Ruby master - Feature #2012
Set event_flags on thread creation if hook exists
08/30/2009 02:21 PM - mark-moseley (Mark Moseley)

<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>
</tbody>
</table>

**Description**

In ruby-debug, when a user sets a breakpoint that ends up executed by a later-created thread, then nothing will happen.

The only solution that I have at this point is to always set RUBY_EVENT_VM for all living threads every time the hook proc is called. I've tried to optimize this and failed; the cost of figuring out accurately if there's a new thread is always greater than the brute-force method of just always forcing the flags to be updated.

I propose adding the following patch to the core:

```c
native_mutex_initialize(&th->interrupt_lock);

if (GET_VM()->event_hooks != NULL)
    th->event_flags |= RUBY_EVENT_VM; /* kick thread */
    st_insert(th->vm->living_threads, thval, (st_data_t) th->thread_id);

native_thread_create(th);
```

This passes all Ruby tests, and fixes the problem that I've outlined.

**History**

#1 - 01/07/2010 11:29 PM - mame (Yusuke Endoh)

=begin
Sorry for very late response!
2009/8/30 Mark Moseley redmine@ruby-lang.org:

In ruby-debug, when a user sets a breakpoint that ends up executed by a later-created thread, then nothing will happen.

Confirmed.

```ruby
> ./ruby.org -e 'set_trace_func{proc {{a| p a }}; Thread.new { puts "foo" }.join

> }'  
[*c-return*, *-e*, 2, :set_trace_func, #<Binding:0x824d10c>, Kernel]  
[*line*, *-e*, 3, nil, #<Binding:0x824cfe0>, nil]  
[*c-call*, *-e*, 3, :new, #<Binding:0x824cfo4>, Thread]  
[*c-call*, *-e*, 3, :initialize, #<Binding:0x824ce00>, Thread]  
[*c-return*, *-e*, 3, :initialize, #<Binding:0x824cfc>, Thread]  
foo  
[*c-return*, *-e*, 3, :new, #<Binding:0x824cc0c>, Thread]  
[*c-call*, *-e*, 3, :join, #<Binding:0x824cb30>, Thread]  
[*c-return*, *-e*, 3, :join, #<Binding:0x824ca54>, Thread]
```

(c-call" event to the method 'puts' is not found)
This issue affects the coverage feature too. It fails to measure coverage that is run in sub thread. [ruby-dev:39950]

I propose adding the following patch to the core:

**Index: thread.c**

```c
--- thread.c (revision 24710)
+++ thread.c (working copy)
@@ -497,6 +497,9 @@
    th->thgroup = GET_THREAD()->thgroup;

    native_mutex_initialize(&th->interrupt_lock);

-   if (GET_VM())->event_hooks != NULL)
-       th->event_flags |= RUBY_EVENT_VM; + /* kick thread */ st_insert(th->vm->living_threads, thval, (st_data_t) th->thread_id);
-    native_thread_create(th);

Basically I agree with your patch. But it leaves event_flags true even if all vm-level hooks are removed. How about this patch:

```c
diff --git a/thread.c b/thread.c
index 0ba41c7..c082471 100644
--- a/thread.c
+++ b/thread.c
@@ -531,6 +531,9 @@ thread_create_core(VALUE thval, VALUE args, VALUE
    th->thgroup = GET_THREAD()->thgroup;

    native_mutex_initialize(&th->interrupt_lock);

-   if (GET_VM())->event_hooks != NULL)
-       th->event_flags |= RUBY_EVENT_VM; +   /* kick thread */ st_insert(th->vm->living_threads, thval, (st_data_t) th->thread_id);
-   err =
-    native_thread_create(th); @@ -3750,7 +3753,12 @@
- exec_event_hooks(th->event_hooks, flag, self, id, klass);
- if (wait_event & RUBY_EVENT_VM) {
- exec_event_hooks(th->vm->event_hooks, flag, self, id, klass);
-   }
- else {
-      exec_event_hooks(th->vm->event_hooks, flag, self, id, klass);
-   }  
   th->errinfo = errinfo; }

---
Yusuke ENDOH mame@tsg.ne.jp
=end

#2 - 01/12/2010 02:31 AM - mame (Yusuke Endoh)
- Category set to core
- Assignee set to ko1 (Koichi Sasada)

=begin
=end

#3 - 01/28/2010 11:57 AM - rogerdpack (Roger Pack)
=begin
This patch seems to fix the threading difficulties for ruby-prof as well.

Koichi could you take a look at it sometime?
-t
=end

#4 - 03/24/2010 11:57 PM - mame (Yusuke Endoh)
- Status changed from Open to Closed
- % Done changed from 0 to 100
This issue was solved with changeset r27033.
Mark, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

Hi,

2010/1/7 Yusuke ENDOH mame@tsg.ne.jp:

2009/8/30 Mark Moseley redmine@ruby-lang.org:

In ruby-debug, when a user sets a breakpoint that ends up executed by a later-created thread, then nothing will happen.

I've committed the patch I wrote ([ruby-core:27471]).

I believe that this is certainly a bug and my patch is benign, but I'm happy to revert it if ko1 disagree.

--
Yusuke ENDOH mame@tsg.ne.jp