# Ruby - Feature #7087

## ::ConditionVariable#wait does not work with Monitor because Monitor#sleep does not exist

09/30/2012 12:19 AM - rklemme (Robert Klemme)

| | | |
|---|---|---|
| **Status:** | Assigned | |
| **Priority:** | Normal | |
| **Assignee:** | matz (Yukihiro Matsumoto) | |
| **Target version:** | | |

**Description**

See program attached to bug #7086: timeout_4 always throws:

ERROR: method "timeout_4": #<NoMethodError: private method `sleep' called for #Monitor:0x87e49f8>

```
$ irb19 -r monitor
irb(main):001:0> Monitor.new.method(:sleep)
=> #<Method: Monitor(Kernel)#sleep>
irb(main):002:0> Monitor.instance_methods.grep /sleep/
=> []
```

### History

#### #1 - 09/30/2012 12:52 AM - kosaki (Motohiro KOSAKI)

At least, this is intentional. condtion variable and monitor have different inspiration source.
condition variable is based on POSIX CV and monitor is based on Java monitor semantics. To be
honest, I'm not familiar Java's conditon variable. Could you please explain the detail of your
suggestion? Which semantics do you hope?

#### #2 - 09/30/2012 01:32 AM - rklemme (Robert Klemme)

kosaki (Motohiro KOSAKI) wrote:

> At least, this is intentional. condtion variable and monitor have different inspiration source.
> condition variable is based on POSIX CV and monitor is based on Java monitor semantics. To be
> honest, I'm not familiar Java's conditon variable.

For me the difference between Mutex and Monitor is only reentrancy.  Other than that I'd have expected them to be identical (usage and interface).

> Could you please explain the detail of your
> suggestion? Which semantics do you hope?

I would have expected that I can use a Monitor / instance which includes MonitorMixin the same way as a Mutex with ::ConditionVariable.  If that is not intended then I'd rather have an exception TypeError from ::ConditionVariable#wait indicating that a Mutex must be used than the exception about the private sleep method (which happens to be the one from Kernel).

#### #3 - 10/01/2012 06:38 PM - shugo (Shugo Maeda)

kosaki (Motohiro KOSAKI) wrote:

> monitor is based on Java monitor semantics.

My Java knowledge is rusted, but monitor is not based on Java monitor semantics, at least when it was designed.  Java's monitor mechanism mixes a mutex and a condition variable into one object, but I don't think it's well designed.  I guess Java has a better solution (java.util.concurrent.locks.{Lock,Condition}?) now.  The name monitor was taken from Java, but it was just because the name mutex had been already taken at that time.

rklemme (Robert Klemme) wrote:

> kosaki (Motohiro KOSAKI) wrote:
>
> > At least, this is intentional. condtion variable and monitor have different inspiration source.
> > condition variable is based on POSIX CV and monitor is based on Java monitor semantics. To be
> > honest, I'm not familiar Java's conditon variable.

> For me the difference between Mutex and Monitor is only reentrancy.  Other than that I'd have expected them to be identical (usage and interface).

Monitor has its own version of ConditionVariable.  Use Monitor#new_cond to get its instance.

>> Could you please explain the detail of your
>> suggestion? Which semantics do you hope?

> I would have expected that I can use a Monitor / instance which includes MonitorMixin the same way as a Mutex with ::ConditionVariable.  If that is not intended then I'd rather have an exception TypeError from ::ConditionVariable#wait indicating that a Mutex must be used than the exception about the private sleep method (which happens to be the one from Kernel).

It's not intended, but TypeError is not preferred in Ruby because it breaks duck typing.
I think it's better to add Mutex#new_cond and to make the argument of ConditionVariable#wait obsolete.
Java's java.util.concurrent.locks.Lock seems to have an equivalent method called newCondition().

**#4 - 10/02/2012 03:47 AM - rklemme (Robert Klemme)**

shugo (Shugo Maeda) wrote:

> kosaki (Motohiro KOSAKI) wrote:
>
>> monitor is based on Java monitor semantics.
>
> My Java knowledge is rusted, but monitor is not based on Java monitor semantics, at least when it was designed.  Java's monitor mechanism mixes a mutex and a condition variable into one object, but I don't think it's well designed.

Mutex is not reentrant so there is one reason why Mutex cannot be designed after Java monitor.  Monitor on the other hand is reentrant and so is synchronize in Java.  Regarding the monitor semantics Monitor and Java's synchronize behave identical.  You are right that Java's monitor also includes a CV.

> I guess Java has a better solution (java.util.concurrent.locks.{Lock,Condition}?) now.

Yes, it's better - IMHO mainly because the user can control how many condition variables he wants to associate with the lock.  Without that you always have to wake up all waiting threads also those who wait for another condition.

