More Bytes Copied by memcpy in C++ than expected

Issue:

In C++ source, in Visual Studio 2013, memcpy copies more bytes than expected.

Experimentation shows that for this to occur:

- C++ source must be compiled with optimization turned on
- Compilation must be targeting a 64-bit platform
- memcpy target address must be that of an arithmetic item
- memcpy's specified number of bytes to copy must be a
  - o literal
  - o valid arithmetic length

Example:

Example consists of two parts for which source is appended below.

memcpy_test.cpp: Simple driver invoking external function ExtFun01(my_str);

memcpy_test_extfun.cpp: Contains function ExtFun01 which demonstrates two memcpy invocations.

The first memcpy works as expected. In this case the number of bytes to copy, 2, is specified in a variable.

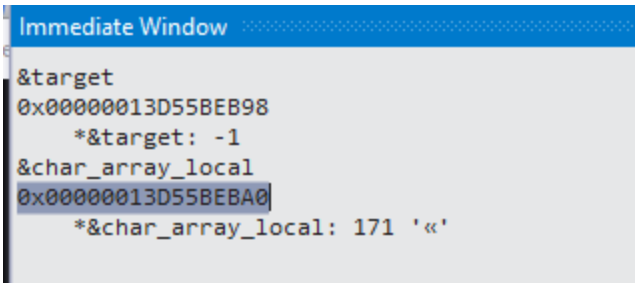
The second memcpy copies 8 bytes of data even though the number of bytes to copy is specified as 2. In this case all of the conditions listed above have been met. In particular the number of bytes to copy, 2, is specified as a literal.

Using Visual Studio 2013, the two parts are compiled and bound into an executable. The build command sequence and compile options are shown below.

Stepping through the execution using the Visual Studio Debugger the following is observed:

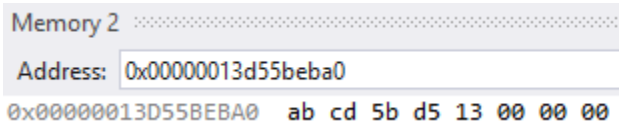
Stop at the first memcpy, before the first memcpy is executed.

In the Debug Immediate Window display the address of the variables "target" and "char_array_local":



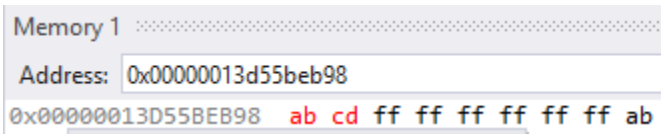
Display the memory at address of “target” variable:

Display the memory at address of “char_array_local”:



Step over first memcpy, stopping at 2nd memcpy, prior to its execution.

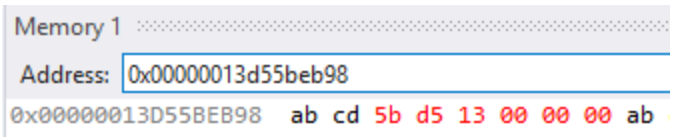
The memory at address of “target” shows:



Note that only the first 2 bytes have been modified by the memcpy, as expected.

Step over the second memcpy:

The memory at address of “target” shows:



All 8 bytes of “target” have been modified, which is not as expected. The first 2 bytes show as black because they have the same value as before the 2nd memcpy.

