Ruby master - Bug #2488
thread usage can result in bad HANDLE
12/17/2009 03:34 AM - rogerdpack (Roger Pack)

<table>
<thead>
<tr>
<th>Status:</th>
<th>Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td>ko1 (Koichi Sasada)</td>
</tr>
<tr>
<td>Target version:</td>
<td>2.0.0</td>
</tr>
<tr>
<td>ruby -v:</td>
<td>ruby 1.9.2dev (2009-12-31 trunk 26205) [i386-mswin32]</td>
</tr>
<tr>
<td>Backport:</td>
<td></td>
</tr>
</tbody>
</table>

**Description**

```ruby
=begin
require 'thwait'
loop {
  a = []
  10.times { a << Thread.new {}}
  ThreadsWait.all_waits(a)
  print '.'
}
C:\dev\digitalarchive_trunk>ruby -v
ruby 1.9.1p376 (2009-12-07 revision 26041) [i386-mingw32]
C:\dev\digitalarchive_trunk>ruby stress_th.rb
..........[BUG] The handle is invalid.
ruby 1.9.1p376 (2009-12-07 revision 26041) [i386-mingw32]
-- control frame ---------

-- Ruby level backtrace information-------------------------------

[NOTE]
You may encounter a bug of Ruby interpreter. Bug reports are welcome.
For details: [http://www.ruby-lang.org/bugreport.html](http://www.ruby-lang.org/bugreport.html)

This application has requested the Runtime to terminate it in an unusual way.
Please contact the application's support team for more information.

Works fine with 1.9.2

Thanks.
-r
=end
```

**Related issues:**

Related to Backport191 - Bug #3183: "[BUG] The handle is invalid." when worki... Closed 04/22/2010

**History**

**#1 - 12/31/2009 02:32 AM - usa (Usaku NAKAMURA)**
- Assignee set to usa (Usaku NAKAMURA)
- Target version set to 2.0.0

```ruby
=begin
can reproduce with trunk.
=end
```

**#2 - 12/31/2009 03:33 AM - usa (Usaku NAKAMURA)**
- Status changed from Open to Assigned

03/19/2020
#3 - 12/31/2009 08:44 PM - usa (Usaku NAKAMURA)
- Category set to core
- Assignee changed from usa (Usaku NAKAMURA) to ko1 (Koichi Sasada)

#4 - 04/23/2010 01:23 AM - rogerdpack (Roger Pack)

#5 - 04/23/2010 02:18 AM - rogerdpack (Roger Pack)

#6 - 04/23/2010 02:07 PM - luislavena (Luis Lavena)

#7 - 05/01/2010 09:24 PM - wanabe (_ wanabe)
- File lock_before_reset.patch added

#8 - 05/05/2010 06:35 AM - rogerdpack (Roger Pack)

luis (Luis Lopez): I am able to reproduce #3183 (sometimes it takes a few minutes, and requires the puts in there). Ex backtrace:

```ruby
[BUG] w32_reset_event: The handle is invalid.
```

ruby 1.9.2dev (2010-05-05 trunk 27620) [i386-mingw32]
[NOTE]
You may have encountered a bug in the Ruby interpreter or extension libraries.
Bug reports are welcome.
For details: http://www.ruby-lang.org/bugreport.html

This application has requested the Runtime to terminate it in an unusual way.
Please contact the application's support team for more information.

(Windows 7 32 bit, 4GB RAM). I assume it's just a timing issue.

-rp
=end

#9 - 05/05/2010 02:42 PM - usa (Usaku NAKAMURA)
=begin
Hello,
In message "[ruby-core:30005] [Bug #2488] thread usage can result in bad HANDLE"
on May.05,2010 06:35:12, redmine@ruby-lang.org wrote:
| (Windows 7 32 bit, 4GB RAM). I assume it's just a timing issue.

I think so, too.
wanabe-san, please check in the patch and close #2488 and #3183.
Regards
--
U.Nakamura usa@garbagecollect.jp
=end

#10 - 05/05/2010 08:57 PM - wanabe (_ wanabe)
=begin
This issue was solved with changeset r27630.
Roger, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.
=end

#11 - 05/09/2010 07:03 PM - wanabe (_ wanabe)
=begin
Hello,
2010/5/6, Caleb Clausen vikkous@gmail.com:

I don't understand win32 programming that well...

Me too.

Why is it necessary to re-check that intr is set to what it was set to
just two lines before? I assume it might be changed by the action of
another thread, but if that's the case, wouldn't it be better to just
check that field once while inside the global_vm_lock? (ie move the
lock/unlock to be around the whole if statement.)

Yes, you are right.
I think first that it is good to avoid native_mutex_lock() as much as possible.
Now, I realize the condition (th && !intr) is fulfilled in a only few moments.
But if we delete check of outside, new check of mutex's existence will
be needed.
I'm afraid that it doesn't make sense from the standpoint of cost down.
For that matter, shouldn't ruby be using the api provided by windows (WaitForSingleObject, apparently) to check the state of this 'event object' (really a semaphore)? If you do it that way, then maybe the global_vm_lock doesn't need to be touched.

I guess it is not very good idea.
Because th->native_thread_data.interrupt_event can be destroyed by native_thread_destroy() between check and lock, but GVL can't.

And if ruby use 'event object', we should insert locking to somewhere such as thread_cleanup_func(), but lock with GVL is in thread_start_func_2() already. There is not a new cost.

But if you know the use case the lock can be harm, I'm glad to change it.

--
wanabe

=end

Files

<table>
<thead>
<tr>
<th>name</th>
<th>size</th>
<th>date</th>
<th>author</th>
</tr>
</thead>
<tbody>
<tr>
<td>lock_before_reset.patch</td>
<td>1.22 KB</td>
<td>05/01/2010</td>
<td>wanabe (_ wanabe)</td>
</tr>
</tbody>
</table>