Subject: Re: Sharing and Locking
Posted by gridem on Tue, 16 Mar 2010 07:04:37 GMT
View Forum Message <> Reply to Message

luzr wrote on Tue, 16 March 2010 08:02
I would stop right there and asked "why?" I would prefer using the data instead of pointer.

The answer is simple: U++ already uses the same idiom . See for example:

CtrlCore.h:

```
static  Ptr<Ctrl> focusCtrl;
static  Ptr<Ctrl> focusCtrlWnd;
static  Ptr<Ctrl> lastActiveWnd;
static  Ptr<Ctrl> caretCtrl;
```

luzr wrote on Tue, 16 March 2010 08:02
However, IF I would insist on using the pointer, then the pointer itself is shared resource and you need a lock while accessing it. No matter if it is raw pointer or Pte/Pte.

Yes, you are completely right.

luzr wrote on Tue, 16 March 2010 08:02
However, I agree that the existence of weak_ptr::lock is an advantage in some cases, but

boost manual
Even if p.reset() is executed in another thread, the object will stay alive until r goes out of scope or is reset. By obtaining a shared_ptr to the object, we have effectively locked it against destruction.

scares my insticts to the death - this is exactly the case I was speaking about - you are still accessing zombie object that is not supposed to exist anymore.

No, the considered situation is a bit more complicated. Because I used not shared_ptr for global variable but weak_ptr, the object will live until it will be destroyed in thread 1. But if I was successfull on converting from weak_ptr to shared_ptr, than the object lifetime will be longer and will be destroyed when loop in thread 1 and thread 2 will be restarted. In any case the object will not be in partial (or zombie) state when it will be destoyed in destructor instead of some method like Close, Destroy or other.
luzr wrote on Tue, 16 March 2010 08:02
Well, obviously, the code is missing serialization of DataAccess...

Yes. I don't serialize because my primary goal was to show the race in usage pattern if(data) data->... But of course, accurate solution must have two locks: global and internal.
luzr wrote on Tue, 16 March 2010 08:02
Now, perhaps we should try hard to add some sort of "Lock" to Ptr and make it wholy atomic, if that is possible. But I do not think that the impact in real world apps would be worth of it.

Mirek
I think that for GUI application and GUI controls like Ctrl it's not necessary because it serialize access to it using global locks. It also serializes when constructions like PostCallBack are used. If I use the main thread to manipulate the data and to destroy it, then there is no any problems. The problems may occurs when I want to create the real MT application without GUI and try to access to global variables or global list of variables through Ptr.