Ruby master - Bug #595

Fiber ignores ensure clause

09/24/2008 07:28 PM - ko1 (Koichi Sasada)

Status:	Assigned		
Priority:	Normal		
Assignee:	ioquatix (Samuel Williams)		
Target version:			
ruby -v:	-	**Backport:**	

Description

Ruby プロセス終了時，Fiber が ensure を無視します．
これは，前から直そうと思って手がついていなかった問題です．
10月末までには直そうと思います．結構複雑なので，後回しにしていましました．

```
fib = Fiber.new{
  begin
    Fiber.yield :ok
  ensure
    puts "should be print out"
  end
}
p fib.resume
```

Related issues:

Related to Ruby master - Bug #10540: Yielded fibers do not execute ensure blocks	**Closed**	
Related to Ruby master - Feature #10344: [PATCH] Implement Fiber#raise	**Closed**	
Is duplicate of Ruby master - Bug #2460: RubySpecでFiberのSpecがおちる	**Closed**	**12/08/2009**

History

#1 - 09/28/2008 04:57 PM - yugui (Yuki Sonoda)

- Target version set to 1.9.1 Release Candidate

=begin

=end

#2 - 10/29/2008 01:13 PM - rogerdpack (Roger Pack)

=begin
Help me out--shouldn't this print out only when you call fib.resume twice?

fib = Fiber.new{
begin
Fiber.yield :ok
ensure
puts "should be print out"
end
}
p fib.resume
p fib.resume

prints out all right.
=end

#3 - 12/11/2008 10:40 AM - yugui (Yuki Sonoda)

- Target version changed from 1.9.1 Release Candidate to 2.0.0

=begin

=end

#4 - 12/11/2008 10:41 AM - yugui (Yuki Sonoda)

=begin
パッチを書くか、もしくはドキュメントにKNOWN BUGとして書く、ということで。
=end

#5 - 06/02/2009 08:41 PM - wanabe (_ wanabe)

=begin
ワナベと申します。

かなり前のチケットですが、題名の件についてパッチを書きました。
もしまだどなたもパッチを書かれていないようならご検討ください。

Index: thread.c
===
--- thread.c (リビジョン 23617)
+++ thread.c (作業コピー)
@@ -293,6 +293,8 @@

static void rb_mutex_unlock_all(mutex_t *mutex, rb_thread_t *th);

+void rb_fiber_terminate_all(rb_thread_t *th);
+
void
rb_thread_terminate_all(void)
{
@@ -310,6 +312,7 @@

```
  thread_debug("rb_thread_terminate_all (main thread: %p)\n", (void *)th);
  st_foreach(vm->living_threads, terminate_i, (st_data_t)th);
```

- rb_fiber_terminate_all(th);

  while (!rb_thread_alone()) {
  PUSH_TAG();
  @@ -1210,6 +1213,7 @@
  thread_debug("rb_thread_execute_interrupts: %ld\n", err);

  ```
  if (err == eKillSignal || err == eTerminateSignal) {
  ```

- [redacted]

  ```
  th->errinfo = INT2FIX(TAG_FATAL);
  TH_JUMP_TAG(th, TAG_FATAL);
  }
  ```

  # Index: cont.c

  --- cont.c (リビジョン 23617)
  +++ cont.c (作業コピー)
  @@ -534,6 +534,7 @@
  case 0:
  return Qnil;
  case 1:

- [redacted]

  return argv[0];
  default:
  return rb_ary_new4(argc, argv);
  @@ -946,6 +947,36 @@
  return fib->status != TERMINATED ? Qtrue : Qfalse;
  }

+static VALUE
+terminate_all_i(VALUE fibval)
+{

- if (rb_fiber_alive_p(fibval)) {
- VALUE value = rb_exc_new2(rb_eSystemExit, "terminate");
- return fiber_switch(fibval, -1, &value, 0);
- } +} + +void +rb_fiber_terminate_all(rb_thread_t *th) +{
- VALUE fibval;
- rb_fiber_t *fib, *root_fib;
- rb_thread_t *_th = GET_THREAD(); +

- rb_thread_set_current(th);
- fibval = th->root_fiber;
- if (!RTEST(fibval)) return;
- GetFiberPtr(fibval, root_fib); +
- fib = root_fib->prev_fiber;
- while (fib != root_fib) {
- rb_rescue2(terminate_all_i, fib->cont.self,
- 0, 0, rb_eSystemExit);
- fib = fib->prev_fiber;
- }
- rb_thread_set_current(_th); +} + /*
    - call-seq:
    - fiber.resume(args, ...) -> obj

