Ruby master - Bug #1501

Enumerator.new { }.take(1).inject(&:+) causes stack overflow

05/21/2009 10:54 PM - mame (Yusuke Endoh)

Status:	Closed	
Priority:	Normal	
Assignee:	ko1 (Koichi Sasada)	
Target version:	2.0.0	
ruby -v:	ruby 1.9.2dev (2009-05-19 trunk 23489) [i686-linux]	**Backport:**

Description

=begin
遠藤です。

以下のようにすると、謎の SystemStackError が出てきます。

$ ./ruby -ve 'Enumerator.new { }.take(1).inject(&:+)'
ruby 1.9.2dev (2009-05-19 trunk 23489) [i686-linux]
-e:1:in proc': stack level too deep (SystemStackError)
from -e:1:into_proc'
from -e:1:in `'

1.9.1-p0 でも同じでした。

調べてみたところ、現在のブロックの cfp->lfp[0] が指すブロックが
自分自身になっているようです (正確には、他のブロックを解して
間接的に循環しているようでした) 。
そのせいで、vm.c の rb_vm_make_proc と vm_make_proc_from_block が
相互に呼び出しあって無限再帰しているようです。

cfp->lfp[0] がおかしくなる原因はたぶん enumerator.c で、

- yielder_new が rb_iterate(yielder_new_i, ...) を呼ぶ
- passed_block が設定される
- yielder_new_i は Ruby レベルのメソッドを呼ばずに終了する
- passed_block が設定されたまま、yielder_new が終わり、YARV の eval ループに戻る
- passed_block がどこか変なところで cfp->lfp[0] に代入される

という流れになっているように感じました (yielder がどんなものかは
よくわかっていません) 。

そこで、以下のように、yielder_new_i で proc メソッドを呼ぶことで
バグは直りました。

Index: enumerator.c
===
--- enumerator.c (revision 23508)
+++ enumerator.c (working copy)
@@ -720,7 +720,7 @@
static VALUE
yielder_new_i(VALUE dummy)
{

- return yielder_init(yielder_allocate(rb_cYielder), rb_block_proc());
- return yielder_init(yielder_allocate(rb_cYielder), rb_funcall(Qnil, rb_intern("proc"), 0)); }

static VALUE

ささださんがいいと言ってくれたら (もしくは何も言わないなら)
コミットしようと思います。

上に関係して、Ruby のソースコードには以下のアサーションが暗黙に

存在すると思ったのですが、正しいでしょうか。

- passed_block が設定されたら、eval ループに戻る前に Ruby レベルの メソッドを呼んで passed_block を回収させないといけない
- rb_iterate や rb_block_call の第一引数に渡される関数は、その中で Ruby レベルのメソッドを呼ばないといけない

後者は C API に関わる話なので、正しいようなら README.EXT に書き 加えた方がいいと思います。というか書き加えようと思います。

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

History

#1 - 05/21/2009 10:58 PM - mame (Yusuke Endoh)

- Category set to core

- Status changed from Open to Assigned

- Assignee set to ko1 (Koichi Sasada)

- Priority changed from 3 to Normal

- Target version set to 2.0.0

- ruby -v set to ruby 1.9.2dev (2009-05-19 trunk 23489) [i686-linux]

=begin

=end

#2 - 05/22/2009 05:40 AM - ko1 (Koichi Sasada)

=begin
□ささだです.

Yusuke ENDOH wrote::

> そこで、以下のように、yielder_new_i で proc メソッドを呼ぶことで
> バグは直りました。

□似たような話なんですが，そもそも rb_iterate を使う場面じゃなさそうな.

Index: enumerator.c
===
--- enumerator.c (リビジョン 23496)
+++ enumerator.c (作業コピー)
@@ -718,12 +718,6 @@
}

static VALUE
-yielder_new_i(VALUE dummy)
-{

- return yielder_init(yielder_allocate(rb_cYielder), rb_block_proc()); -} - -static VALUE yielder_yield_i(VALUE obj, VALUE memo, int argc, VALUE *argv) { return rb_yield_values2(argc, argv); @@ -732,7 +726,7 @@ static VALUE yielder_new(void) {
- return rb_iterate(yielder_new_i, (VALUE)0, yielder_yield_i, (VALUE)0);
- return yielder_init(yielder_allocate(rb_cYielder), rb_proc_new(yielder_yield_i, 0)); }

/*

> 上に関係して、Ruby のソースコードには以下のアサーションが暗黙に
> 存在すると思ったのですが、正しいでしょうか。
>
> - passed_block が設定されたら、eval ループに戻る前に Ruby レベルの メソッドを呼んで passed_block を回収させないといけない

yes.

- rb_iterate や rb_block_call の第一引数に渡される関数は、その中で Ruby レベルのメソッドを呼ばないといけない

yes.

