Subject: PROPOSAL: Monitor mutex in objects
Posted by kohait00 on Thu, 15 Jul 2010 09:42:20 GMT
View Forum Message <> Reply to Message

hi guys,

in C# there exists a nice feature,

```
lock(objectinstance)
{
//method code on now protected object
}
```

one can protect another object of beeing somehow accessed / modified, while oneself is treating with it. the object itself doesnt know about that and does not need to take care on providing any synchonisation features inside.

would it be possible to make something like that in upp as well?
as far as i know, c# compiler helps here somehow..

meanwhile:

here is another maybe usefull construction, came in mind while i was dealing with Mt.h..

```
#define LOCKER UPP::Mutex __locker
#define LOCKED INTERLOCKED_(__locker)

class MyClass
{
 void Method()
 {
  LOCKED
  {
  //your per object locked code
  }
 }
};
private:
 LOCKER;
}
```

it's unlicke the pure INTERLOCKED, which would generate a StaticMutex in every method the LOCKED is used.

if it is sensefull...

Subject: Re: PROPOSAL: Monitor mutex in objects
Posted by kohait00 on Thu, 15 Jul 2010 12:11:44 GMT
View Forum Message <> Reply to Message

here comes package on how i thought of it...
short description:

first section uses 2 Threads, operating on same String, using INTERLOCKED, which doesnt work, because it creates internal static Mutex,

second secton uses a helper construction, LOCKED, which uses a LOCKER placed Mutex instance in class to protect things, this could be done explicitly by placing a Mutex in class, and using Mutex::Lock(instance) scoped helper as well.

third section uses a Single() Mutex map, to protect arbitrary objects from 'outside' as i understood it from C#

maybe its helpfull..

File Attachments

1) SigMon.rar, downloaded 388 times

Subject: Re: PROPOSAL: Monitor mutex in objects
Posted by mirek on Sun, 18 Jul 2010 07:05:49 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Thu, 15 July 2010 05:42hi guys,

in C# there exists a nice feature,

```
lock(objectinstance)
{
//method code on now protected object
}
```

one can protect another object of beeing somehow accessed / modified, while oneself is treating with it. the object itself doesnt know about that and does not need to take care on providing any synchonisation features inside.

would it be possible to make something like that in upp as well?
as far as i know, c# compiler helps here somehow..

meanwhile:

here is another maybe usefull construction, came in mind while i was dealing with Mt.h..

```
#define LOCKER UPP::Mutex __locker
#define LOCKED INTERLOCKED_(__locker)

class MyClass
{
 void Method()
 {
  LOCKED
  {
  //your per object locked code
  }
 }
};
private:
 LOCKER;
}
```

it's unlicke the pure INTERLOCKED, which would generate a StaticMutex in every method the LOCKED is used.

if it is sensefull...

```
class MyClass {
    Mutex lock;
    void Method() {
       INTERLOCKED_(lock) {
           ...
       }
    }
};
```

or (usually I like that more)

```
class MyClass {
    Mutex lock;
    void Method() {
       Mutex::Lock __(lock);
       ...
    }
};
```

I do not think we really need more...

Subject: Re: PROPOSAL: Monitor mutex in objects
Posted by mirek on Sun, 18 Jul 2010 07:08:50 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Thu, 15 July 2010 08:11
third section uses a Single() Mutex map, to protect arbitrary objects from 'outside' as i understood it from C#

maybe its helpfull..

Maybe you might just want to inherit Mutex for class where you need that?

Still, my limited experience with MT tells me that such constructs in U++/MT are rarely needed.

Subject: Re: PROPOSAL: Monitor mutex in objects
Posted by kohait00 on Mon, 19 Jul 2010 12:17:29 GMT
View Forum Message <> Reply to Message

your're right KISS is better here. and your last example is less work typing... i might change the example though to be an INTERLOCKED ... Mutex::Lock example if not present..