server_socket`	Put socket description retrieved on the top

`backlog`	Define the maximum of awaiting request

accept function

`server_socket`	Put socket description retrieved on the top
`client_address`	Put `null` here if there is no special request to specify address.
`address_len`	Put `null` here if second parameter is `null`
`return`	Return information of client socket description. Use it for interaction between client and server.

connect function

`client_socket`	Put socket description retrieved on the top
`address`	Put the `struct sockaddr` defined on the top
`address_len`	Put the length of the address

Finally, after the request is accepted, what should server and client do is send and read data. It is the most simple part in this entire article. `read` function is used to read the buffer data and `write` function is used to send the data. That's all.

read function

`socket_description`	Put server or client socket description depending on reading data from server or client
read buffer	Content of the data retrieved
read buffer length	Length of the output string

write function

`socket_description`	Put server or client socket description depending on sending data to server or client
write buffer	Data to be sent
write buffer length	Length of the output string

Personal Comment

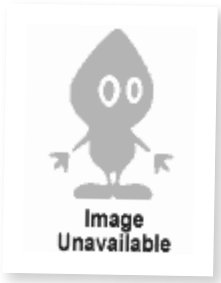
This article was published on 2013/5/1 and I was still new to networking programming at that time. Maybe there is some point that I am not making clear enough. I have tried my best to present all my knowledge to this article. Hope you can get the good basic beginning over here. Thank you!

License

Share

EMAIL

About the Author

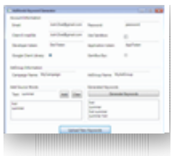


Edison Heng
Japan

Hi! Thank you everyone who reading my article. My major is electronic and programming. Right now I am doing foreign study at Japan. I will like sharing to everyone with my works and if you do interesting with my works, please leave you comment on my blog. Any comments is welcoming.

You may also be interested in...

Programming Windows TCP Sockets in C++ for the Beginner


Generate and add keyword variations using AdWords API

Multithreading Tutorial

Window Tabs (WndTabs) Add-In for DevStudio

SAPrefs - Netscape-like Preferences Dialog

OLE DB - First steps

Comments and Discussions

You must Sign In to use this message board.

Search Comments Go

Spacing Relaxed Layout Open All Per page 25 Update

First Prev Next

good one Member 10068201 27-Nov-16 20:04

nice and very usefull

Sign In · View Thread · Permalink

Socket programming in C Member 12835742 6-Nov-16 11:55

After a long search, it's the first article that helped me to start with this topic.
I am very impressed and grateful. Thank you !
Steven

Sign In · View Thread · Permalink

Re: Socket programming in C vbfingers 1-Jan-17 19:23

Within Codeproject, Also after a lengthy search, I found the same article that you found. One that is easy to understand for beginners. Within each of the "C" programs, do you have all of the includes that it takes. I am running MSVC++ V6. With the includes, I know I am going to be short quite a few includes. Can I get copies of the ones(the .h files) that I am missing? Can you help? Thad.
The US States(utah). Jan 1, 2017. RSVP Please.

Sign In · View Thread · Permalink

Re: Socket programming in C Member 12835742 2-Jan-17 12:34

Hello,
I wish, I could to help you but:
Since my comment, I haven't done any further investigation in this topic due to my other activities, so actually I should be who may ask your help.
Please update this blog about your advances and success
Actually I'm using Linux.
Steven
Canada

Sign In · View Thread · Permalink

My vote of 5 Member 12835742 6-Nov-16 11:44

Excellent article ! It's exactly what I was looking for and never found (until now)
Thank you very much

Sign In · View Thread · Permalink

How to find Time Delay. Govind Nakum 11-Sep-16 20:37

how to establish program where client program and server program are on different system.then client send packet to server and server find out the travelling time from client to sever. In short amount of time that being reached at server from client.

Sign In · View Thread · Permalink

Very nice article **Abhimanyu Aryan** **10-Feb-16 8:58**

Beginner friendly content

Sign In · View Thread · Permalink

tcp ip/ TCP ACK unseen packet **Muhammad Azym** **28-Aug-15 5:24**

Hi,

you have done great job and I learnt a lot from this example. I want to send some pre populated data to server automatically when it make connection with the server. how can I do that I am working in Digi Module Connect me 9210.

Thanks
Azeem

Sign In · View Thread · Permalink

Implementation in LabVIEW? **RAJU PONNAGANTI** **3-Apr-15 10:38**

Can I implement same program in LabVIEW?
In LabVIEW there are TCP/IP and datasockets.which one I have to select to implement?

Raju

Sign In · View Thread · Permalink

bind() in Client **Atinesh** **21-Feb-15 1:00**

I don't think client will issue bind() system call

Sign In · View Thread · Permalink

I don't see bind in client side **Nhat-Tan-Mai** **27-Nov-14 17:47**

As you said it must be bind in both client and server but I can't see bind function in client side. Could you please tell me why?

Sign In · View Thread · Permalink 1.00/5 (1 vote)

Re: I don't see bind in client side Atinesh 21-Feb-15 1:02

I do also agree with you. Client will not issue bind() system call

Sign In · View Thread · Permalink

Re: I don't see bind in client side

mmkh0 30-Apr-15 4:26

Yes, client uses connect instead to poke the server which accepts the client connect attempt.

