oleksiiBobko / tcp_server.c

Last active last month • Report abuse

Simple socket server in C using threads (pthread library) Compiles on linux

tcp_server.c

```c
/*
    C socket server example, handles multiple clients using threads
    Compile
    gcc server.c -lpthread -o server
*/

#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;
    struct sockaddr_in server , client;

    //Create socket
    socket_desc = socket(AF_INET , SOCK_STREAM , 0);
```

```
25      if (socket_desc == -1)
26      {
27          printf("Could not create socket");
28      }
29      puts("Socket created");
30
31      //Prepare the sockaddr_in structure
32      server.sin_family = AF_INET;
33      server.sin_addr.s_addr = INADDR_ANY;
34      server.sin_port = htons( 8888 );
35
36      //Bind
37      if( bind(socket_desc,(struct sockaddr *)&server , sizeof(server)) < 0)
38      {
39          //print the error message
40          perror("bind failed. Error");
41          return 1;
42      }
43      puts("bind done");
44
45      //Listen
46      listen(socket_desc , 3);
47
48      //Accept and incoming connection
49      puts("Waiting for incoming connections...");
50      c = sizeof(struct sockaddr_in);
51
52
53      //Accept and incoming connection
54      puts("Waiting for incoming connections...");
55      c = sizeof(struct sockaddr_in);
56          pthread_t thread_id;
57
58      while( (client_sock = accept(socket_desc, (struct sockaddr *)&client, (socklen_t*)&c)) )
59      {
```

```
60          puts("Connection accepted");
61
62          if( pthread_create( &thread_id , NULL ,  connection_handler , (void*) &client_sock) < 0)
63          {
64              perror("could not create thread");
65              return 1;
66          }
67
68          //Now join the thread , so that we dont terminate before the thread
69          //pthread_join( thread_id , NULL);
70          puts("Handler assigned");
71      }
72
73      if (client_sock < 0)
74      {
75          perror("accept failed");
76          return 1;
77      }
78
79      return 0;
80  }
81
82  /*
83   * This will handle connection for each client
84   * */
85  void *connection_handler(void *socket_desc)
86  {
87      //Get the socket descriptor
88      int sock = *(int*)socket_desc;
89      int read_size;
90      char *message , client_message[2000];
91
92      //Send some messages to the client
93      message = "Greetings! I am your connection handler\n";
94      write(sock , message , strlen(message));
```

```
95
96      message = "Now type something and i shall repeat what you type \n";
97      write(sock , message , strlen(message));
98
99      //Receive a message from client
100     while( (read_size = recv(sock , client_message , 2000 , 0)) > 0 )
101     {
102         //end of string marker
103                 client_message[read_size] = '\0';
104
105                 //Send the message back to client
106         write(sock , client_message , strlen(client_message));
107
108                 //clear the message buffer
109                 memset(client_message, 0, 2000);
110     }
111
112     if(read_size == 0)
113     {
114         puts("Client disconnected");
115         fflush(stdout);
116     }
117     else if(read_size == -1)
118     {
119         perror("recv failed");
120     }
121
122     return 0;
123 }
```

jybaek commented on 16 Mar 2017

Hello? I accidentally saw your code, but there are two things that are wrong. First, the used socket is not closed. Second, you have to rethink the return type of pthread_create and modify line 62.

```
if( pthread_create( &thread_id , NULL ,  connection_handler , (void*) &client_sock) < 0)
```

thanks

MAZHARMIK commented on 25 May 2017

Why have you used thread ? Is this because that this code will help to accept multiple client requests ?
I need help in this. Will you please answer me as soon as possible

MAZHARMIK commented on 25 May 2017

If I run this multi threaded server in one terminal and two or three clients in other terminals, and lets say, client1 sent a message and client2 also sent a message, and after that if the server replies then how would I know which client is being sent the message. I wrote a client as shown below and run this multi threaded server . I am stuck in the confusion I mentioned above. Please help me out with this.

`//client:

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

```
int main()
{
int socket_desc, val;
struct sockaddr_in client_addr;
char buffer[256];
socket_desc = socket(AF_INET, SOCK_STREAM, 0);


  printf("Enter the port number\n");
  int port;
  scanf("%d", &port);
  client_addr.sin_family = AF_INET;
  client_addr.sin_addr.s_addr =  INADDR_ANY;
  client_addr.sin_port = htons(port);

  if(connect(socket_desc, (struct sockaddr*)&client_addr, sizeof(client_addr)) == 0)
          printf("CONNECT STATE: Connected to Server on port %d\n", port);
  else
          printf("Connection to server failed !\n");

  while(1)
  {

          printf("Message to server: ");
          bzero(buffer,256);
          scanf("%s", buffer);

          write(socket_desc,buffer,strlen(buffer));
          bzero(buffer,256);

          read(socket_desc,buffer,255);
          printf("Message from server: %s\n",buffer);

  }
          close(socket_desc);
          return 0;


}`
```

nwork commented on 12 Nov 2017

@MAZHARMIK
in regards to your question from my own research use "recvfrom()" and "sendto()"

Francoisbeche commented on 27 Jan 2018

@MAZHARMIK

You can see , the function connection_handler from server take a socket_desc as paramater, so when you accept is beeing called you can for example create a struct representing a client with all information like IP, FD etc.. and stock it in an array, so you can read this array from your thread and know which client is by using your FD as array's index.

maartenintel commented on 2 May 2018 • edited

Better to pass the accepted socket to the thread by value rather than by reference since there might be two accept()s before connection_handler() runs. When this happens, the second accept() overwrites *client_socket* before connection_handler() can grab it into *sock* at line 88 and both threads will get the same socket descriptor. Two threads servicing the same socket is problematic.

praveen-nair commented on 16 May 2018

I see that you have a commented pthread_join (//pthread_join(thread_id , NULL);) So now you are creating 'n' number of threads and not closing the. Why don't you keep an array (thread pool) to save the thread state and clear it as soon as the connection_handler job is completed?

Nazar2 commented on 21 Jun 2020 • edited

Is it true that there is no code to check if thread has finished it's work? What if "client_sock < 0" but there still is some number of unfinished threads?

halloweeks commented on 22 Feb • edited

Check this
https://github.com/halloweeks/networking/blob/main/server.cpp