> 後者は C API に関わる話なので、正しいようなら README.EXT に書き
> 加えた方がいいと思います。というか書き加えようと思います。

□これは，実は 1.8 以前との非互換の問題だったんですよね．

□これを真面目に解決しようとすると，このようなレアケースのためにブロック
をチェックする全てのコードに passed_block をチェックするように変更する必
要が出てきて，ちょっと現実的じゃないのです．

＊フレームをその時だけいじっちゃう，という回避手段があるような，
□嫌なところに波及しそうな．

□例えば，rb_iterate は obsolete にしてしまって（制限付きで存在），
rb_block_call を使うようにして下さい，とドキュメントするのはどんなもんで
しょうか．

--
// SASADA Koichi at atdot dot net
// 久しぶりに Ruby のソースコードをいじった

=end

#3 - 05/22/2009 07:41 AM - matz (Yukihiro Matsumoto)

=begin
まつもと ゆきひろです

In message "Re: [ruby-dev:38521] Re: [Bug:1.9] Enumerator.new { }.take(1).inject(&:+) causes stack overflow"
on Fri, 22 May 2009 05:39:46 +0900, SASADA Koichi ko1@atdot.net writes:

|□これは，実は 1.8 以前との非互換の問題だったんですよね．
|
|□これを真面目に解決しようとすると，このようなレアケースのためにブロック
|をチェックする全てのコードに passed_block をチェックするように変更する必
|要が出てきて，ちょっと現実的じゃないのです．
|
|＊フレームをその時だけいじっちゃう，という回避手段があるような，
|□嫌なところに波及しそうな．
|
|
|□例えば，rb_iterate は obsolete にしてしまって（制限付きで存在），
|rb_block_call を使うようにして下さい，とドキュメントするのはどんなもんで
|しょうか．

まあ、それはそれで構いません。

=end

#4 - 05/22/2009 10:00 AM - mame (Yusuke Endoh)

=begin
遠藤です。

2009/05/22 5:39 SASADA Koichi ko1@atdot.net:

> Yusuke ENDOH wrote::
>
> > そこで、以下のように、yielder_new_i で proc メソッドを呼ぶことで
> > バグは直りました。
>
> □似たような話なんですが，そもそも rb_iterate を使う場面じゃなさそうな．

確かに。ささださんのパッチでも直ることを確認しました。

> 例えば，rb_iterate は obsolete にしてしまって（制限付きで存在），
> rb_block_call を使うようにして下さい，とドキュメントするのはどんなもんで
> しょうか．

なるほど、rb_block_call は必ずメソッドを呼び出すから大丈夫
なんですね。

ドキュメント修正案のパッチです。

Index: README.EXT
===
--- README.EXT (revision 23508)
+++ README.EXT (working copy)
@@ -1159,11 +1159,22 @@

** Control Structure

- VALUE rb_iterate(VALUE (*func1)(), void *arg1, VALUE (*func2)(), void *arg2)
- VALUE rb_block_call(VALUE recv, ID mid, int argc, VALUE * argv,
- VALUE (*func) (ANYARGS), VALUE data2)

+Calls a method on the recv, with the method name specified by the
+symbol mid, supplying func as the block. func will receive the
+value from yield as the first argument, data2 as the second, and
+argc/argv as the third/fourth arguments.
+

- [OBSOLETE] VALUE rb_iterate(VALUE (*func1)(), void *arg1, VALUE (*func2)(), void *arg2) + Calls the function func1, supplying func2 as the block. func1 will be called with the argument arg1. func2 receives the value from yield as the first argument, arg2 as the second argument. + +When rb_iterate is used in 1.9, func1 has to call some Ruby-level method. +This function is obsolete since 1.9; use rb_block_call instead.

VALUE rb_yield(VALUE val)

Index: README.EXT.ja
===
--- README.EXT.ja (revision 23508)
+++ README.EXT.ja (working copy)
@@ -1258,11 +1258,22 @@

** 制御構造

-VALUE rb_iterate(VALUE (*func1)(), VALUE arg1, VALUE (*func2)(), VALUE arg2)
+VALUE rb_block_call(VALUE obj, ID mid, int argc, VALUE * argv,

- [redacted]

- funcをブロックとして設定し, objをレシーバ, argcとargvを引

- 数としてmidメソッドを呼び出す. funcは第一引数にyieldされた

- 値, 第二引数にdata2, 第三, 第四引数にargcとargvを受け取る.
  +
  +[OBSOLETE] VALUE rb_iterate(VALUE (*func1)(), VALUE arg1, VALUE
  (*func2)(), VALUE arg2)
  +
  func2をブロックとして設定し, func1をイテレータとして呼ぶ.
  func1には arg1が引数として渡され, func2には第1引数にイテレー
  タから与えられた値, 第2引数にarg2が渡される.
  +

- 1.9でrb_iterateを使う場合は, func1の中でRubyレベルのメソッド

- を呼び出さなければならない.

- 1.9でobsoleteとなった. 代わりにrb_block_callが用意された.

VALUE rb_yield(VALUE val)

--
Yusuke ENDOH mame@tsg.ne.jp

=end

#5 - 06/15/2009 06:36 AM - ko1 (Koichi Sasada)

=begin
[redacted]ささだです.

□返事が随分遅くなってしまってすみません.

Yusuke ENDOH wrote::

> なるほど、rb_block_call は必ずメソッドを呼び出すから大丈夫
> なんですね。
>
> ドキュメント修正案のパッチです。

□よろしいかと思いました.

--
// SASADA Koichi at atdot dot net

=end

#6 - 07/14/2009 01:17 AM - mame (Yusuke Endoh)

- Status changed from Assigned to Closed

- % Done changed from 0 to 100

=begin
Applied in changeset r24094.
=end