> rklemme (Robert Klemme) wrote:
>
>> kosaki (Motohiro KOSAKI) wrote:
>>
>>> At least, this is intentional. condtion variable and monitor have different inspiration source.
>>> condition variable is based on POSIX CV and monitor is based on Java monitor semantics. To be
>>> honest, I'm not familiar Java's conditon variable.
>>
>> For me the difference between Mutex and Monitor is only reentrancy.  Other than that I'd have expected them to be identical (usage and interface).
>
> Monitor has its own version of ConditionVariable.  Use Monitor#new_cond to get its instance.

I know (see the test program).  But if I use a ::ConditionVariable with a Monitor the error message is irritating.

>> I would have expected that I can use a Monitor / instance which includes MonitorMixin the same way as a Mutex with ::ConditionVariable.  If that is not intended then I'd rather have an exception TypeError from ::ConditionVariable#wait indicating that a Mutex must be used than the exception about the private sleep method (which happens to be the one from Kernel).
>
> It's not intended, but TypeError is not preferred in Ruby because it breaks duck typing.

I think that aspect is less important in this case because apparently nobody wants to add sleep to Monitor so ::ConditionVariable will work and also ::CV heavily depends on the passed Mutex's internals so it seems unlikely that there will be another instance usable with CV.

> I think it's better to add Mutex#new_cond and to make the argument of ConditionVariable#wait obsolete.

Of course that's an even better solution but it may break existing code - unless you allow #wait to accept 0, 1 or 2 arguments.  With one argument one also needs to check whether it's a timeout value or a Mutex.

Btw, I am not sure why Mutex is still in the language as Monitor / MonitorMixin seem more versatile.  Do you know why?  Is it to keep compatibility? We could alias Mutex to Monitor though.

> Java's java.util.concurrent.locks.Lock seems to have an equivalent method called newCondition().

Yes.

**#5 - 10/02/2012 03:11 PM - shugo (Shugo Maeda)**

*- Category set to lib*

*- Assignee set to matz (Yukihiro Matsumoto)*

rklemme (Robert Klemme) wrote:

> shugo (Shugo Maeda) wrote:
>
> > kosaki (Motohiro KOSAKI) wrote:
> >
> > > monitor is based on Java monitor semantics.
> >
> > My Java knowledge is rusted, but monitor is not based on Java monitor semantics, at least when it was designed.  Java's monitor mechanism mixes a mutex and a condition variable into one object, but I don't think it's well designed.
>
> Mutex is not reentrant so there is one reason why Mutex cannot be designed after Java monitor.

I heard the word "reentrant mutex" in the late 90's, but I don't know whether it's Java's invention or not.

> Monitor on the other hand is reentrant and so is synchronize in Java.  Regarding the monitor semantics Monitor and Java's synchronize behave identical.  You are right that Java's monitor also includes a CV.
>
> > I guess Java has a better solution (java.util.concurrent.locks.{Lock,Condition}?) now.
>
> Yes, it's better - IMHO mainly because the user can control how many condition variables he wants to associate with the lock.  Without that you always have to wake up all waiting threads also those who wait for another condition.

Yes, that's why I've separated MonitorMixin::ConditionVariable from MonitorMixin.

> > Monitor has its own version of ConditionVariable.  Use Monitor#new_cond to get its instance.
>
> I know (see the test program).  But if I use a ::ConditionVariable with a Monitor the error message is irritating.
>
> > > I would have expected that I can use a Monitor / instance which includes MonitorMixin the same way as a Mutex with ::ConditionVariable.  If that is not intended then I'd rather have an exception TypeError from ::ConditionVariable#wait indicating that a Mutex must be used than the exception about the private sleep method (which happens to be the one from Kernel).
> >
> > It's not intended, but TypeError is not preferred in Ruby because it breaks duck typing.
>
> I think that aspect is less important in this case because apparently nobody wants to add sleep to Monitor so ::ConditionVariable will work and also ::CV heavily depends on the passed Mutex's internals so it seems unlikely that there will be another instance usable with CV.

Hmm.... What do you think of it, Matz?

> > I think it's better to add Mutex#new_cond and to make the argument of ConditionVariable#wait obsolete.
>
> Of course that's an even better solution but it may break existing code - unless you allow #wait to accept 0, 1 or 2 arguments.  With one argument one also needs to check whether it's a timeout value or a Mutex.
>
> Btw, I am not sure why Mutex is still in the language as Monitor / MonitorMixin seem more versatile.  Do you know why?  Is it to keep compatibility?  We could alias Mutex to Monitor though.

One of the reasons is a historical reason.
In chronological order:

- Ruby had only Mutex, no ConditionVariable.
- I implemented MonitorMixin, MonitorMixin::ConditionVariable, and Monitor as a reentrant mutex.
- Someone (Hara-san?) implemented ConditionVariable for Mutex, but he chose pthread-like ConditionVariable#wait.
  I don't know why.

Another reason may be a performance issue, but I'm not sure.

