

Forum:[Hack Forums](#)

Thread: [Fully Commented Source Code of a Simple TCP Server in C/C++](#)

?si?qo? - October 31, 2008, 11:56 am

Simple server that listens on port 10030. Once a client connects to it, the server sends back a message to the client says "Hello To Socket Programming World!" then closes the socket. Use telnet as the client.

make sure to open another terminal & execute the server, the server will be listening on port 10030 for incoming connections, then open another terminal and connect to the server through telnet by entering

```
telnet localhost 10030
```

This means, connect to the same machine on port 10030, where the server is waiting for connections.

```
Code #include<iostream>
#include<string>
#include<sstream>
#include<sys/types.h>
#include<sys/socket.h>
#include<netdb.h>
#include<arpa/inet.h>
#include<errno.h>
using namespace std;

#define PORT "10030" // the port users will be connecting to - the port we (server) will be listening
on
#define BACKLOG 10 // how many pending connections queue will hold

//describe the program
void progDesc()
{
    cout<<"This server listens for a connection from client, andreplies\n";
    cout<<"back to the client with a message, then closes the socket"<<endl;
}

//get sockaddr, IPv4 or IPv6:
void *get_in_addr(struct sockaddr *clientAddress)
{
    void *addr;

    //check the version of client's IP
    if (clientAddress->sa_family == AF_INET) {
        struct sockaddr_in *ipv4 = (struct sockaddr_in *)clientAddress;
        addr = &(ipv4->sin_addr);
        return addr;
    }else{
```

```

    struct sockaddr_in6 *ipv6 = (struct sockaddr_in6 *)clientAddress;
    addr = &(ipv6->sin6_addr);
    return addr;
}
}

```

```

int main()
{
    int serverSockD;        //the server's socket that listen incoming connections
    int clientSockD;        //the client's socket that works as a medium between client and server to
    send(), recv(), etc
    struct addrinfo hints;  //get inputs about the server information

    /**
    gets output about the server information in more specific details.
    If the server has more than one IP Address, servInfo stores them all.
    To obtain all the IP Addresses, you can traverse through a linkedlist
    servInfo->ai_next; every linkedList will carry one IP address with more
    information about that IP address, such as whether its IPv4 or IPv6. However,
    in this example, regardless of how many ip address the server has, we will
    be biniding to any of the ip addresses we have, only one IP & ignore the rest.
    */
    struct addrinfo *servInfo;

    struct sockaddr clientInfo;        //obtains clients information once accept succeed
    socklen_t clientInfo_size;        //holds the size of cleintInfo
    char msg[] = "Hello To Socket Programming World!\n"; //message that is sent back to client

    //set size of hints to zero

    memset(&hints, 0, sizeof hints);

    //fill some of the server info (your current host info)

    hints.ai_family = AF_UNSPEC;

    hints.ai_socktype = SOCK_STREAM;
    hints.ai_flags = AI_PASSIVE;    //your (the server's) local ip address will be specified
                                    //automatically. The server can have more than one ip address.

    //get a more specific server's information, such as IP Addresses, and pass the output to servInfo
    int status = getaddrinfo(NULL, PORT, &hints, &servInfo);

    //make sure getaddrinfo succeeded
    if(status != 0)
    {

        fprintf(stderr, "getaddrinfo error: %s\n", gai_strerror(status));

        return 1;
    }
}

```

```

}

//create a socket - this socket will be used to listen on PORT for incoming connections
serverSockD = socket(servInfo->ai_family, servInfo->ai_socktype, servInfo->ai_protocol);

if(serverSockD == -1 )

{

    cout<<"Error: failed to create a socket.\n";

    return 2;

}

//now bind the server socket to the ip address & port specified
if(bind(serverSockD, servInfo->ai_addr, servInfo->ai_addrlen) == -1)
{
    perror("server: bind");
    return 3;
}

//listen to the incoming connection, take up to 10 connections
if(listen(serverSockD, BACKLOG) == -1)
{
    perror("listen");
    return 4;
}

//get the size of sockaddr struct of clientAddr
clientInfo_size = sizeof clientInfo;

//holds number of bytes sent
int bytes_sent;

//holds an IP address of the client's

char printableClientIP[INET6_ADDRSTRLEN];

//holds the exact IP address for the client() you accepted his/her connection
void *clientIP;

//listen forever, and accept all connections that come in
//The only way we get out of the loop is in case of a crash
//or a user-initiated shutdown of the server
while(1)
{
    //accept pending connection. Information of incoming connection will go to clientInfo
    clientSockD = accept(serverSockD, &clientInfo, &clientInfo_size);

    if(clientSockD == -1)