--
ワナベ

=end

#6 - 06/15/2009 06:33 AM - ko1 (Koichi Sasada)

=begin
　ささだです．

　返事が随分遅くなってしまってすみません．

wanabe wrote::

> かなり前のチケットですが、題名の件についてパッチを書きました。
> もしまだどなたもパッチを書かれていないようならご検討ください。

　実は，以前似たようなものを作ったのですが，さくっと SEGV の嵐で，面倒く
さいなぁ，と思って放置していたのでした．このパッチですと，たとえば
test-all とかはどうでした？

--
// SASADA Koichi at atdot dot net

=end

#7 - 06/15/2009 06:44 PM - wanabe (_ wanabe)

=begin
ワナベです。

2009/06/15 6:33 に SASADA Koichiko1@atdot.net さんは書きました：

> wanabe wrote::
>
>> かなり前のチケットですが、題名の件についてパッチを書きました。
>> もしまだどなたもパッチを書かれていないようならご検討ください。
>
> 実は，以前似たようなものを作ったのですが，さくっと SEGV の嵐で，面倒く
> さいなぁ，と思って放置していたのでした．このパッチですと，たとえば
> test-all とかはどうでした？

ruby 1.9.2dev (2009-06-14 trunk 23691) [i386-mingw32] では
make test-all で SEGV が出てしまいました。
TestFiber#test_many_fibers_with_threads が E で終わった直後の
TestFiber#test_normal で落ちているようです。
ですがなぜか test_fiber.rb を直接起動するとエラーなしで完走します。

またパッチはLinux環境で書いたのですが、その時のmake test-allでは問題はなく、
今改めて ruby 1.9.2dev (2009-06-15 trunk 23692) [i686-linux] で試してみても
SEGV は発生しませんでした。

どういう事かよく分かりませんが、不安定であることは間違いないので
このパッチは役に立たなさそうです。申し訳ありません。

--
ワナベ

=end

#8 - 06/16/2009 11:24 AM - ko1 (Koichi Sasada)

=begin
▯ささだです．

wanabe wrote::

> ruby 1.9.2dev (2009-06-14 trunk 23691) [i386-mingw32] では
> make test-all で SEGV が出てしまいました。
> TestFiber#test_many_fibers_with_threads が E で終わった直後の
> TestFiber#test_normal で落ちているようです。
> ですがなぜか test_fiber.rb を直接起動するとエラーなしで完走します。
>
> またパッチはLinux環境で書いたのですが、その時のmake test-allでは問題はなく、
> 今改めて ruby 1.9.2dev (2009-06-15 trunk 23692) [i686-linux] で試してみても
> SEGV は発生しませんでした。
>
> どういう事かよく分かりませんが、不安定であることは間違いないので
> このパッチは役に立たなさそうです。申し訳ありません。

▯いえいえ．もうちょっと追跡すればなんとかなると思うのですが，ここはどう
にも難しいですよね．Fiber の集合を保存する，私が最後にあわてて突っ込んだ
部分にバグがあるような気がしています．

▯1.9.2 には間に合わせたいところ．

--
// SASADA Koichi at atdot dot net

=end

#9 - 01/13/2010 06:45 PM - wanabe (_ wanabe)

- File ensure_fiber.patch added

- ruby -v set to ruby 1.9.2dev (2010-01-13) [i386-mingw32]

=begin
以前このチケットについて書いたパッチが SEGV すると書きましたが、
パッチの問題ではなく Bug #1325 の影響だったようです。
改めて確認したところ make test-all では SEGV しませんでした。

それとは別の問題として、例外でジャンプさせると rescue される恐れがあったので
throw/catch を使うように変更したパッチを添付します。
ご検討くだされば幸いです。
=end

#10 - 01/13/2010 06:45 PM - wanabe (_ wanabe)

- File deleted (ensure_fiber.patch)

=begin

=end

#11 - 04/10/2010 11:20 PM - wanabe (_ wanabe)

=begin
ささださん

このチケットおよびパッチについてコメントいただければ幸いです。
この対処で問題ないかどうか、もし問題ないとしたら
1.9.2 リリース前に入れるべきかどうかが気になっています。
そもそもアプローチがまずいようでしたら別の方法を考えます。
=end

#12 - 04/21/2010 11:55 PM - mame (Yusuke Endoh)

=begin
遠藤です。

2010年4月10日23:20 _ wanabe redmine@ruby-lang.org:

