

Stack Overflow is a question and answer site for professional and enthusiast programmers. It's 100% free, no registration required.

Take the 2-minute tour ✕

Sending a structure from client to server in TCP/IP socket

I have the following structure in my header file `structure.h`. Now i need to use this structure in my `main.c` file.

I need to fill this structure with some values and I need to send this from TCP/IP client to the TCP/IP server on the same system.

```
#ifndef STRUCTURE_H
#define STRUCTURE_H

typedef struct
{
    unsigned int variable3;
    char variable4[8];
}NUMBER_ONE,*PNUMBER_ONE;

typedef struct
{
    unsigned int variable5;
    char variable6[8];
}NUMBER_TWO,*PNUMBER_TWO;

typedef struct
{
    char name[32];
    unsigned int a;
    unsigned int b;
    NUMBER_ONE variable1;
    NUMBER_TWO variable2;
}NUMBER_THREE,*PNUMBER_THREE;

#endif
```

I have tried this but, I am not good in C, so please can anyone tell me how to do it, by taking the above structure as an example? Till socket connection establishment is ok for me, but after establishing connection, how do I send this structure from the client to the server?

I am doing this in my Linux Ubuntu 12.04 system.

[c](#) [sockets](#) [tcp](#)

edited Dec 16 '13 at 11:02



[Holger Just](#)

16.9k 4 32 51

asked Dec 16 '13 at 10:57



[user3094304](#)

1

- 1 I would not recommend doing that, since different machines can have different byte ordering (Big Endian/Little Endian), structure packing... etc, so the structure might not be correctly decoded on the other side. You should better use an interchange format, you can devise your own, or use one of the many existing formats (JSON, BSON, XML, AMF...) – [SirDarius](#) Dec 16 '13 at 11:03

Don't use structs as network protocols. You are introducing half a dozen or so dependencies. Define a wire protocol in octets, and write the code to send and receive it. – [EJP](#) Dec 16 '13 at 23:08

[add a comment](#)

2 Answers

When sending information using sockets three ways used:

- 1)fixed size message (we will use it, plus assuming we are writing on the same machine byteorder match).Simply, like we will send 100 bytes and on receive we will read 100bytes
- 2)message.len + message.(first we send message len then message itself. used for binary send receive mostly)
- 3)marker method (mostly used sending text messages or commands. for exampling marking with \n newline)

Next coming on representing our data (serialize). It is easy with c cause on c we can directly write our object and retrieve it without additional efforts. Object will be the same as in memory.

```
// PNUMBER_THREE structAddr;
send(socket_id, structAddr, sizeof(NUMBER_THREE), 0);
```

or

```
write(socket_id, structAddr, sizeof(NUMBER_THREE));
```

or safer

```
write_socket(socket_id, structAddr, sizeof(NUMBER_THREE));
//It is safer to do so though we are using blocking mode
int write_socket(int fd, const char *buf, int len) {
    int currentsize=0;
    while(currentsize<len){
        int count=write(fd, buf+currentsize, len-currentsize);
        if(count<0) return -1;
        currentsize+=count;
    }
    return currentsize;
}
```

when reading we will be using **the same structure** plus it must meet condition


```
sizeof(NUMBER_THREE)==SizeInsideClient //SizeInsideClient is sizeof on client
SizeInsideClient=sizeof(NUMBER_THREE)
```

```
//SizeInsideClient structure size on client program
assert(sizeof(NUMBER_THREE)==SizeInsideClient);
readblock(socket_id, structAddr, sizeof(NUMBER_THREE));
```

```
int readblock(int fd, char* buffer, int len) {
    int ret = 0;
    int count = 0;
    while (count < len) {
        ret = read(fd, buffer + count, len - count);
        if (ret <= 0) {
            return (-1);
        }
        count += ret;
    }
    return count;
}
```

edited Dec 16 '13 at 13:24

answered Dec 16 '13 at 11:04

 **qwr**
2,209 4 18

Just one simple call to send/write does not necessarily do the job. Please closely read the related man-pages with respect to sockets. – [alk](#) Dec 16 '13 at 11:08

This is the fast and ugly way with a list of caveats as long as your arm. If this is anything other than an experiment please find a way to marshal and un-marshal your data which embeds a little meta-data (and maybe some versioning information). – [Speed8ump](#) Dec 16 '13 at 12:00

@qwr how can i receive that at the receiving end,(in the server side). can you please explain me, since i dont know how to serialize and deserialise the strcture data, can you show that by example. – [user3094304](#) Dec 16 '13 at 12:26

@user3094304 see if it is helpful – [qwr](#) Dec 16 '13 at 12:56

[add a comment](#)

Short example with no error checking:

Server

```
struct sockaddr_in serv_addr;

listenfd = socket(AF_INET, SOCK_STREAM, 0);

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));

listen(listenfd, 10);

while(1)
{
```

```

connfd = accept(listenfd, (struct sockaddr*)NULL, NULL);
snprintf(sendBuff, sizeof(sendBuff), "%.24s\r\n", ctime(&ticks));
write(connfd, sendBuff, strlen(sendBuff));
close(connfd);
sleep(1);
}

```

Client

```

struct sockaddr_in serv_addr;

sockfd = socket(AF_INET, SOCK_STREAM, 0);

serv_addr.sin_family = AF_INET; serv_addr.sin_port = htons(5000);

inet_pton(AF_INET, argv[1], &serv_addr.sin_addr);
connect(sockfd,
(struct sockaddr *)&serv_addr, sizeof(serv_addr))

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

```

Pitfalls

Is not "safe" so send plain struct. Sooner or later you will deal with endianness (byte order), packing (which can still be a problem even with #pragma pack) and sizes of types like 'int' that can vary between platforms

answered Dec 16 '13 at 13:04

 [vlg789](#)
336 1 13

[add a comment](#)

Not the answer you're looking for? Browse other questions tagged [c](#) [sockets](#) [tcp](#) or [ask your own question](#).