Subject: Big issue with Visual Studio 2019 (some versions)
Posted by Alboni on Sat, 08 Feb 2020 01:19:40 GMT
View Forum Message <> Reply to Message

Hello,
There is a big issue with Visual Studio 2019 (some versions)
The example gives a warning and with reason
It may seem like a small issue but it'd not. This is used all over UPP and TheIde.
I can't compile a working non crashing version of TheIde (and my sofware) with it.
It's roulette whether it works or not. On my collegues computer it works, on mine it doesn't

See example below.

```
#include <Core/Core.h>

using namespace Upp;

// this construction is used a lot in upp
String SomeFunction()
{
 StringBuffer s;
 s.Cat("M\0nkey", 6); // some binary data
 return s;  // warning C4927: illegal conversion; more than one user-defined conversion has been implicitly applied

 // this contruction (returning a StringBuffer to a String return type) is used a LOT in UPP
 // it relies on the constructor of String(StringBuffer& )  to do the copying
 // However another path is from StringBuffer::Begin() -> String::String(const char*)
 // this however does not work for binary data (containing a 0)
 // Visual Studio 2019 picks the first or the second option depending on which *EXACT* version of it you're using, and gives this warning that it flipped a coin
 // IT IS NOT A BENIGN WARNING! It's roulette if the software works or not.
 // I cannot compile a stable executable of TheIde with VS2019 on my computer. It just keeps
 // crashing. My collegue can.
}

CONSOLE_APP_MAIN
{
 String rv = SomeFunction();
 Cout() << rv.GetCount() << "\r\n"; // Should print 6, but prints 1
}
```

File Attachments

1) StringBuffer.exe, downloaded 505 times
2) StringBuffer.zip, downloaded 454 times
3) error.txt, downloaded 423 times

Subject: Re: Big issue with Visual Studio 2019 (some versions)
Posted by Alboni on Sat, 08 Feb 2020 10:50:48 GMT
View Forum Message <> Reply to Message

If I make the following change:

```
//return s;
return String(s);
```

I get the correct results and the warning is gone.

I also tried changing it the same way in UPP in all 86 places (yes some were WString and there was a T) and that also works.

I can make a patch if that is appreciated.

Perhaps someone knows a more elegant solution?

Subject: Re: Big issue with Visual Studio 2019 (some versions)
Posted by mirek on Sat, 08 Feb 2020 10:55:32 GMT
View Forum Message <> Reply to Message

It would be worth to write here explicit version of visual studio that mibehaves.

If it is the most recent, we can file a bug if we can simplify the issue to nonU++. Last time the fix was fast.

As everything seems to be fine with gcc and clang, I do not think we should hurry for workaround here.