> ささださん
>
> このチケットおよびパッチについてコメントいただければ幸いです。
> この対処で問題ないかどうか、もし問題ないとしたら
> 1.9.2リリース前に入れるべきかどうかが気になっています。
> そもそもアプローチがまずいようでしたら別の方法を考えます。

ささださんではないですが勝手にコメントします。

1) プロセス終了時でなく、FiberがGCで回収される時にensure節が
実行されません。

```
1000.times do
Fiber.new do
begin
Fiber.yield
ensure
puts "foo!"
end
end.resume
GC.start
end
```

```
# foo!が1個しか出ない(期待は1000個)
$ ./ruby t.rb
foo!
```

実行中のThreadと同様に、yield中のFiberはGCしないようにする
くらいしか思いつきません。

2) Fiberを作ったスレッドが終了した場合、Fiberのensure節が実行
されません。

```
Thread.new do
Fiber.new do
begin
Fiber.yield
ensure
puts "foo!"
end
end.resume
end
sleep 1
puts "end"
```

```
# foo!が出ない
$ ./ruby t.rb
end
```

プロセスの終了時でなく、スレッドの終了時にFiberを起こさないと
行けない?

どちらも仕様レベルで検討しないといけないことのような気がします。
まだ結構大変そうなので、1.9.2は見送った方がいいんじゃないかなと
思います。早くコメントしていれば、wanabeさんなら余裕で直せたと
思うので残念ですが。

--
Yusuke Endoh mame@tsg.ne.jp

=end

#13 - 04/22/2010 03:30 AM - ko1 (Koichi Sasada)

=begin
(2010/04/21 23:55), Yusuke ENDOH wrote::

> どちらも仕様レベルで検討しないといけないことのような気がします。
> まだ結構大変そうなので、1.9.2は見送った方がいいんじゃないかなと
> 思います。早くコメントしていれば、wanabeさんなら余裕で直せたと
> 思うので残念ですが。

□返事が出来ていなくてすみません．最近，色々と余裕がなくて，じっくり考え

て返事が出来ず．

--
// SASADA Koichi at atdot dot net

=end

#14 - 09/14/2010 04:54 PM - shyouhei (Shyouhei Urabe)

- Status changed from Open to Assigned

=begin

=end

#15 - 06/11/2011 02:52 PM - ko1 (Koichi Sasada)

未だに考え中ですが，これは 1.9.3 には仕様変更になるので入りませんよね?

#16 - 06/11/2011 10:23 PM - matz (Yukihiro Matsumoto)

- ruby -v changed from ruby 1.9.2dev (2010-01-13) [i386-mingw32] to -

#17 - 06/11/2011 10:23 PM - matz (Yukihiro Matsumoto)

まつもと ゆきひろです

In message "Re: [ruby-dev:43715] [Ruby 1.9 - Bug #595] Fiber ignores ensure clause"
on Sat, 11 Jun 2011 14:52:03 +0900, Koichi Sasada redmine@ruby-lang.org writes:

|未だに考え中ですが，これは 1.9.3 には仕様変更になるので入りませんよね?

この動作はバグだと考えているので、Yuguiさんが拒絶しない限り、
直せるのであれば1.9.3で直せばよいと思います。間に合わないな
らしょうがない。

#18 - 06/11/2011 10:23 PM - naruse (Yui NARUSE)

RubySpec 的にバグ扱いになってますね。
core/fiber/resume_spec.rb

#19 - 06/26/2011 11:24 PM - nahi (Hiroshi Nakamura)

- Target version changed from 2.0.0 to 1.9.3

#20 - 06/29/2011 11:08 PM - kosaki (Motohiro KOSAKI)

- Status changed from Assigned to Closed

#21 - 03/13/2012 03:06 PM - naruse (Yui NARUSE)

- Status changed from Closed to Assigned

- Target version changed from 1.9.3 to 3.0

#22 - 07/25/2012 07:08 PM - ko1 (Koichi Sasada)

あれ，これ 3.0 でいいんだっけ.

#23 - 07/25/2012 10:24 PM - naruse (Yui NARUSE)

ko1 (Koichi Sasada) wrote:

> あれ，これ 3.0 でいいんだっけ.

