Subject: Re: MT/Locking Questions
Posted by mirek on Mon, 28 Apr 2008 17:41:59 GMT
View Forum Message <> Reply to Message

captainc wrote on Mon, 28 April 2008 09:23This is a general question about using Mutex locking. Does locking occur on a per object basis? what about static variables?

Also, how do locks work in class hierarchies?
For example:

```
class Parent{
  Mutex  lock;
  String mydata;
  virtual void DoSomething(){
     for(int i=0;i<10;++i){
       Cout() << mydata << "\n;"
     }
  };
};

class Child1{
  void DoSomething(){
    INTERLOCKED_(lock){
       mydata = "Foo";
       Parent::DoSomething();
    }
  }
};

class Child2{
 void DoSomething(){
    INTERLOCKED_(lock){
       mydata = "Bar";
       Parent::DoSomething();
    }
  }
};

CONSOLE_APP_MAIN{
  Parent * c1 = new Child1();
  Parent * c2 = new Child2();

  Thread().Run(callback(&c1, &Parent::DoSomething);
  Thread().Run(callback(&c2, &Parent::DoSomething);

  delete c1;
  delete c2;
}
```

Since 2 separate instances of the object are created, both DoSomething()'s can run concurrently, correct?

Yes and you do not even need the lock

General practice for C++ programming is, well, to put it simple:

Ignore MT in the class design as long as you do not need it.

In practice, this means that you need to serialize all write method calls for exclusive access and all read methods calls for shared access if you access single instance from more than one thread concurently.... (client code is responsible for locking).

Mirek