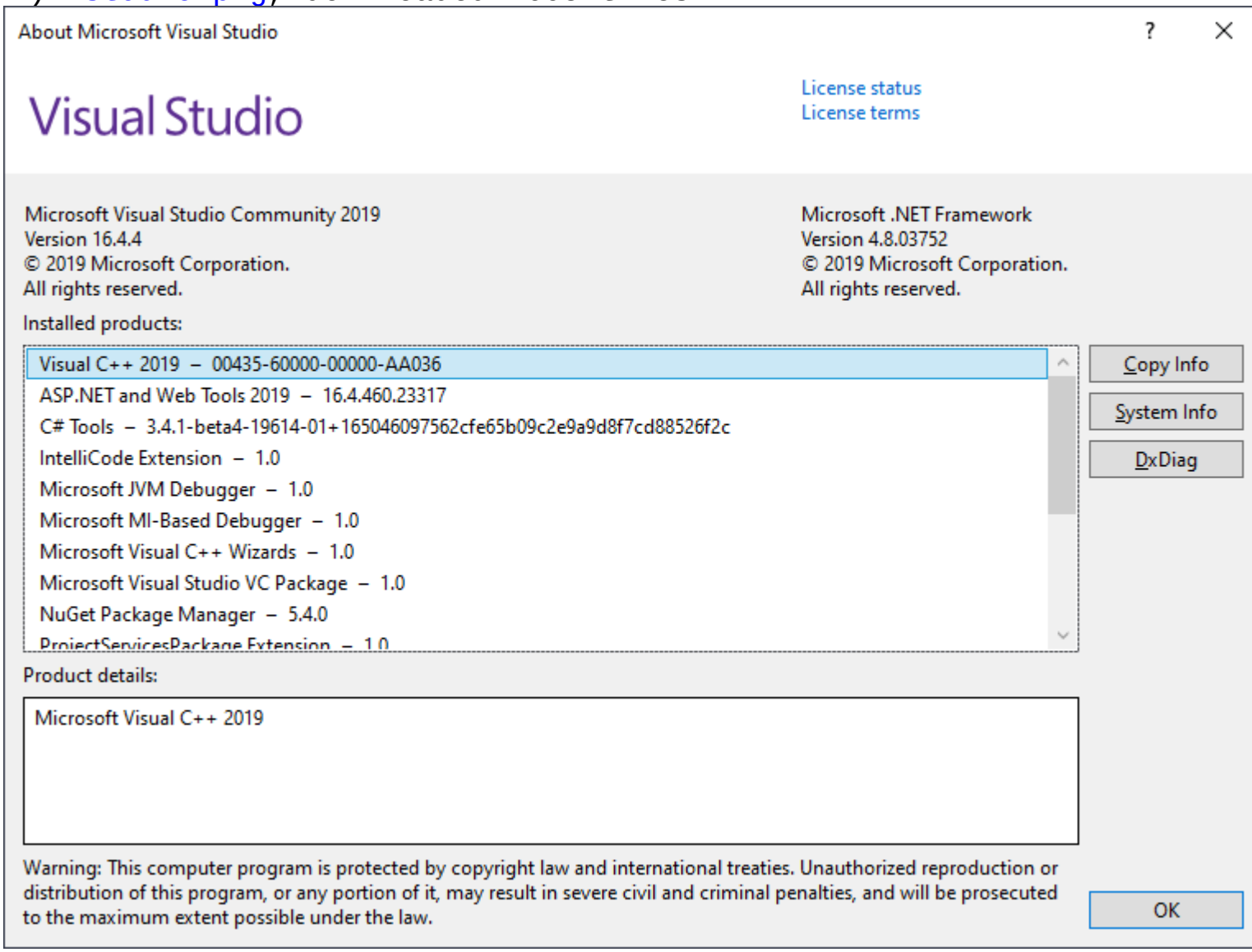
Mirek

Subject: Re: Big issue with Visual Studio 2019 (some versions)
Posted by Alboni on Sat, 08 Feb 2020 12:02:29 GMT
View Forum Message <> Reply to Message

It is the most recent. I downloaded it yesterday.
I'll look up the exact number in a minute

Subject: Re: Big issue with Visual Studio 2019 (some versions)
Posted by Alboni on Sat, 08 Feb 2020 12:10:32 GMT
View Forum Message <> Reply to Message

File Attachments
1) vstudio.png, downloaded 1605 times



Subject: Re: Big issue with Visual Studio 2019 (some versions)
Posted by Alboni on Sat, 08 Feb 2020 12:17:05 GMT
View Forum Message <> Reply to Message

My collegue's computer (on which it works without workaround) does still emit the warning....

Subject: Re: Big issue with Visual Studio 2019 (some versions)
Posted by mirek on Sun, 09 Feb 2020 09:34:11 GMT
View Forum Message <> Reply to Message

OK, now I have to admit that I am reluctant to break my environment by downloading possibly broken compiler....

So thinks that come to mind:

- first of all, I believe that this is compiler bug, not ours, because operator char * is two conversions far, while StringBuffer is direct parameter. Do you agree?

- are you able to create some working single file testcase for microsoft?

- as much as I hate the idea, we can perhaps try workaround by removing String(StringBuffer&) constructor and adding operator String to StringBuffer.

Mirek

Subject: Re: Big issue with Visual Studio 2019 (some versions)
Posted by Alboni on Sun, 09 Feb 2020 13:33:36 GMT
View Forum Message <> Reply to Message

Yeah, don't break your computer.

I came up with this:

```
#include <iostream>

typedef unsigned char bYte;
#define min(a,b) ( a<b ? a : b )

class B;

class A {
 public:
  bYte data[256]; bYte datalen;

 public:
  A(const void* b, bYte len); // assign binary data
  A(char* string); // assign C string
  A(B& b);
  operator char*() // to c string
  {
   data[sizeof(data)-1]=0;
   return (char*)data;
```

```
  };
};

class B {
 public:
  bYte data[256]; bYte datalen;

 public:
  B(const void* b, bYte len); // assign binary data
  B(char* string); // assign C string
  operator char* ()  // to c string
  {
   data[sizeof(data)-1]=0;
   return (char*)data;
  }
};


A::A(const void* b, bYte len) // assign binary data
{
 datalen = min(len, sizeof(data)-1);
 memcpy(data, b, datalen);
 data[datalen]=0;
}

A::A(char* string) // assign C string
{
 datalen = min(strlen(string)+1, sizeof(data));
 memcpy(data, string, datalen);
 data[sizeof(data)-1]=0;
}

A::A(B& b)
{
 datalen = b.datalen;
 memcpy(data, b.data, sizeof(data));
}

B::B(const void* b, bYte len) // assign binary data
{
 datalen = min(len, sizeof(data)-1);
 memcpy(data, b, datalen);
 data[datalen]=0;
}

B::B(char* string) // assign C string
{
 datalen = min(strlen(string)+1, sizeof(data));
```

```
 memcpy(data, string, datalen);
 data[sizeof(data)-1]=0;
}

A ReturnB()
{
 const char binary[10] = {1,2,0,3,4,5,6,7,8,0};
 B b(binary, 10);
 return b;
}

int main(int argc, const char *argv[])
{
 A a = ReturnB();
 int len = a.datalen;
 std::cout << "This should return 10 and it returns " << len << "\r\n";
 return 0;
}
```

