

Re: strange problem with send() and recv()

Source: http://coding.derkeiler.com/Archive/C_CPP/comp.lang.c/2005-08/msg01579.html

- *From:* Dave Thompson <david.thompson1@xxxxxxxxxxxxxxxxxxxx>
 - *Date:* Sun, 14 Aug 2005 08:18:05 GMT
-

On 8 Aug 2005 09:03:31 -0700, "Tom" <bauer@xxxxxx> wrote:

> I try to write a simple tcp based job queue server. It's purpose is to
> get commands from a client and store them in a queue (STL). The server

As others have noted the STL part is C++ and offtopic in c.l.C, but this is relatively small and easily ignorable. "Mixed" declarations (that is, not solely at the beginning of a block) and double-slash comments which were introduced in C++ `_are_` standard in C as of C99, but not yet universally implemented. And double-slash comments are still unwise in news postings, since those may get line breaks (wraps) added at various points, and this breaks `//` comments but does not harm `/* */` comments. (It also harms long preprocessor `#directives`, the only other place lines are significant.)

> has a thread which checks periodically the queue, executes the commands
> in it and removes them from the queue after done so. It also has a
> thread for every client connection. I am using the low level SOCKET

Server thread per connection/client scales poorly; see any week (of the last N years) of `comp.programming.threads`. But leave that for now.

> API. My server is a Win32 console app and my client is MFC. The
> followinf struct is passed through the sockets:

```
> typedef struct  
> {  
> int iCmdId;  
> char szJobID[JOBIDLENGTH];  
> char szWorkPath[PATHLENGTH];  
> char szUser[USERLENGTH];  
> } SERVERCMD;  
>
```

In general it's a poor idea to send C-language structs over a network; compilers on different types of systems can lay them out differently, and sometimes also different compilers or the same compiler with different options on the same system type. This is why you see proper network protocols specified in terms of actual bits (or nowadays usually octets) on the wire and not in C or other HLL. But since you apparently are using Wintel and probably the same compiler at both

Re: strange problem with send() and recv()

(all) endpoints, and structs that are `_mostly_` chars, leave that also.

> The client can query the server about the jobs currently queued. The
> server sends then each item (the whole struct) in the queue to the
> client. The wired problem with this function is that for a while
> everything seems fine. I get all queued jobs listed in my clients
> CListBox. But after repeatedly calling the ListJobs() function my data
> get somehow corrupted (although the queue doesn't change). In the
> following I post the sourcode for the server's and client's ListJobs()
> function and the debugging output which clearly shows the problem.

>

> server:

> void ListJobs(SOCKET client)

> {

> int bytesSent = 0;

> char buffer[5];

> itoa(CmdQueue.size(), buffer, 10);

>

itoa() is not standard in C or C++. sprintf() is. If the number of jobs is > 9999 this overflows the buffer, formally causing Undefined Behavior although in practice on nearly all if not all machines this particular UB doesn't actually cause harm until you get to at least 6 digits and probably 8 digits. And the latter at least probably won't happen because you'll hit other limits first. (Like uptime!)

> bytesSent = send(client, (char*)&buffer, sizeof(buffer), 0);

> LogMessage("ListJobs: sent %d bytes -> nJobs=%d\n", bytesSent,

> CmdQueue.size());

>

> for (int i=0; i<CmdQueue.size(); i++)

> {

> SERVERCMD job = CmdQueue[i];

> bytesSent = send(client, (char*)&job, sizeof(SERVERCMD), 0);

Perhaps clearer to use sizeof(job), and for an object/expression (but not a typename) can omit the parentheses = sizeof job .

> LogMessage("ListJobs: sent %d bytes of jobdata\n", bytesSent

>);

> }

> LogMessage("\n");

> }

>

> client:

> SERVERCMD * ListJobs(int * nJobs)

> {

> if (sockClient == NULL)

> return NULL;

>

> int bytesSent,bytesRecv;

> SERVERCMD * jobs;

Re: strange problem with send() and recv()

Re: strange problem with send() and recv()

```
>
> // send list request to server
> SERVERCMD job;
> job.iCmdId = CMD_LSTJOB;
> bytesSent = send( sockClient, (char*)&job, sizeof(SERVERCMD), 0 );
> if ( bytesSent == SOCKET_ERROR )
> return NULL;
>
> // receive number of jobs
> *nJobs = 0;
> char buffer[5];
> bytesRecv = recv( sockClient, (char*)&buffer, sizeof(buffer), 0 );
>
> if ( bytesRecv != SOCKET_ERROR )
> {
> *nJobs = atoi( buffer );
> debug("ListJobs: received %d bytes -> nJobs=%d\n", bytesRecv,
> *nJobs);
>
>
```

If the server numjobs was > 9999 this reads an unterminated (and possibly very wrong) value on which atoi() isn't safe. In fact atoi() isn't safe in the presence of almost any error; strtol (and ul, and ll and ull on C99 systems) handles some errors but not unterminated.

```
> // allocate space for jobs
> size_t jobsSize = *nJobs * sizeof(SERVERCMD);
> jobs = (SERVERCMD*)malloc( jobsSize );
```

The cast wouldn't be needed in C, but is in C++, and you don't check the result is nonnull before using it (below). In C++ it is briefer, clearer, and more robust to just write `jobs = new SERVERCMD [*nJobs]`, which also throws (by default) on allocation failure.

```
> debug("ListJobs: allocating %d bytes\n", jobsSize);
>
> %d expects an int (signed) but jobsSize is size_t which is definitely
> unsigned and may be a different size than int or unsigned int. In C99
> there is a specific modifier for this; in C90 and C++ probably best to
> cast to and specify unsigned long, or perhaps unsigned long long if
> you have it. Or in C++ use << instead, to a stringstream if necessary.
```

```
> // receive jobs
> for (int i=0;i<*nJobs; i++)
> {
> bytesRecv = recv( sockClient, (char*)&(jobs[i]),
> sizeof(SERVERCMD), 0 );
> if ( bytesRecv == SOCKET_ERROR ) return NULL;
> debug("ListJobs: received %d bytes of jobdata\n",
> bytesRecv);
> }
> debug("\n");
```

Re: strange problem with send() and recv()

Re: strange problem with send() and recv()

```
> }
> else
> return NULL;
>
> return jobs;
> }
>
> In the following debugging output 2 jobs were put into the queue. The
> queue was then queried multiple times.
```

> As already noted, recv() on a bytestream protocol like TCP can receive only part of what you asked for (and was sent). You need to use MSG_COMP if available, which I don't think it is on (most?) Winsock, or be prepared to read multiple pieces and combine them.

– David.Thompson1 at worldnet.att.net

• *Follow-Ups:*

◆ [Re: strange problem with send\(\) and recv\(\)](#)

◇ *From:* Keith Thompson

• *References:*

◆ [strange problem with send\(\) and recv\(\)](#)

◇ *From:* Tom

- Prev by Date: [Re: unsigned to signed integer conversion](#)
- Next by Date: [Re: difference in header files](#)
- Previous by thread: [Re: strange problem with send\(\) and recv\(\)](#)
- Next by thread: [Re: strange problem with send\(\) and recv\(\)](#)
- Index(es):
 - ◆ [Date](#)
 - ◆ [Thread](#)