

---

Subject: Assertion in Debug.cpp:246 "nesting\_depth == 0" [BUGS?]

Posted by [hojtsy](#) on Wed, 22 Mar 2006 15:25:55 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

I have an assertion failure in my application while quitting. Assertion is triggered in this method of Core/Debug.cpp:

```
TimingInspector::~TimingInspector() {  
    ASSERT(nesting_depth == 0);
```

I have running timers and threads when I quit. Could this be the reason for this assertion? I am using 603-dev2 on Windows.

---

---

Subject: Re: Assertion in Debug.cpp:246 "nesting\_depth == 0"

Posted by [mirek](#) on Wed, 22 Mar 2006 16:38:41 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

hojtsy wrote on Wed, 22 March 2006 10:25: I have an assertion failure in my application while quitting. Assertion is triggered in this method of Core/Debug.cpp:

```
TimingInspector::~TimingInspector() {  
    ASSERT(nesting_depth == 0);
```

I have running timers and threads when I quit. Could this be the reason for this assertion? I am using 603-dev2 on Windows.

Very likely - if you have threads running at exit. Actually, at the moment, it seems to me like bad practice - IMO code process should start and end with single thread...

Unfortunately, it is also quite possible that this fails with MT app even if ended as one thread - TimingInspector has very specific use and is not MT safe (and likely will never be).

The real question is: are you using TIMING or RTIMING macros somewhere or they are just forgotten in U++ library code?

Mirek

---

---

Subject: Re: Assertion in Debug.cpp:246 "nesting\_depth == 0"

Posted by [hojtsy](#) on Wed, 22 Mar 2006 18:24:42 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

I have threads running on exit because those threads are waiting for very slow HTTP servers to respond (inside library method). Since U++ does not provide thread termination, and I can not notify the library function to stop waiting, I would have to wait for the download attempts to time out, which could be 1 minute. It is not acceptable to wait 1 minute for application termination.

I am not using TIMING or RTIMING in my application, so they should be in the library.

---

---

Subject: Re: Assertion in Debug.cpp:246 "nesting\_depth == 0"

Posted by [mirek](#) on Wed, 22 Mar 2006 19:37:11 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

hojtsy wrote on Wed, 22 March 2006 13:24I have threads running on exit because those threads are waiting for very slow HTTP servers to respond (inside library method). Since U++ does not provide thread termination, and I can not notify the library function to stop waiting, I would have to wait for the download attempts to time out, which could be 1 minute. It is not acceptable to wait 1 minute for application termination.

Actually, this is quite interesting topic to resolve:

- first, it is IMO impossible to correctly terminate the thread the "hard" way - in C++, many resources would be left allocated. What would indeed be needed is some form of "forced thread exception"... (is that possible)?

- lefts us with "soft" way - somehow signalling to the thread that it should terminate. In "GuiMT" example, I am doing that using active checking of "terminated" state, but that of course is not a very good way...

Any suggestions in this area are welcome!

Quote:

I am not using TIMING or RTIMING in my application, so they should be in the library.

OK, I will run "find in files"

Mirek

---

---

Subject: Re: Assertion in Debug.cpp:246 "nesting\_depth == 0"

Posted by [mirek](#) on Wed, 22 Mar 2006 19:39:56 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

Found one forgotten RTIMING in XML code. Were you using XML?

Mirek

---

---

Subject: Re: Assertion in Debug.cpp:246 "nesting\_depth == 0"

Posted by [hojtsy](#) on Wed, 22 Mar 2006 23:14:12 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

I was using XML.

The library function that is running in the threads during exit is the `HttpClientGet`. I see no other clean way for using the `HttpClient` than creating own thread for the communication, so that GUI remains responsive.

I guess I could create and pass a progress callback which stops these functions when exit is requested. Then I would need to count active threads somehow: the best solution would be to manage the thread count in the `sThreadRoutine` library function, instead of attempting to implement this in multiple client apps. Also the `Core` package could provide a `WaitUntilThreadsExit` function, which would wait on the semaphor of this thread count. Than I would somehow invoke the `WaitUntilThreadsExit` function when exit is requested. These desired library features seems to be usefull for practically every MT app.

---