File Attachments

1) msvc10bug.exe, downloaded 456 times
2) msvc10bug.cpp, downloaded 407 times

Subject: Re: Big issue with Visual Studio 2019 (some versions)
Posted by Alboni on Sun, 09 Feb 2020 13:35:17 GMT
View Forum Message <> Reply to Message

An interesting thing happens:

if I make the following change:

```
//   A(B& b);
  A(const B& b);
```

Then the warning goes away and I get the correct results.....

That also works for String::String(StringBuffer&) but that one can't be const of course.

Subject: Re: Big issue with Visual Studio 2019 (some versions)
Posted by mirek on Mon, 10 Feb 2020 09:32:58 GMT
View Forum Message <> Reply to Message

Sorry, but way too long. Have a pity for those poor compiler maintainers...

Does this show the problem?

```
#include <iostream>

struct B;

struct A {
 A(char*) { std::cout << "A::A(char *)\n"; }
 A(B&)    { std::cout << "A::A(B&)\n"; }
};

struct B {
 operator char*() { std::cout << "B::operator char *()\n"; }
};

A Test()
{
 B b;
 return b;
}

int main(int argc, const char *argv[])
{
 A a = Test();
 return 0;
}
```

Subject: Re: Big issue with Visual Studio 2019 (some versions)
Posted by Alboni on Mon, 10 Feb 2020 09:58:02 GMT
View Forum Message <> Reply to Message

Yes, it does. Excellent pruning But it needs to return a value to compile:

```
// operator char*() { std::cout << "B::operator char *()\n"; }
 operator char*() { std::cout << "B::operator char *()\n";  return ""; }
```

Subject: Re: Big issue with Visual Studio 2019 (some versions)
Posted by mirek on Mon, 10 Feb 2020 10:50:07 GMT

View Forum Message <> Reply to Message

https://developercommunity.visualstudio.com/content/problem/912723/recent-version-of-visual-c-compiler-might-have-ove.ht ml

Subject: Re: Big issue with Visual Studio 2019 (some versions)
Posted by gocubsgo on Tue, 28 Jul 2020 16:12:22 GMT
View Forum Message <> Reply to Message

Is there any update on this? It seems like Microsoft never did respond to your feedback?

With the latest Visual Studio 2019 (16.6.5) I'm still getting many C4927 warnings. The result in my app is the UI (all defaults) is missing elements, like some assets aren't being loaded. Forcing to the older Platform Toolset (v141) when building upp results in things looking as they should.

By the way, this is using the 2020.1 release. Looking at the latest nightly, I don't see anything suggesting it would be different in this respect.

Subject: Re: Big issue with Visual Studio 2019 (some versions)
Posted by mirek on Wed, 29 Jul 2020 08:32:36 GMT
View Forum Message <> Reply to Message

gocubsgo wrote on Tue, 28 July 2020 18:12Is there any update on this? It seems like Microsoft never did respond to your feedback?

With the latest Visual Studio 2019 (16.6.5) I'm still getting many C4927 warnings. The result in my app is the UI (all defaults) is missing elements, like some assets aren't being loaded. Forcing to the older Platform Toolset (v141) when building upp results in things looking as they should.

By the way, this is using the 2020.1 release. Looking at the latest nightly, I don't see anything suggesting it would be different in this respect.

No update. This is clearly Visual C++ bug and current compiler is broken.

Fortunately Windons CLANG is now good enough as replacement (which works out of box).

Mirek

Subject: Re: Big issue with Visual Studio 2019 (some versions)
Posted by Alboni on Wed, 29 Jul 2020 12:17:30 GMT
View Forum Message <> Reply to Message

I can't use clang or mingw on windows because it misses certain libraries that I need like winscard

and plugin/wav also doesn't compile on either.
However I find applying the workaround* relatively easy and I don't upgrade all that often anyway, so I'll see when it happens and if not, no big deal. Just have to be mindful of that compiler quirk.