```

```

{
    perror("accept");
    return 5;
}

//send msg to the client & number of bytes sent will be returned
bytes_sent = send(clientSockD, msg, strlen(msg), 0);

//get_in_addr gives you the current client IP Address
clientIP = get_in_addr(&clientInfo);

//convert clientIP from numeric binary form to standard text form
inet_ntop(clientInfo.sa_family, clientIP, printableClientIP, sizeof printableClientIP);

//print to screen the IP Address of the client & how many bytes are sent
printf("Sent %d bytes to client : %s\n", bytes_sent, printableClientIP);

//close the current connection & repeat the loop to take the next connection
close(clientSockD);
}

//free linkedlist of struct addrinfo *servInfo

freeaddrinfo(servInfo);

return 0;
}

```

If you gotta question, lemme know, i'll answer what i know! ?

Compiles & works under linux.

an balker - December 28, 2019, 5:00 am

(Oct 31, 2008 11:56 AM)?si?qqq? Wrote: [\[CITE\]](#) Simple server that listens on port 10030. Once a client connects to it, the server sends back a message to the client says "Hello To Socket Programming World!" then closes the socket. Use telnet as the client.

make sure to open another terminal & execute the server, the server will be listening on port 10030 for incoming connections, then open another terminal and connect to the server through telnet by entering

```
telnet localhost 10030
```

This means, connect to the same machine on port 10030, where the server is waiting for connections.

```

Code #include<iostream>
#include<string>
#include<sstream>
#include<sys/types.h>
#include<sys/socket.h>
#include<netdb.h>
#include<arpa/inet.h>
#include<errno.h>