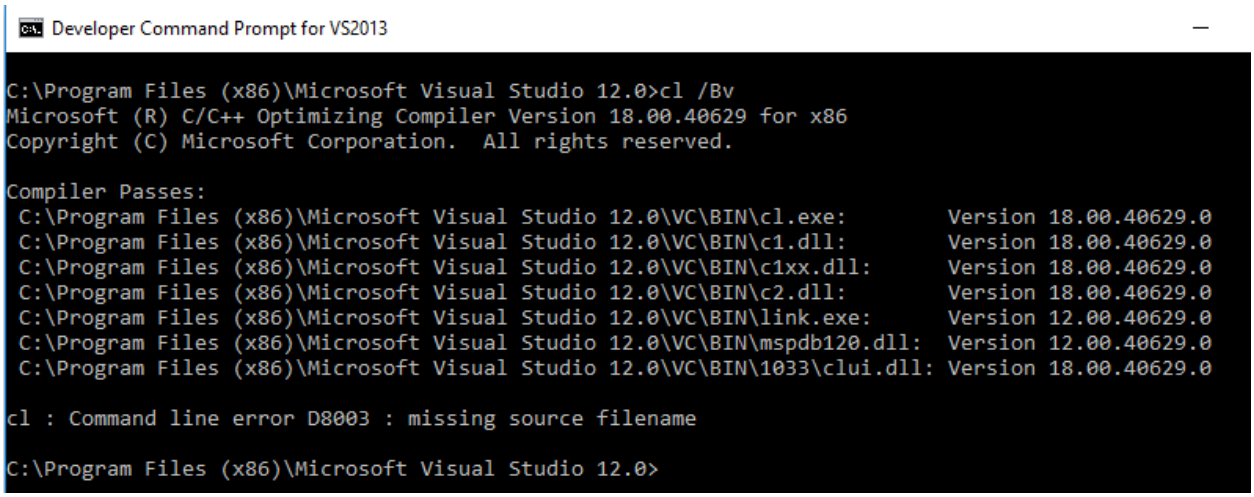
Display the Debug Disassembly Window (Ctrl+Alt+D) while stopped at the 2nd memcpy, in part memcpy_test_extfun.cpp to show:

```
            // Only 2 bytes should be copied and only 2 bytes are actually copied
            size_t        bytes_to_copy = 2;
                    memcpy(&target, char_array_local, bytes_to_copy);
00000054  lea         rdx,[rbp-10h]
00000058  lea         rcx,[rbp-18h]
0000005c  mov         r8d,2
00000062  movsxd      r8,r8d
00000065  call        000000005F5FED20

                    // Only 2 bytes should be copied but 8 bytes are actually copied
            memcpy(&target, char_array_local, 2);
0000006a  mov         rdx,qword ptr [rbp-10h]
0000006e  mov         qword ptr [rbp-18h],rdx
```

While the first memcpy was implemented by a call to which an argument of “2” has been passed, the second memcpy, the memcpy with the odd behavior, was implemented by moving a qword of data, which is 8 bytes from a register to the target storage address.

Full Version of C++ Compiler Being Used:



CL.command.1.tlog contents:

^C:\PROJECTS\CENCONDEVTRUNK\CENCONCOMMON\MEMCPY_TEST_EXTFUN.CPP
/c /AI"C:\Program Files (x86)\Reference Assemblies\Microsoft\Framework\.NETFramework\v4.0"
/AI"C:\Program Files (x86)\Windows Kits\8.1\References\CommonConfiguration\Neutral" /Zi /clr /W3
/WX- /sdl /O2 /Oi /GL /D _MBCS /EHa /MD /GS /Gy /fp:precise /Zc:wchar_t /Zc:forScope /FAs
/Fa"C:\PROJECTS\CENCONDEVTRUNK\X64\RELEASE\\" /Fo"X64\RELEASE\\"
/Fd"X64\RELEASE\VC120.PDB" /TP /FU"C:\Program Files (x86)\Reference
Assemblies\Microsoft\Framework\.NETFramework\v4.0\mscorlib.dll" /clr:nostdlib
C:\PROJECTS\CENCONDEVTRUNK\CENCONCOMMON\MEMCPY_TEST_EXTFUN.CPP

Verbose output from compilation showing compile command:

1>------ Rebuild All started: Project: CSImemcpy, Configuration: Release x64 ------
1> Microsoft (R) C/C++ Optimizing Compiler Version 18.00.40629
1> for Microsoft (R) .NET Framework version 4.07.2650.0
1> Copyright (C) Microsoft Corporation. All rights reserved.
1>
1> cl /c /AI"C:\Program Files (x86)\Reference Assemblies\Microsoft\Framework\.NETFramework\v4.0"
/AI"C:\Program Files (x86)\Windows Kits\8.1\References\CommonConfiguration\Neutral" /Zi /clr /W3
/WX- /sdl /O2 /Oi /GL /D _MBCS /EHa /MD /GS /Gy /fp:precise /Zc:wchar_t /Zc:forScope /FAs
/Fa"C:\Projects\CenconDevTrunk\x64\Release\\" /Fo"x64\Release\\" /Fd"x64\Release\vc120.pdb" /TP
/FU"C:\Program Files (x86)\Reference
Assemblies\Microsoft\Framework\.NETFramework\v4.0\mscorlib.dll" /errorReport:prompt /clr:nostdlib
..\CenconCommon\memcpy_test.cpp ..\CenconCommon\memcpy_test_extfun.cpp
1>
1> memcpy_test.cpp

```
1>  memcpy_test_extfun.cpp
1>  Generating Code...
1>  Microsoft (R) C/C++ Optimizing Compiler Version 18.00.40629
1>  for Microsoft (R) .NET Framework version 4.07.2650.0
1>  Copyright (C) Microsoft Corporation.  All rights reserved.
1>
1>  cl /c /Zi /clr /W3 /WX- /sdl /O2 /Oi /GL /D _MBCS /EHa /MD /GS /Gy /fp:precise /Zc:wchar_t
/Zc:forScope /FAs /Fa"C:\Projects\CenconDevTrunk\x64\Release\\" /Fo"x64\Release\\"
/Fd"x64\Release\vc120.pdb" /TP /errorReport:prompt
"C:\Users\vic.gettler\AppData\Local\Temp\.NETFramework,Version=v4.0.AssemblyAttributes.cpp"
1>
1>  .NETFramework,Version=v4.0.AssemblyAttributes.cpp
1>  CSImemcpy.vcxproj -> C:\Projects\CenconDevTrunk\x64\Release\CSImemcpy.exe
```