* workaround is putting return String(s); instead of return s; everywhere the warning throws.

Subject: Re: Big issue with Visual Studio 2019 (some versions)
Posted by Alboni on Wed, 29 Jul 2020 12:27:14 GMT
View Forum Message <> Reply to Message

It would however be handy if this patch could be made:
cppbuilder.cpp line 469

before:

```
 if(package == mainpackage)
  info << Join(SvnInfo(package), "\r\n");
```

after patch

```
 if(package == mainpackage)
  info << Join(SvnInfo(package), "\r\n") << "\r\n";
```

This is because I am including "build_info.h" in the .rc file and the microsoft resource compiler throws an error if the last line of an included file does not end on a \r\n. Yeah, also their fault but it's an ancient bug that will likely never be fixed.
It would avoid me having to recompile theide on a broken system.

Subject: Re: Big issue with Visual Studio 2019 (some versions)
Posted by mirek on Wed, 23 Sep 2020 10:45:06 GMT
View Forum Message <> Reply to Message

Alboni wrote on Wed, 29 July 2020 14:17I can't use clang or mingw on windows because it misses certain libraries that I need like winscard and plugin/wav also doesn't compile on either.
However I find applying the workaround* relatively easy and I don't upgrade all that often anyway, so I'll see when it happens and if not, no big deal. Just have to be mindful of that compiler quirk.

* workaround is putting return String(s); instead of return s; everywhere the warning throws.

In the I decided to go with workaround, so trunk should now work fine out of box...

Subject: Re: Big issue with Visual Studio 2019 (some versions)
Posted by JeyCi on Thu, 20 Mar 2025 05:58:22 GMT
View Forum Message <> Reply to Message

gocubsgo wrote on Tue, 28 July 2020 18:12
With the latest Visual Studio 2019 (16.6.5) I'm still getting many C4927 warnings.
the same problem now/today - e.g. used for this code

mirek wrote on Wed, 29 July 2020 10:32
No update. This is clearly Visual C++ bug and current compiler is broken.
Mirek

Subject: Re: Big issue with Visual Studio 2019 (some versions)
Posted by Alboni on Thu, 20 Mar 2025 12:38:58 GMT
View Forum Message <> Reply to Message

Yes, all versions give the warning, but not all of them make the wrong choice as to which execution path to follow.

Subject: Re: Big issue with Visual Studio 2019 (some versions)
Posted by mirek on Thu, 20 Mar 2025 14:37:15 GMT
View Forum Message <> Reply to Message

Alboni wrote on Thu, 20 March 2025 13:38Yes, all versions give the warning, but not all of them make the wrong choice as to which execution path to follow.

Uh, I am suprised this has returned. MSC 2022 and current U++ -> no warnings...

Are you testing this?

Subject: Re: Big issue with Visual Studio 2019 (some versions)
Posted by Alboni on Thu, 20 Mar 2025 20:12:06 GMT
View Forum Message <> Reply to Message

No, I haven't tested anything with VS22 and later. I spoke from memory. Sorry for the confusion.

Subject: Re: Big issue with Visual Studio 2019 (some versions)
Posted by JeyCi on Sun, 01 Jun 2025 17:19:24 GMT
View Forum Message <> Reply to Message

mirek wrote on Thu, 20 March 2025 15:37
Uh, I am suprised this has returned. MSC 2022 and current U++ -> no warnings...

I could not even suppose how and why, but after some experiments, having created new project with VSBT2019, - it started to compile OK for about a month -- just build_method file should be correct, perhaps... but after reinstalling VSBT2019 - were 2 unresolved functions... GetTempPathW function (fileapi.h) and GetModuleFileNameW function (libloaderapi.h) - both headers from in C:\Program Files\Windows Kits\10\Include\10.0.19041.0\um... OK, they reference windef.h that is in C:\Program Files\Windows Kits\10\Include\10.0.19041.0\shared, include together with um & ucrt. And for c-runtime to PATH C:\Program Files\Microsoft Visual Studio\2019\BuildTools\VC\Redist\MSVC\v142 and bin, include, lib from ..\VC\Tools\..

P.S. C:\Program Files\Microsoft SDKs\Windows Kits - no more used - C4927 warning appears with C:\Program Files\Microsoft SDKs\Windows Kits\10\ExtensionSDKs\Microsoft.UniversalCRT.Debug\10.0.1904 1.0\Redist\Debug, C:\Program Files\Windows Kits\10 - use instead

---