Ruby - Bug #10540

Yielded fibers do not execute ensure blocks

11/24/2014 08:11 PM - wycats (Yehuda Katz)

Status:	Closed		
Priority:	Normal		
Assignee:	ko1 (Koichi Sasada)		
Target version:	2.6		
ruby -v:	ruby 2.1.2p95 (2014-05-08 revision 45877) [x86_64-darwin13.0]	**Backport:**	2.0.0: UNKNOWN, 2.1: UNKNOWN

Description

When a thread has paused fibers, if an exception occurs anywhere in the thread, ensure blocks in the paused fibers do not execute.

The effect of this is that block-scoped resources (like File.open, Mutex#synchronize, ActiveRecord transactions) can easily leak due to non-local effects. This may result in strange effects, like checking connections back into a connection pool that are in the wrong state, where the correct state was guarded with an ensure block.

Here is a very simple repro of this situation:

```
Thread.new {
  Fiber.new {
    begin
      puts "YIELD"
      Fiber.yield
    ensure
      puts "UNWIND"
    end
  }.resume
  raise
}
```

Expected result: YIELD is printed, followed by UNWIND
Actual result: YIELD is printed, but UNWIND is never printed

Related issues:

Related to Ruby - Bug #595: Fiber ignores ensure clause	**Closed**

History

#1 - 11/24/2014 09:08 PM - normalperson (Eric Wong)

This seems unfortunate. I'm not sure if there's a good way to do this automatically with current APIs because Fibers require explicit scheduling.

Perhaps storing Fibers in Thread.current and having a terminate_all_fibers method can be helpful:

```
Thread.new do
  begin
    do_something_which_creates_fibers
  ensure
    # all thread-local fibers are resumed until they're dead
    terminate_all_fibers
  end
end
```

#2 - 11/25/2014 07:58 AM - ko1 (Koichi Sasada)

- Assignee set to ko1 (Koichi Sasada)

- Target version set to 2.6

This issue is a known bug.
[Bug #595]: Fiber ignores ensure clause
(3 digits!)

Rubinius supports it.

This feature is not impossible, but a bit difficult because there are two ENSURE points.

1. Owner thread is terminated.
2. Fiber object is collected.

In this issue, (1) is pointed out. It is easy to implement by sending EXIT exception for owned fibers.
However, (2) is a problem. We can not use finalizers for each fibers because it is too late (in finalizer, Fiber object should be already terminated). To solve this issue, we need to separate fiber handler and fiber cotext itself. In other words, it is possible, but tough hack.

And another issue is compatibility. It is too late to fix this issue for 2.2.
I try to implement this fix and apply in early 2015.

#3 - 01/31/2017 01:52 PM - ko1 (Koichi Sasada)

- Related to Bug #595: Fiber ignores ensure clause added

#4 - 01/31/2017 01:53 PM - ko1 (Koichi Sasada)

- Status changed from Open to Closed

See #595