Subject: Re: Use same variable in different threads
Posted by gprentice on Sun, 02 Jan 2011 02:08:35 GMT
View Forum Message <> Reply to Message

It's time you learnt C++ templates Koldo

ScopedLock is a template class because it's declared like this

```
template<class MUTEX>
class ScopedLock
{
```

It has one template parameter named MUTEX. When you use a template class you have to specify template arguments corresponding to all the template parameters
i.e.

```
#include <Core/Mt.h>
struct mySharedData {
 int a, b;
 String c;
 Mutex m;
};

mySharedData data;

    // ...
    ScopedLock<Mutex> xyz(data.m);
```

where Mutex (the template argument in between <>) is the U++ mutex class in core/mt.h
You also need to change the call mutex.lock() to mutex.Enter() and mutex.unLock() to mutex.Leave

It's actually pretty much pointless to make the ScopedLock class a template class so you could either change it to a normal class or maybe put a default argument

```
template<class MUTEX = Mutex>
class ScopedLock
{
```

then you can do

```
    // ...
    ScopedLock xyz(data.m);
```

When xyz goes out of scope, the destructor is called which releases the mutex.

By the way, this code is invalid as you can't call a constructor.

```
    ScopedLock(data.m);// Enter the section that ...
```

The Mutex::Lock class that Mirek mentioned, creates a temporary mutex so can't be used in your case if you have other code elsewhere that is accessing the data.

Graeme