**#6 - 10/20/2012 01:23 PM - olegshaldybin (Oleg Shaldybin)**

On Oct 1, 2012, at 2:38 AM, "shugo (Shugo Maeda)" redmine@ruby-lang.org wrote:

> Issue #7087 has been updated by shugo (Shugo Maeda).
>
> kosaki (Motohiro KOSAKI) wrote:
>
> > monitor is based on Java monitor semantics.
>
> My Java knowledge is rusted, but monitor is not based on Java monitor semantics, at least when it was designed.  Java's monitor mechanism mixes a mutex and a condition variable into one object, but I don't think it's well designed.  I guess Java has a better solution (java.util.concurrent.locks.{Lock,Condition}?) now.  The name monitor was taken from Java, but it was just because the name mutex had been already taken at that time.
>
> rklemme (Robert Klemme) wrote:
>
> > kosaki (Motohiro KOSAKI) wrote:
> >
> > > At least, this is intentional. condtion variable and monitor have different inspiration source.
> > > condition variable is based on POSIX CV and monitor is based on Java monitor semantics. To be
> > > honest, I'm not familiar Java's conditon variable.
> >
> > For me the difference between Mutex and Monitor is only reentrancy.  Other than that I'd have expected them to be identical (usage and interface).
>
> Monitor has its own version of ConditionVariable.  Use Monitor#new_cond to get its instance.
>
> > > Could you please explain the detail of your
> > > suggestion? Which semantics do you hope?
> >
> > I would have expected that I can use a Monitor / instance which includes MonitorMixin the same way as a Mutex with ::ConditionVariable.  If that is not intended then I'd rather have an exception TypeError from ::ConditionVariable#wait indicating that a Mutex must be used than the exception about the private sleep method (which happens to be the one from Kernel).
>
> It's not intended, but TypeError is not preferred in Ruby because it breaks duck typing.
> I think it's better to add Mutex#new_cond and to make the argument of ConditionVariable#wait obsolete.
> Java's java.util.concurrent.locks.Lock seems to have an equivalent method called newCondition().
>
> ---
>
> Bug #7087: ::ConditionVariable#wait does not work with Monitor because Monitor#sleep does not exist
> https://bugs.ruby-lang.org/issues/7087#change-29819
>
> Author: rklemme (Robert Klemme)
> Status: Open
> Priority: Low
> Assignee:
> Category:
> Target version:
> ruby -v: ruby 1.9.3p194 (2012-04-20) [i686-linux]
>
> See program attached to bug #7086: timeout_4 always throws:
>
> ERROR: method "timeout_4": #<NoMethodError: private method `sleep' called for #Monitor:0x87e49f8>
>
> ```
> $ irb19 -r monitor
> irb(main):001:0> Monitor.new.method(:sleep)
> => #<Method: Monitor(Kernel)#sleep>
> irb(main):002:0> Monitor.instance_methods.grep /sleep/
> => []
> ```

--
http://bugs.ruby-lang.org/

**#7 - 10/23/2012 12:55 AM - headius (Charles Nutter)**

A few JRuby tidbits that might help shed some light.

Java's per-object monitors are reentrant, as already stated. They do not, however, provide a mechanism for querying who has locked, how many are waiting, etc. Because of that, JRuby implements Mutex using ReentrantLock and just ensures the same thread cannot lock the same Mutex twice:

https://github.com/jruby/jruby/blob/master/src/org/jruby/ext/thread/Mutex.java

The standard 'thread' library ConditionVariable is implemented atop Mutex, as in Ruby:
https://github.com/jruby/jruby/blob/master/src/org/jruby/ext/thread/ConditionVariable.java

And we use the same monitor.rb as 1.9.3.

**#8 - 11/05/2012 08:47 PM - mame (Yusuke Endoh)**

*- Tracker changed from Bug to Feature*

*- Status changed from Open to Assigned*

In similar way to #7087, I think this is not a bug, but a feature request.
Moved to feature tracker.

--
Yusuke Endoh mame@tsg.ne.jp

**#9 - 11/05/2012 08:47 PM - mame (Yusuke Endoh)**

*- Target version set to 2.6*

**#10 - 12/16/2012 02:56 AM - kosaki (Motohiro KOSAKI)**

Following patch fix this issue, I think.

diff --git a/lib/monitor.rb b/lib/monitor.rb
index 07394b5..30701c7 100644
--- a/lib/monitor.rb
+++ b/lib/monitor.rb
@@ -215,6 +215,10 @@ module MonitorMixin
end
alias synchronize mon_synchronize

- def sleep(timeout = nil)
- @mon_mutex.sleep timeout
- end

- ## Creates a new MonitorMixin::ConditionVariable associated with the receiver.

**#11 - 12/25/2017 06:15 PM - naruse (Yui NARUSE)**

*- Target version deleted (2.6)*