3月に話したときに大変そうだから先かなー的な雰囲気のことを仰っていたので、3.0かなぁ、と。
早く直るに越した事はありません。

#24 - 08/05/2012 01:05 PM - wanabe (_ wanabe)

- File ensure_fiber2.patch added

ワナベと申します。

(1) GC で mark と sweep の間に、mark されていない Fiber を対象に
(2) ruby_cleanup 中に、メインスレッドに所属するすべての Fiber を対象に
(3) 子スレッド終了時（vm->living_threads から外されるとき）、所属するすべての Fiber を対象に
の 3 つのタイミングで、throw/catch により ensure 節を実行するパッチを書きました。

[ruby-dev:41035] で遠藤さんがおっしゃっているような「yield 中の Fiber は GC しない」
という手法もやってみたのですが、test/ruby/test_fiber.rb がとても終わりそうにないことや
Fiber のマーク処理の重さや Fiber 自体のメモリ消費量などにより、断念しました。
そのため上記(1)のように、rb_gc_marked_p() という関数が必要になるなど強引な手段を使っています。

また、以下のようにして速度低下を計ってみました。

```
require "benchmark"
GC.start
Benchmark.bm(4) do |x|
tms = Benchmark::Tms.new
10.times do |i|
tms += x.report("   #{i}:") do
30000.times do
Fiber.new{Fiber.yield}.resume
end
end
end
puts " sum:#{tms}"
end
```

素の r36623：2.980000   3.120000   6.100000 (6.107164)
パッチ適用後: 3.580000   3.480000   7.060000 (7.061093)

と、無視できない程度に（約 14%）速度低下してしまいました。
とはいえこれ以上の方法は思いつかないのですが、いかがでしょうか。

#25 - 08/20/2012 05:59 PM - ko1 (Koichi Sasada)

こちら，返事が大変遅くなって済みません．

(2012/08/05 13:05), wanabe (_ wanabe) wrote:

> (1) GC で mark と sweep の間に、mark されていない Fiber を対象に
> (2) ruby_cleanup 中に、メインスレッドに所属するすべての Fiber を対象に
> (3) 子スレッド終了時（vm->living_threads から外されるとき）、所属するすべての Fiber を対象に
> の 3 つのタイミングで、throw/catch により ensure 節を実行するパッチを書きました。
>
> [ruby-dev:41035] で遠藤さんがおっしゃっているような「yield 中の Fiber は GC しない」
> という手法もやってみたのですが、test/ruby/test_fiber.rb がとても終わりそうにないことや
> Fiber のマーク処理の重さや Fiber 自体のメモリ消費量などにより、断念しました。
> そのため上記(1)のように、rb_gc_marked_p() という関数が必要になるなど強引な手段を使っています。

▯とりとめもなく 3 つほど．

▯(1) ですが，ファイナライザのように実行するのはどうかと思っておりました．ただ，その場合，ちょっと管理が大変なんですよね．

▯あと，throw よりは，

#define eKillSignal INT2FIX(0)
#define eTerminateSignal INT2FIX(1)

この辺がいいのかなぁ，と思っていました．いや，むしろここを thorw にしたほうが設計は綺麗かもしれませんね…．現在のの実装は，絶対 catch 出来ない，ってのを目指している感じです．

▯速度ですが，小崎さんが仰っているように，あまり気にしてもしょうがない，
という気もするのですが，例えば ensure 節でひっかける可能性があるか，というチェックをするのは結構効くのではないかと思っています．

▯というのも，殆どの用途はガチで使っているわけじゃないと思うので，ensure な処理を必要とする Fiber は，実は少ないのではないかと，

▯別の案としては，Fiber 生成時に「ensure ちゃんと処理して」フラグを新設し，ガチで使う人（ensure をきっちり動かして欲しい人）はこれでやってね
（参考： http://bugs.ruby-lang.org/issues/6694），というのはどうだろう，
とか思うんですが，ちょっと手抜きしすぎでしょうか．

--
// SASADA Koichi at atdot dot net

#26 - 01/13/2014 10:59 AM - funny_falcon (Yura Sokolov)

What about this ticket?
Guaranteed ensure inside of Fiber and Fiber.raise (as complement for Thread.raise) will be usefull for full coroutine based environment ala python's gevent.

#27 - 10/12/2015 01:16 PM - Eregon (Benoit Daloze)

Could we clarify what is the desired behavior and what prevents it to be implemented?

There is a very old RubySpec about this and I would like to know whether this might be guaranteed in a future release or not.