Also, note that the article says:

Quote:
On client side, serv_addr.sin_port = htons(127.0.0.1) is declared

but actually port 5000 and the IP address in the client is set for the server so should read:

Quote:
On client side,

Hide Copy Code

```
serv_addr.sin_port = htons(5000);
```

and

Hide Copy Code

```
serv_addr.sin_addr.s_addr = inet_addr("127.0.0.1");
```

are declared in order to connect to the server on the internal network

Sign In · View Thread · Permalink

Really good one

Member 11252652 **20-Nov-14 23:57**

Very nice presentation & really a good start for beginners like me.It helped a lot in understanding sockets. Thank you very much

Sign In · View Thread · Permalink

Generalize this example for Windows and Unix

Vaclav Naydenov **29-Oct-14 8:04**

To properly disconnect TCP session from either side one should explicitly call `shutdown()`. Nice diagram also lacks this step.

On the diagram we see a TCP session, but calls `sendto()` and `recvfrom()` are used to send/receive UDP datagrams without explicit connection initiation.

It seems, that `bind()` may fail much more likely than `listen()` (e.g. server TCP port is in use), so add some diagnostics there.

Cross-platform programming targetting both Windows and Unix using socket API is not that hard:
1. Use different set of header files - `windows.h` plus `winsock.h`
2. Remember to initialize socket library - `WSAStartup()`

3. Use `send()` and `recv()` rather than `write()` and `read()`
4. Call `closesocket()` instead of `close()`
Of course, in real life examples things get not so simple.

Below are the Windows-friendly versions.
server.c:

Hide Expand Copy Code

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#ifdef _WIN32
#include <windows.h>
#include <winsock.h>
struct WSAData wsaData;
void init_socket_lib() { WSAStartup(MAKEWORD(1, 1), &wsaData); }
#else
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <errno.h>
#include <sys/types.h>
typedef int SOCKET;
#define INVALID_SOCKET -1
#define closesocket(s) close(s)
void init_socket_lib() {}
#endif

int main(void)
{
  SOCKET listenfd = INVALID_SOCKET, connfd = INVALID_SOCKET;

  struct sockaddr_in serv_addr;

  char sendBuff[1500]; // to fit better within the Ethernet MTU
```

client.c

Hide Expand Copy Code

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#ifdef _WIN32
#include <windows.h>
#include <winsock.h>
struct WSAData wsaData;
void init_socket_lib() { WSAStartup(MAKEWORD(1, 1), &wsaData); }
#else
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <netdb.h>
#include <unistd.h>
#include <errno.h>
#include <arpa/inet.h>
typedef int SOCKET;
#define INVALID_SOCKET -1
#define closesocket(s) close(s)
void init_socket_lib() {}
```

```
#endif

int main(void)
{
  SOCKET sockfd = INVALID_SOCKET;
  int n = 0;
  char recvBuff[1500]; // to fit better within the Ethernet MTU
```

modified 29-Oct-14 14:15pm.

Sign In · View Thread · Permalink 1.00/5 (1 vote)

comment **Member 11052790** **2-Sep-14 1:21**

excellent.., very neat and clear explanation..,

Sign In · View Thread · Permalink

My vote of 5 **Member 11036887** **1-Sep-14 21:54**

helpful article

Sign In · View Thread · Permalink

memset() error **Chad3F** **27-Aug-14 21:20**

In the code you have `memset()`'s like:

Hide Copy Code

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

But `'0'` is not a nul byte, `'\0'` is (or just `0` without quotes). So you are really setting all bytes to 0x30 (the binary value for `'0'`).

If you set all fields afterward, then it doesn't really matter (and the `memset()` is technically redundant). But in cases were there are extension fields (such as `sockaddr_in` on some platforms, which also have the field `sin_len`), there may be data set to values that the OS implementation don't interpret as defaults.

Sign In · View Thread · Permalink 5.00/5 (1 vote)

Re: memset() error Edison Heng 27-Aug-14 22:17

Oops, thank you for mention me about this. Yes, you are right, `memset()` is not needed in this case. The value set at `memset()` is overwrite afterward by `strcpy` function. And some might misunderstood including me, that memset have the ability to allocate memory, but is wrong. It means that `memset` NOT EQUAL TO `malloc`, so it is not necessary in this code.

About `sockaddr_in`, well I am not sure `memset` is required depend on platforms. I have tested this code in Ubuntu LTS 12.04 and Mac OS X. It shows me that `sockaddr` is not require `memset`.

Thanks again for your knowledge sharing. You comment is so helpful to me.

Sign In · View Thread · Permalink 5.00/5 (1 vote)

not working **Member 10902205** **18-Aug-14 5:18**

Where is the definitions of functions write and read? And also close and sleep functions give error.

Sign In · View Thread · Permalink

Re: not working Edison Heng 18-Aug-14 15:45

Make sure you are working in linux not windows because my code will only work in linux environment. And, can you provide more details about how you work on it?

Sign In · View Thread · Permalink

comment **Member 10800476** **13-May-14 21:59**

Thanks a lot. You have very good skills to explain things. Excellent.

Sign In · View Thread · Permalink

Re: comment Member 10902205 18-Aug-14 5:17

how that sample can be good. It doesnt work. What is read write functions???

Sign In · View Thread · Permalink

Very nice representation. **BaldevS** **18-Feb-14 20:21**

happy to see the representation of the article and nice information also.

Sign In · View Thread · Permalink

Great Article! Excellent Explanations, code, diagrams.... **Member 10601147** **18-Feb-14 9:16**

Hi Edison,

To echo the other comments, thank you for posting such a well-done article. I have Richard Stevens' TCP/IP books and your writeup is better than anything in them. Did you create the diagrams yourself? They're very professional. I hope you'll post some more articles here soon!

PCB

Sign In · View Thread · Permalink

Last Visit: 31-Dec-99 18:00 Last Update: 22-Jul-17 3:41 Refresh 1 2 Next »

General News Suggestion Question Bug Answer Joke Praise Rant Admin