Subject: Simple class to handle variables used by different threads
Posted by koldo on Fri, 21 Aug 2015 11:41:14 GMT
View Forum Message <> Reply to Message

Hello all

I wanted to ask you if this class could serve to manage variables that may be read and changed from different threads

```
template <class T>
class threadSafe {
public:
 threadSafe()   {}
 threadSafe(T v)  {val = v;}
 void operator=(T v) {BarrierWrite(val, v);}
 operator T()   {return ReadWithBarrier(val);}

private:
 volatile T val;
};
```

It can be used as simple as this:

```
threadSafe<int> val = 23;
double d = val + 3.5;
```

Other sample in two threads:

```
// Main thread
threadSafe<bool> thread1Busy;

// Thread 1
thread1Busy = true;
...
thread1Busy = false;

// Thread 2
...
while (thread1Busy)
 Sleep(100);
...
```

Subject: Re: Simple class to handle variables used by different threads
Posted by Klugier on Sat, 22 Aug 2015 12:00:22 GMT
View Forum Message <> Reply to Message

Hello,

IMHO, You should name your class ThreadSafe instead of threadSafe (According to the Ultimate++ coding standard).

Sincerely,
Klugier

Subject: Re: Simple class to handle variables used by different threads
Posted by koldo on Sat, 22 Aug 2015 14:08:54 GMT
View Forum Message <> Reply to Message

And do you think is right? It is so simple that I am imagining that something is wrong, either in the implementation or in the concept.

Subject: Re: Simple class to handle variables used by different threads
Posted by mirek on Sun, 23 Aug 2015 18:34:24 GMT
View Forum Message <> Reply to Message

koldo wrote on Fri, 21 August 2015 13:41Hello all

I wanted to ask you if this class could serve to manage variables that may be read and changed from different threads

```
template <class T>
class threadSafe {
public:
 threadSafe()   {}
 threadSafe(T v)  {val = v;}
 void operator=(T v) {BarrierWrite(val, v);}
 operator T()   {return ReadWithBarrier(val);}

private:
 volatile T val;
};
```

It can be used as simple as this:

```
threadSafe<int> val = 23;
double d = val + 3.5;
```

Other sample in two threads:

```
// Main thread
threadSafe<bool> thread1Busy;

// Thread 1
thread1Busy = true;
...
thread1Busy = false;
```

```
// Thread 2
...
while (thread1Busy)
 Sleep(100);
...
```

I would be quite worried about

```
threadSafe<int> x;

x = x + 1;
```

Also, I am not quite sure that barriers are used correctly here.. What are they supposed to do?

BTW, second example would work fine without any synchronization at all...

Mirek

Subject: Re: Simple class to handle variables used by different threads
Posted by koldo on Mon, 24 Aug 2015 07:05:03 GMT
View Forum Message <> Reply to Message

Hello Mirek

The goal is to have variables that can be read and written safely in different threads. Of course there will be an efficiency penalty .

Subject: Re: Simple class to handle variables used by different threads
Posted by mirek on Mon, 24 Aug 2015 09:34:12 GMT
View Forum Message <> Reply to Message

koldo wrote on Mon, 24 August 2015 09:05Hello Mirek

The goal is to have variables that can be read and written safely in different threads. Of course there will be an efficiency penalty .

Well, but "can be read and written" is very broad term...

E.g. most primitive types can already be written/read "safely" (without synchronization). The issue with MT is not this, but "transactions", where several reads and writes are grouped and should form a transaction. Like x = x + 1.

Subject: Re: Simple class to handle variables used by different threads
Posted by koldo on Mon, 24 Aug 2015 13:05:41 GMT
View Forum Message <> Reply to Message

Does this mean that declaring a variable as volatile is enough to read and write it from different threads safely?

