Subject: Core multithread dangers
Posted by hojtsy on Fri, 03 Feb 2006 22:09:17 GMT
View Forum Message <> Reply to Message

I was told before that Core & Web packages are thread-safe. But some methods of Core are not thread-safe, and something like
CriticalSection::LockMain should be added to them to fix this. Here is a list of them.
static TimingInspector& s_zero()
TimingInspector::TimingInspector(const char *_name)
static void sInit()
String& sHomeDir()
String ConfigFile(const char *file)
byte *Alloc4KBRaw()
void *MemoryAllocPermanent(size_t size)
void *MemoryAlloc(size_t size)
void MemoryProbe(const char *name, dword flags)
const LanguageInfo& GetLanguageInfo(int lcode)
static Array<LngModule>& sMod()
CriticalSection& MainCriticalSection()
PageInfo *NewPage(int magnitude)
static CriticalSection& sHeapLock()
static CriticalSection& sHeapLock2()
static inline void sInitTables()
void MemoryInitDiagnostics()
RHITCOUNT(n)
static int sMappingGranularity_()
VectorMap<String, VectorMap<String, VectorMap<String, Topic > > >& TopicBase()
String GetIniKey(const char *name)

These other functions of Core need bigger rewrite to be threadsafe:sDumpPtr(void *ptr)
sDump(dword w)
const char *Dump(PageInfo *page)
const char *MemoryCounters()

These functions of the Web package are not thread-safe:
void HttpServer::ReadPostData(Socket& socket, HttpQuery& query)
void SSLInit()
static const dword *GetCRCTable()
RefPtr<HttpQuery::Data> HttpQuery::Empty()
dword WINAPI HttpExtensionProc(EXTENSION_CONTROL_BLOCK *ecb)

The problem is always with the local static variables which are not protected from parallel initialization on two threads.

Subject: Re: Core multithread dangers
Posted by mirek on Fri, 03 Feb 2006 22:51:23 GMT

View Forum Message <> Reply to Message

You are right in some cases, but look better. Many of those are in fact serialized elsewhere (that is esp. true for heap - that one IS thread safe to my knowledge).

OTOH, you have missed some that might cause a trouble in future (e.g. when working on GuiMT, I have fount that Ptr/Pte are not really thread safe - something to fix ASAP).

Also, note that StaticCriticalSection is zero-initilized POD, so it does not perform any further form of initilization (nice simple solution to "how to serialize initialization of serializer").

Anyway, thanks. I must admit that multithreading is generally less tested / fixed than the rest of library. Any warnings/suggestions here are welcome.

Mirek

Subject: Re: Core multithread dangers
Posted by mirek on Sat, 04 Feb 2006 11:14:14 GMT
View Forum Message <> Reply to Message

Well, thining about issue and going through the Core...

I have found many places where MT is broken - most of them will not affect normal apps, but anyway...

I already have MT on ToDo list, but it turns out that careful audit of sources will be needed. I hope to get it done while working on new Draw.

I am also considering another MT option for GUI - perhaps it is not that bad idea to provide single GUI critical section that would got locked while processing input events - other threads of MT GUI app would simply lock this section before calling methods of any widget.

Mirek

Subject: Re: Core multithread dangers
Posted by hojtsy on Sat, 04 Feb 2006 13:53:15 GMT
View Forum Message <> Reply to Message

luzr wrote on Fri, 03 February 2006 17:51Also, note that StaticCriticalSection is zero-initilized POD, so it does not perform any further form of initilization (nice simple solution to "how to serialize initialization of serializer").Yes, that is a very good idea. But I needed almost an hour to figure out how that works before you posted it - it was quite tricky. Even a short code comment would have saved me lots of time.

I think this new macro is not threadsafe:#define ONCELOCK \

```
for(static volatile bool b = true; b;) \
 for(static CriticalSection section; b;) \
  for(CriticalSection::Lock lock(section); b; b = false)
```
The problem is that the b flag is not protected from repeated default-initialization to true.

Subject: Re: Core multithread dangers
Posted by mirek on Sat, 04 Feb 2006 17:20:13 GMT
View Forum Message <> Reply to Message

Yes, my fault, thanks. Fixed.

Mirek

Subject: Re: Core multithread dangers
Posted by mirek on Mon, 06 Feb 2006 10:46:45 GMT
View Forum Message <> Reply to Message

Hojtsy, I have one MT trouble I would like to share....

It is about Ptr implementation. While current version is MT safe to my knowledge, it is inferior to previous one as it allocates shared data for the lifetime of Pte target.

