Subject: AssertST problem in initialization when debugging on MSBT17x64
Posted by Tom1 on Tue, 27 Nov 2018 13:59:05 GMT
View Forum Message <> Reply to Message

Hi,

As the title states, something goes wrong here when starting a debugging session of a GUI (MT) application:
#ifdef _DEBUG
inline void AssertST() { ASSERT(Thread::IsST()); }
#endif

The ASSERTion happens. If I comment out the ASSERT() code, the debugging session starts just fine. I'm running on Windows 10 Professional 64-bit and using MSBT17x64 compiler with up-to-date U++. Also, Release code works OK, which is obvious since the above code is not active there. Is there something wrong with my setup or is there some issue in Core?

Best regards,

Tom

Subject: Re: AssertST problem in initialization when debugging on MSBT17x64
Posted by mr_ped on Tue, 27 Nov 2018 17:44:29 GMT
View Forum Message <> Reply to Message

From the name of it I guess it's asserting that you are in single-thread mode, but you are not ("GUI (MT)").

So it depends, where this assert is, if you did it write yourself, then why? If it's part of UPP sources, then that part is either not meant to be used in MT, or the flag "MT" was not delegated properly during compilation and the ST variant of code got picked up, or there's a bug.

Unfortunately you didn't indicate where that AssertST is used, so hard to say where it is and why.

(edit: if it is directly in the IDE code (not in app code) in the start-debug-session, then the author of that code has to explain, if it's really supposed to be used only for ST debugging, or this is some obsolete artefact and can be removed now.)

Subject: Re: AssertST problem in initialization when debugging on MSBT17x64
Posted by Tom1 on Wed, 28 Nov 2018 08:23:36 GMT
View Forum Message <> Reply to Message

Hi Peter,

Sorry for being so unspecific.

First, it is my own program where this happens when running in debugger. Here's the backtrace:

DebugBreak()
Upp::AssertFailed(file=7ff6022e6700 "c:\\upp-12514\\upp.src\\uppsrc\\core\\Mt.h", line=116 't', cond=7ff6022e66e8 "Thread::IsST()") at c:\upp-12514\upp.src\uppsrc\core\util.cpp 167
Upp::AssertST() at c:\upp-12514\upp.src\uppsrc\core\mt.h 116
Upp::Value::Register(w=40 '(', c=7ff600fa7ea5, name=7ff6022d46ac "Font") at c:\upp-12514\upp.src\uppsrc\core\value.cpp 560
Upp::Value::Register<Upp::Font>(name=7ff6022d46ac "Font") at c:\upp-12514\upp.src\uppsrc\core\value.hpp 318
Upp::s__sF0_19_fn() at c:\upp-12514\upp.src\uppsrc\draw\font.cpp 21
Upp::Callinit::Callinit(fn=7ff6013188f0, cpp=7ff6022d4618 "C:\\upp-12514\\upp.src\\uppsrc\\Draw\\Font.cpp", line=19) at c:\upp-12514\upp.src\uppsrc\core\defs.h 176
Upp::`dynamic initializer for 's__sF0_19''() at c:\upp-12514\upp.src\uppsrc\draw\font.cpp 19
_initterm(first=7ff60225d000->0, last=7ff60225e180->0)
__scrt_common_main_seh()
__scrt_common_main()
WinMainCRTStartup()
BaseThreadInitThunk()
RtlUserThreadStart()

After digging a bit deeper, I found out that my own code contains a static instance of a class that starts a couple of threads in the background. These threads are already running when the above assertion happens and if I do not start those threads, the assertion does not happen. However, I do need those threads for my software to work properly.

I worked around this issue by switching from "static MyClass a;" to "One<MyClass> a;" and appropriate initialization code "a.Create();" that runs only in GUI_APP_MAIN after everything else is initialized within U++.

Thanks and best regards,

Tom

Subject: Re: AssertST problem in initialization when debugging on MSBT17x64
Posted by mr_ped on Wed, 28 Nov 2018 09:34:03 GMT
View Forum Message <> Reply to Message

I believe your fix is more "robust", and will bring you less ugly surprises in case the static class would try to use something from U++ Core... but if it was some simple stand-alone class, then this

is a bit like extra-price for using the U++.

So I think in the end everything worked as supposed, and you fixed a bug in your code (although it may be just "U++ code style" "bug", not real bug causing some wrong results at runtime).

Maybe that assert in mt.h can have some comment associated, that it may happen in such specific case, so one wouldn't have to dig that deep to figure out what is the cause.

Subject: Re: AssertST problem in initialization when debugging on MSBT17x64
Posted by Tom1 on Wed, 28 Nov 2018 12:32:20 GMT
View Forum Message <> Reply to Message

mr_ped wrote on Wed, 28 November 2018 11:34Maybe that assert in mt.h can have some comment associated, that it may happen in such specific case, so one wouldn't have to dig that deep to figure out what is the cause.

That might save some time from others. Anyway, I guess I learned this already! :lol:

Best regards,

Tom

Subject: Re: AssertST problem in initialization when debugging on MSBT17x64
Posted by peterh on Wed, 28 Nov 2018 13:58:27 GMT
View Forum Message <> Reply to Message

The documentation is (not always, but often) in the source:

If you click the small green square left from AsserST(), you see this:

void AssertST()
This operation only has effect in DEBUG mode. If any Thread was started prior to calling AssertST, it will stop the execution with diagnostic message. The purpose is that some global initialization routines are best performed before any multi-threading starts. AssertST can be used to assure this as runtime check.

Subject: Re: AssertST problem in initialization when debugging on MSBT17x64
Posted by mirek on Wed, 28 Nov 2018 17:22:47 GMT
View Forum Message <> Reply to Message

mr_ped wrote on Wed, 28 November 2018 10:34
So I think in the end everything worked as supposed, and you fixed a bug in your code (although it may be just "U++ code style" "bug", not real bug causing some wrong results at runtime).

Well, it is all about optimization. Global initialization often creates various maps, e.g. map of image decoders. If we can say that these happen before any threads start, we can access these maps without mutexes. So the general rule is established as "never start threads before main" and AssertST is here to enforce it.

Mirek