Subject: Re: Simple class to handle variables used by different threads
Posted by Mindtraveller on Mon, 24 Aug 2015 17:08:19 GMT
View Forum Message <> Reply to Message

Of course, making variable 'volatile' actually doesn't guarantee anything when we discuss multithreading issues. It just says to compiler 'please, don't optimize it as it may be changed outside your source code'.

The reason it 'sort of' works under modern x86/x64 systems is that modern CPUs change char/int variable with single CPU instruction which effectively avoids multithreading issues. If your variable is anything but char/int (+int64 under x64) you may have multithreading issues when variable is accessed while being updated in another thread.

You should also notice that under arm/mips systems you may have issues even with int32/int64 depending on particular CPU used.

I guess we have nice solution in U++. There's Atomic type which guarantees no multithreading issues with possible support with native OS/CPU commands. So why should we really duplicate its functionality?

Subject: Re: Simple class to handle variables used by different threads
Posted by mirek on Mon, 24 Aug 2015 17:13:37 GMT
View Forum Message <> Reply to Message

Well, depends on type and CPU...

Usually, if type is directly supported by CPU, it works.

Example of type that does not work is e.g. int64 in 32-bit mode. It is because it is, on CPU level, compound type...

Anyway, MT is not hard because of transfering data between threads. If you think about it, they have to get transfered at some point... The real issues are order:

if one thread does

```
a = a1;
b = b1;
```

then another thread can see b updated first. This is what barriers are for.

(Then, of course, there is an issue that you cannot never tell WHEN the change is visible in another thread, but that is really not a problem...)

And then, of course, serialization, which is about transactions:

a = a + 1;

now this, on CPU level is something like

read a into register
increment register // remember this point A
write register into a

Now imagine if another thread starts incrementing a at point A...

Well, there really is complex theory about lock-less multithreading, it is pretty tough stuff. I would not dare suggesting new MT tools before reading it all

Mirek

Subject: Re: Simple class to handle variables used by different threads
Posted by koldo on Mon, 24 Aug 2015 22:13:03 GMT
View Forum Message <> Reply to Message

Thank you Mindtraveller and Mirek

In summary, could a general all purpose thread safe class be like this?

```
template <class T>
class ThreadSafe {
public:
 ThreadSafe()   {}
 ThreadSafe(T v)  {val = v;}
 void operator=(T v) {
  mutex.EnterWrite();
  val = v;
  mutex.LeaveWrite();
 }
 operator T() {
  T tmp;
  mutex.EnterRead();
  tmp = val;
  mutex.LeaveRead();
  return tmp;
```

```
 }
 operator++()....

private:
 volatile T val;
 RWMutex mutex;
};
```

Subject: Re: Simple class to handle variables used by different threads
Posted by sergeynikitin on Mon, 24 Aug 2015 22:28:54 GMT
View Forum Message <> Reply to Message

It's better if you decide with mutex, volatile, atomic and barriers and in every custom case.
If you build multi-thread program, then you must achieve some goals, like performance or other.

But situation is change very fast. Still come in Intel CoreI7, single thread programs is faster or same as multi-thread (in high-load segment), because is needed some time for switching processor context.

Subject: Re: Simple class to handle variables used by different threads
Posted by mirek on Tue, 25 Aug 2015 13:44:23 GMT
View Forum Message <> Reply to Message

koldo wrote on Tue, 25 August 2015 00:13Thank you Mindtraveller and Mirek

In summary, could a general all purpose thread safe class be like this?

```
template <class T>
class ThreadSafe {
public:
 ThreadSafe()   {}
 ThreadSafe(T v)  {val = v;}
 void operator=(T v) {
  mutex.EnterWrite();
  val = v;
  mutex.LeaveWrite();
 }
 operator T() {
  T tmp;
  mutex.EnterRead();
  tmp = val;
  mutex.LeaveRead();
  return tmp;
 }
 operator++()....
```

```
private:
 volatile T val;
 RWMutex mutex;
};
```

The problem is this does not solve many issues.... (like, almost nothing

Mirek

Subject: Re: Simple class to handle variables used by different threads
Posted by koldo on Tue, 25 Aug 2015 15:04:41 GMT
View Forum Message <> Reply to Message

Thank you all for your posts. You all are experts in MT and underline the things I am doing wrong.