Compile options from compile listing from compile under Visual Studion 2013:

```
cl

/c
/AI"C:\Program Files (x86)\Reference Assemblies\Microsoft\Framework\.NETFramework\v4.0"
/AI"C:\Program Files (x86)\Windows Kits\8.1\References\CommonConfiguration\Neutral"
/Zi
/clr
/W3
/WX-
/sdl
/O2
/Oi
/GL
/D _MBCS
/EHa
/MD
/GS
/Gy
/fp:precise
/Zc:wchar_t
/Zc:forScope
/FAs
/Fa"C:\Projects\CenconDevTrunk\x64\Release\\"                    /Fo"x64\Release\\"
/Fd"x64\Release\vc120.pdb"
/TP
/FU"C:\Program                    Files                    (x86)\Reference
Assemblies\Microsoft\Framework\.NETFramework\v4.0\mscorlib.dll"
/errorReport:prompt
/clr:nostdlib
```

..\CenconCommon\memcpy_test.cpp
..\CenconCommon\memcpy_test_extfun.cpp

Source Parts:

memcpy_test.cpp

```
void ExtFun01( System::String^% my_str );

int main() {
	System::String^  my_str;
	ExtFun01( my_str );
	return 0;
}
```

memcpy_test_extfun.cpp

```
/*

Experimentation shows that

   •   Must be compiled with optimization turned on
   •   Compilation must be targeting a 64-bit platform
   •   Memcpy target address must be that of an arithmetic item
   •   Memcpy's specified number of bytes to copy must be a
        o literal
        o valid arithmetic length

*/

/*
Compiler command .... with options

 cl
   /c
   /AI"C:\Program Files (x86)\Reference
Assemblies\Microsoft\Framework\.NETFramework\v4.0"
   /AI"C:\Program Files (x86)\Windows Kits\8.1\References\CommonConfiguration\Neutral"
   /Zi
   /clr
   /W3
   /WX-
   /sdl
   /O2
   /Oi
   /GL
   /D _MBCS
   /EHa
   /MD
   /GS
   /Gy
   /fp:precise
   /Zc:wchar_t
   /Zc:forScope
   /FAs
   /Fa"C:\Projects\CenconDevTrunk\x64\Release\\" /Fo"x64\Release\\"
/Fd"x64\Release\vc120.pdb"
   /TP
   /FU"C:\Program Files (x86)\Reference
Assemblies\Microsoft\Framework\.NETFramework\v4.0\mscorlib.dll"
   /errorReport:prompt
   /clr:nostdlib
   ..\CenconCommon\memcpy_test.cpp
   ..\CenconCommon\memcpy_test_extfun.cpp


Assembler from Visual Studio Debug Disassembly Window:  Ctrl+Alt+D

   // Only 2 bytes should be copied and only 2 bytes are actually copied
   size_t        bytes_to_copy = 2;
   memcpy(&target, char_array_local, bytes_to_copy);
   00000054  lea         rdx,[rbp-10h]
```

```
   00000058  lea         rcx,[rbp-18h]
   0000005c  mov         r8d,2
   00000062  movsxd      r8,r8d
   00000065  call        000000005F5FED20

   // Only 2 bytes should be copied but 8 bytes are actually copied
   memcpy(&target, char_array_local, 2);
   0000006a  mov         rdx,qword ptr [rbp-10h]
   0000006e  mov         qword ptr [rbp-18h],rdx


Show memory before and after each memcpy

*/

#include <cstring>

void ExtFun01( System::String^%    my_str )
{
            unsigned char char_array_local[2] = {0xAB, 0xCD};
                    __int64       target = 0xFFFFFFFFFFFFFFFF;


            // Only 2 bytes should be copied and only 2 bytes are actually copied
            size_t        bytes_to_copy = 2;
                    memcpy(&target, char_array_local, bytes_to_copy);

                    // Only 2 bytes should be copied but 8 bytes are actually copied
            memcpy(&target, char_array_local, 2);


            // Use target variable to prevent memcpy from being optimized out
                    if (target > 55555)
                           my_str = "A";

            return;
}
```