#28 - 01/31/2017 07:22 AM - ko1 (Koichi Sasada)

- Description updated

□□□□□□□□□□□□□□□

#29 - 01/31/2017 01:52 PM - ko1 (Koichi Sasada)

- Related to Bug #10540: Yielded fibers do not execute ensure blocks added

#30 - 08/29/2018 03:22 AM - normalperson (Eric Wong)

Koichi Sasada wrote:

> Bug #595: Fiber ignores ensure clause
> http://redmine.ruby-lang.org/issues/595

What's the status of this? (I don't understand Japanese)

It will be difficult/unsafe to use auto-fiber/Thread::Coro
[Feature #13618] without this

Is it a performance problem holding it back?
We have ccan/list nowadays, so maybe I will try to make it fast.

#31 - 08/29/2018 08:12 AM - normalperson (Eric Wong)

Eric Wong wrote:

> It will be difficult/unsafe to use auto-fiber/Thread::Coro
> [Feature #13618] without this

I think I can get around this by making iom a GC root,
so all auto-yielded Fibers get marked.

#32 - 11/08/2018 07:42 AM - normalperson (Eric Wong)

> Eric Wong wrote:
>
> > It will be difficult/unsafe to use auto-fiber/Thread::Coro
> > [Feature #13618] without this
>
> I think I can get around this by making iom a GC root,
> so all auto-yielded Fibers get marked.

Btw, I worked around this by making all auto-Fibers markable
from rb_thread_t.

However, I'm also working on making all sleeping functions
(native_sleep/rb_wait_for_single_fd/rb_thread_fd_select) method
perform auto-Fiber scheduling.

Unfortunately, that still interacts badly when people use
regular ("manual") Fibers because I/O scheduling depend on

rb_ensure heavily... So *shrug* I'll have to think of something...

#33 - 11/14/2018 10:04 PM - normalperson (Eric Wong)

> However, I'm also working on making all sleeping functions (native_sleep/rb_wait_for_single_fd/rb_thread_fd_select) method perform auto-Fiber scheduling.
>
> Unfortunately, that still interacts badly when people use regular ("manual") Fibers because I/O scheduling depend on rb_ensure heavily... So *shrug* I'll have to think of something...

Not auto-calling fiber_switch from regular Fiber seems to do the trick. Need to avoid infinite loops from regular Fibers, though...

#34 - 11/22/2018 12:38 AM - ioquatix (Samuel Williams)

In a GC language, it's impossible to ensure this, unless you want the GC to invoke some functionality.

I suggest adding something like Fiber#stop which causes internally Fiber#resume to raise exception.

It should be up to user code to deal with this IMHO, e.g.

```
fib = Fiber.new{
  begin
    Fiber.yield :ok
  ensure
    puts "should be print out"
  end
}
begin
  p fib.resume
ensure
  fib.stop
end
```

I believe there is no logical solution to avoid this issue except making it explicit of consumer of fiber. It's the same reason why we should do the following:

```
f = File.open
begin
  f.write "..."
ensure
  f.close
end
```

#35 - 12/20/2018 01:11 AM - ioquatix (Samuel Williams)

- Related to Feature #10344: [PATCH] Implement Fiber#raise added

#36 - 12/20/2018 01:15 AM - ioquatix (Samuel Williams)

- Assignee changed from ko1 (Koichi Sasada) to ioquatix (Samuel Williams)

While it's not possible unless we invoke functionality from the GC, if/when we merge https://bugs.ruby-lang.org/issues/10344 it should become simple to do the following

```
while fiber.alive?
    fiber.raise(Exception, "Time to die")
end
```

This would guarantee all ensure blocks are executed.

In theory you could implement fiber.kill which does this, and then invoke that from the GC in the finaliser.. but might be unexpected for user.

#37 - 12/20/2018 04:19 AM - ioquatix (Samuel Williams)

- Target version set to 2.7

After discussion, we decided that:

- Doing this in the GC is probably a bad idea.
- Ruby's general model for resource management is explicit, e.g. f = File.open; f.close or File.open do |f| ... end. The same model should apply to Fiber.
- We can implement something like Fiber.kill or Fiber.close which essentially invokes Fiber.raise(Exception) until the Fiber is dead.

We will test these ideas during 2.7 release. If there are any further thoughts about this approach, please feel free to add them here.

#38 - 01/14/2019 07:32 AM - naruse (Yui NARUSE)

- Target version deleted (2.7)

Target version is used by release engineering; don't use this as just a goal.

#39 - 12/29/2019 10:37 AM - hsbt (Hiroshi SHIBATA)

- Tags set to fiber

Files

ensure_fiber.patch	2.12 KB	01/13/2010	wanabe (_ wanabe)
ensure_fiber2.patch	7.57 KB	08/05/2012	wanabe (_ wanabe)