Could you give some simple examples of how to do the things right using U++?. For example a slow process gathering data of different non int variables (String, ...) and another process that will get that data continuously, but when available.

Subject: Re: Simple class to handle variables used by different threads
Posted by sergeynikitin on Tue, 25 Aug 2015 17:27:30 GMT
View Forum Message <> Reply to Message

Very useful short article
http://blog.csdn.net/win_lin/article/details/8274810
(msdn article, but I found it only on china)

Russian link: http://www.cyberguru.ru/microsoft-net/csharp-net/multithread
-code.html?showall=1

Article found in October'2008 MSDN Magazine
(https://msdn.microsoft.com/en-us/magazine/ee310108.aspx:
http://download.microsoft.com/download/3/a/7/3a7fa450-1f33-4
1f7-9e6d-3aa95b5a6aea/MSDNMagazine2008_10en-us.chm)

(entity actual in .Net and C++)

Subject: Re: Simple class to handle variables used by different threads
Posted by koldo on Tue, 25 Aug 2015 20:45:24 GMT

View Forum Message <> Reply to Message

sergeynikitin wrote on Tue, 25 August 2015 19:27Very useful short article
http://blog.csdn.net/win_lin/article/details/8274810
(msdn article, but I found it only on china)

Russian link: http://www.cyberguru.ru/microsoft-net/csharp-net/multithread
-code.html?showall=1

Article found in October'2008 MSDN Magazine
(https://msdn.microsoft.com/en-us/magazine/ee310108.aspx:
http://download.microsoft.com/download/3/a/7/3a7fa450-1f33-4
1f7-9e6d-3aa95b5a6aea/MSDNMagazine2008_10en-us.chm)

(entity actual in .Net and C++)
Thank you Sergey, but I can only see articles in Visual Basic and C#.

I have read many articles explaining many different ways to do many things wrong (with many contradictions between authors...), but almost nothing about how to do a few things right, including U++

Subject: Re: Simple class to handle variables used by different threads
Posted by sergeynikitin on Wed, 26 Aug 2015 03:01:26 GMT
View Forum Message <> Reply to Message

For me:
1) I decide, that common variables of all threads, I place to ThreadStore (analog of your ThreadSafe).
2) Most of variables not needed to place to ThreadStore class and it's locals for thread.
3) I plan and program custom processing for every member-variable of ThreadStore class.

Note (1), that most of Container type (Vectors, Arrays, Indexes) are not thread-safe!!!

Note (2), that Write/Read barrier is needed same way as mutexes/semafors for synchronisation data access. (Mutex/semafors - prevent inter-thread collision, Write/Read barriers - prevent inter-processor-cores collisions)

Your class uses only Write/Read barriers.
If you place under mutex of all ThreadSafe class and all it's member, then performance is become loss. Custom variable processing more fast and safe way.

PS
If I'm mistaken - pls correct.

Subject: Re: Simple class to handle variables used by different threads
Posted by koldo on Wed, 26 Aug 2015 05:24:57 GMT
View Forum Message <> Reply to Message

Thank you Sergey

I have not found references to "ThreadStore" so I imagine it is a class developed by you.
Custom variable processing more fast and safeCould you put an example of this with U++?

Subject: Re: Simple class to handle variables used by different threads
Posted by sergeynikitin on Wed, 26 Aug 2015 23:37:15 GMT
View Forum Message <> Reply to Message

ThreadStore as Global or Module vars place for thread . Individually for every project. Very useful.

I can't place full example (not mine, proprietary projects).

But it's nothing difficult! Mutexes, Write Read Barriers by atomic. Every Thread interchange vars - volatile.

MSDN Magazine Example - in CHM file (last link) has C \ C++ discussion.