```

```
using namespace std;
```

```
#define PORT "10030" // the port users will be connecting to - the port we (server) will be listening on
#define BACKLOG 10 // how many pending connections queue will hold
```

```
//describe the program
```

```
void progDesc()
```

```
{
    cout<<"This server listens for a connection from client, and replies\n";
    cout<<"back to the client with a message, then closes the socket"<<endl;
}
```

```
//get sockaddr, IPv4 or IPv6:
```

```
void *get_in_addr(struct sockaddr *clientAddress)
```

```
{
    void *addr;
```

```
//check the version of client's IP
```

```
if (clientAddress->sa_family == AF_INET) {
    struct sockaddr_in *ipv4 = (struct sockaddr_in *)clientAddress;
    addr = &(ipv4->sin_addr);
    return addr;
}else{
    struct sockaddr_in6 *ipv6 = (struct sockaddr_in6 *)clientAddress;
    addr = &(ipv6->sin6_addr);
    return addr;
}
}
```

```
int main()
```

```
{
    int serverSockD; //the server's socket that listen incoming connections
    int clientSockD; //the client's socket that works as a medium between client and server to
    send(), recv(), etc
    struct addrinfo hints; //get inputs about the server information
```

```
/**
gets output about the server information in more specific details.
If the server has more than one IP Address, servInfo stores them all.
To obtain all the IP Addresses, you can traverse through a linkedlist
servInfo->ai_next; every linkedList will carry one IP address with more
information about that IP address, such as whether its IPv4 or IPv6. However,
in this example, regardless of how many ip address the server has, we will
be biniding to any of the ip addresses we have, only one IP & ignore the rest.
*/
struct addrinfo *servInfo;

struct sockaddr clientInfo; //obtains clients information once accept succeed
socklen_t clientInfo_size; //holds the size of cleintInfo
char msg[] = "Hello To Socket Programming World!\n"; //message that is sent back to client
```

```
//set size of hints to zero
```

```
memset(&hints, 0, sizeof hints);
```

```
//fill some of the server info (your current host info)
```

```
hints.ai_family = AF_UNSPEC;
```

```
hints.ai_socktype = SOCK_STREAM;
```

```
    hints.ai_flags = AI_PASSIVE;    //your (the server's) local ip address will be specified  
                                   //automatically. The server can have more than one ip address.
```

```
//get a more specific server's information, such as IP Addresses, and pass the output to servInfo  
int status = getaddrinfo(NULL, PORT, &hints, &servInfo);
```

```
//make sure getaddrinfo succeeded  
if(status != 0)
```

```
{
```

```
    fprintf(stderr, "getaddrinfo _error_: %s\n", gai_strerror(status));
```

```
    return 1;
```

```
}
```

```
    //create a socket - this socket will be used to listen on PORT for incoming connections  
    serverSockD = socket(servInfo->ai_family, servInfo->ai_socktype, servInfo->ai_protocol);
```

```
    if(serverSockD == -1 )
```

```
{
```

```
    cout<<"_error_: failed to create a socket.\n";
```

```
    return 2;
```

```
}
```

```
//now bind the server socket to the ip address & port specified
```

```
if(bind(serverSockD, servInfo->ai_addr, servInfo->ai_addrlen) == -1)
```

```
{
```

```
    perror("server: bind");
```

```
    return 3;
```

```
}
```

```
//listen to the incoming connection, take up to 10 connections
```

```
if(listen(serverSockD, BACKLOG) == -1)
```

```
{
```

```
    perror("listen");
```

```
    return 4;
```

```

}

//get the size of sockaddr struct of clientAddr
clientInfo_size = sizeof clientInfo;

//holds number of bytes sent
int bytes_sent;

//holds an IP address of the client's

char printableClientIP[INET6_ADDRSTRLEN];

//holds the exact IP address for the client() you accepted his/her connection
void *clientIP;

//listen forever, and accept all connections that come in
//The only way we get out of the loop is in case of a crash
//or a user-initiated shutdown of the server
while(1)
{
//accept pending connection. Information of incoming connection will go to clientInfo
clientSockD = accept(serverSockD, &clientInfo, &clientInfo_size);

if(clientSockD == -1)
{
perror("accept");
return 5;
}

//send msg to the client & number of bytes sent will be returned
bytes_sent = send(clientSockD, msg, strlen(msg), 0);

//get_in_addr gives you the current client IP Address
clientIP = get_in_addr(&clientInfo);

//convert clientIP from numeric binary form to standard text form
inet_ntop(clientInfo.sa_family, clientIP, printableClientIP, sizeof printableClientIP);

//print to screen the IP Address of the client & how many bytes are sent
printf("Sent %d bytes to client : %s\n", bytes_sent, printableClientIP);

//close the current connection & repeat the loop to take the next connection
close(clientSockD);
}

//free linkedlist of struct addrinfo *servInfo

freeaddrinfo(servInfo);

return 0;
}

```

If you gotta question, lemme know, i'll answer what i know! ?

Compiles & works under linux.
Can you implement in C

akali - December 28, 2019, 6:05 am

(Dec 28, 2019 06:00 AM)an balker Wrote: [\[CITE\]](#) Can you implement in C
That post is from 2008. And it's already in C.

nascoman - December 31, 2019, 8:54 pm

I need someone to coded private rat which I am willing to pay and I please if you are ripped do not contact me. my discord id jameslagesse1111#8741

SIM-3364521 - January 2, 2020, 7:20 am

(Dec 28, 2019 07:05 AM)akali Wrote: [\[CITE\]](#) That post is from 2008. And it's already in C.nah, its not there are usages of the standard c++ library like std::string etc.

(Dec 31, 2019 09:54 PM)nascoman Wrote: [\[CITE\]](#) I need someone to coded private rat which I am willing to pay and I please if you are ripped do not contact me. my discord id jameslagesse1111#8741stop the spam, i saw you post that in some other threads as well.

akali - January 2, 2020, 4:33 pm

(Jan 2, 2020 08:20 AM)Erarnitox Wrote: [\[CITE\]](#) nah, its not there are usages of the standard c++ library like std::string etc.

stop the spam, i saw you post that in some other threads as well.

Remove using namespace std;

Remove iostream,

Remove string

insert stdio.h

replace cout with printf. The code is still in C despite it using iostream and string.