

## Re: Debug Assertion Failure

---

*Source:* <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.mfc/2007-06/msg00876.html>

---

- *From:* MrAsm <[mrasm@xxxxxxx](mailto:mrasm@xxxxxxx)>
  - *Date:* Sat, 16 Jun 2007 08:01:30 GMT
- 

On Sat, 16 Jun 2007 00:15:50 -0700, katz911@xxxxxxxxxx wrote:

I've isolated the problem to the following statement:

```
m_UserFile.ReadFromFile(reinterpret_cast<char*>(&temp_user),iter);
```

Whenever I remove that line from the function, the error doesn't pop; therefore it's probably something in that function.

hmm...

What is your goal?

Are you going to "raw-copy" the 'CUser\_IO temp\_user' from file? I would read/write structure bytes to file *\*only\** if it is a plain simple C structure, e.g.

```
struct User
{
    TCHAR Name[MAX_NAME];
    unsigned int Zip;
    TCHAR Address[MAX_ADDRESS];
    ...
};
```

```
User anUser;
ExampleWriteToFile( &anUser, sizeof(anUser) );
```

Even if you have a member like a CString or a pointer to something, I would *\*not\** read/write structure to file just copying its raw bytes; in these cases a finer serialization mechanics is required.

What is the definition of CUser\_IO ? Is it a plain C structure?

If you want to read/write complex data to file, you might consider serializing using XML (very widespread today! and easy to debug because it's a text format, not a binary format), or using MFC CArchive, ...or something from Boost, or *\*carefully\** write your own

## Re: Debug Assertion Failure

serialization system.

```
bool CFileHandler::ReadFromFile (char* ptr, long offset)
{
....

memset (ptr, 0, m_BlockSize);
```

Is `m_BlockSize == sizeof(CUser_IO)`?

You're clearing memory, you must be sure that you are not touching bytes out-of-boundaries; this may cause memory corruption.

```
in_file.open(m_FileName.c_str(), ios::in | ios::binary);

if (in_file.is_open())
{
in_file.seekg(offset + POINTER_SIZE, ios::beg);
in_file.read(ptr, m_BlockSize);
```

Again: is `'m_BlockSize == sizeof(CUser_IO)'` ?

Is `'offset'` consistent with your binary file format?

(I don't know what is your binary file format that you are using here.)

Another option I thought of is that the stack could simply run out of space (I've looked at the call stack, and it has depth of around 12 levels). Could this be what causes the problems?

....I'm much more suspicious about the way you are reading (and maybe elsewhere writing) data from binary file, the way you are clearing memory with `memset` and using `m_BlockSize`...

MrAsm

.