Original version was able to delete shared data whenever no more pointer were pointing to it, something like

```
release prec:
if(--prec->n == 0) {
   // MT!
   prec->ptr->prec = NULL;
   delete prec;
}
```

However, I do not see a way how to implement it without adding a lock to Pte, which is even worse. The trouble is that at the MT! point, Pte can be destructed and prec is not valid anymore....

(Just for record, ~Pte() { if(prec) prec->ptr = NULL; })

Hm, thinking about it, maybe single _global_ lock would do?

Mirek

Subject: Re: Core multithread dangers
Posted by hojtsy on Mon, 06 Feb 2006 13:18:07 GMT
View Forum Message <> Reply to Message

Mirek, I did not fully understand your post. I will try to sync and see the modified implementation of Ptr, but I only have uvs set up on my home computer, so it will take some time. But my question is: how would a global lock be better then a per-object lock? Are you trying to optimize the memory consumption of the lock objects?

Subject: Re: Core multithread dangers
Posted by mirek on Mon, 06 Feb 2006 13:24:17 GMT
View Forum Message <> Reply to Message

Yes, optimize memory. Per-object lock would add enourmous data to Pte (more that would be saved by avoiding dynamic storage).

Then, "new/delete" itself have global lock and it would be called per each Pte construction/destruction without Pte global lock, so global lock for Pte does not add too much overhead IMHO. And locked areas are extremly slow (a couple of assembly instructions), which makes me believe that concurrent access will not happen very often (it is much likely to happen during new/delete).

I have tried to implement global lock version - now there are both in Core, former version commented out. Sync uvs2.

Mirek

Subject: Re: Core multithread dangers
Posted by hojtsy on Mon, 06 Feb 2006 13:37:31 GMT
View Forum Message <> Reply to Message

OK, I will check it. In the meanwhile let me indicate a bug, which may have been corrected already. I am trying to compile an upp multithreaded application on Linux. TheIde binary is from the 511 release, the library sources are from the latest 20060129 snapshot. Single threaded appliactions are compiling and running OK. But when I compile anything with the MT flag (including single threaded applications, such as the Bombs example) the application just freezes while starting. With some debugging we discovered that it seems to be waiting forever for some mutex. Is it possible that recursive locking happens for the same lock?

Subject: Re: Core multithread dangers
Posted by mirek on Mon, 06 Feb 2006 13:52:16 GMT
View Forum Message <> Reply to Message

Definitely.

Frankly, you are perhaps the first one trying to do MT in Linux

My guess is that it is either the issue of Single or INIT_LOCK.

Mirek

Subject: Re: Core multithread dangers
Posted by mirek on Mon, 06 Feb 2006 18:19:27 GMT
View Forum Message <> Reply to Message

Well, indeed. Some that I missed in my education: linux mutext is not reentrant.

I have spent nice productive day fixing this issue. At least GuiMT example now seems to work.

Mirek

Subject: Re: Core multithread dangers
Posted by hojtsy on Tue, 07 Feb 2006 08:59:53 GMT
View Forum Message <> Reply to Message

luzr wrote on Mon, 06 February 2006 13:19I have spent nice productive day fixing this issue. At least GuiMT example now seems to work.You mean it works on Linux? Because previously it didn't even work on windows for me - but that is fixed now.

Subject: Re: Core multithread dangers
Posted by fudadmin on Tue, 07 Feb 2006 10:09:13 GMT
View Forum Message <> Reply to Message

But I've got this error:

Linking...
LINK : fatal error LNK1104: cannot open file 'LIBCMTD.lib'

this article says something about missing LIB variables for Visual Studio .NET
http://www.celoxica.com/support/view_article.asp?ArticleID=5 27

As I remember, when installing Ultimate++ said: "don't need to setup environment variables for MS toolkit".
Any quick fix?

Subject: Re: Core multithread dangers
Posted by unodgs on Tue, 07 Feb 2006 10:41:17 GMT
View Forum Message <> Reply to Message

I'm not sure but in VisualC++ toolkit mulithreaded libs are missing... (try to get full visual studio..)

Subject: Re: Core multithread dangers
Posted by fudadmin on Tue, 07 Feb 2006 11:49:18 GMT
View Forum Message <> Reply to Message

unodgs wrote on Tue, 07 February 2006 05:41I'm not sure but in VisualC++ toolkit mulithreaded libs are missing... (try to get full visual studio..)

But I've got that library...

Subject: Re: Core multithread dangers
Posted by mirek on Tue, 07 Feb 2006 13:42:05 GMT
View Forum Message <> Reply to Message

The simple way is to download Visual Studio Express 2005. It is free (till the end of year I guess). And you even do not need to install the studio, just commandline compiler

Mirek