Ruby master - Bug #4352

[patch] Fix eval(s, b) backtrace; make eval(s, b) consistent with eval(s)

02/01/2011 11:31 AM - quix (James M. Lawrence)

Status:	Closed		
Priority:	Normal		
Assignee:	matz (Yukihiro Matsumoto)		
Target version:	3.0		
ruby -v:	ruby 1.9.3dev (2011-02-01 trunk 30751) [i386-darwin9.8.0]	**Backport:**	

Description

```
=begin
def ex_message
begin
yield
rescue => e
p e.message
end
end

ex_message { eval('raise') }
ex_message { eval('raise', binding) }

eval('def f ; end')
p method(:f).source_location
eval('def g ; end', binding)
p method(:g).source_location
```

Without patch:

```
"(eval):1:in `block in ': "
""
["(eval)", 1]
["eval_test.rb", 14]
```

With patch:

```
"(eval):1:in block in <main>': "
"(eval):1:inblock in ': "
["(eval)", 1]
["(eval)", 1]
```

Knowing the line of an error inside eval is useful. Passing a binding shouldn't discard that information. Present behavior is even wrong: there's no line 10 in this file.

```
eval %{

# .. code ...
raise

}, binding
```

Without patch:

```
/Users/jlawrence/tmp/raiser.rb:10:in <main>': unhandled exception
from /Users/jlawrence/tmp/raiser.rb:7:ineval'
from /Users/jlawrence/tmp/raiser.rb:7:in `'
```

With patch:

```
/Users/jlawrence/tmp/raiser.rb:7:in eval': (eval):4:in':  (RuntimeError)
from /Users/jlawrence/tmp/raiser.rb:7:in eval'
```

from /Users/jlawrence/tmp/raiser.rb:7:in'
=end

Related issues:

Related to Ruby master - Bug #4487: require_relative fails in an eval'ed file	**Closed**
Related to Ruby master - Bug #4379: [patch] eval(s, b, "(eval)", n) discards ...	**Closed**

Associated revisions

Revision e9e17cbc - 08/02/2019 02:17 PM - mame (Yusuke Endoh)

parse.y: make a warning for **FILE** in eval by default

[Bug #4352]

Revision c6837638 - 08/02/2019 10:44 PM - mame (Yusuke Endoh)

Use source_location instead of eval(**FILE**,binding) in Binding#irb

e9e17cbc051e894dfd27eda5feca2939f65552db (enabling the warning by
default) caused a warning in test-spec:

```
/data/chkbuild/tmp/build/20190802T213005Z/ruby/spec/ruby/core/binding/irb_spec.rb
Binding#irb
- creates an IRB session with the binding in scope/data/chkbuild/tmp/build/20190802T213005Z/ruby/spec/ruby/cor
e/binding/fixtures/irb.rb:3: warning: __FILE__ in eval may not return location in binding; use Binding#source_
location instead
```

https://rubyci.org/logs/rubyci.s3.amazonaws.com/debian/ruby-master/log/20190802T213005Z.log.html.gz

ref: [Bug #4352]

Revision 0eeed5bc - 01/04/2020 04:13 AM - jeremyevans (Jeremy Evans)

Make eval(code, binding) use (eval) as **FILE** and 1 as **LINE**

This removes the warning that was added in
3802fb92ff8c83eed3e867db20f72c53932f542d, and switches the behavior
so that the eval does not use the binding's **FILE** and **LINE**
implicitly.

Fixes [Bug #4352]

History

#1 - 02/02/2011 05:35 AM - quix (James M. Lawrence)

I came across this issue when I noticed that source_location gives non-useful info when a binding is passed to eval. Thus the patch fixes two somewhat different problems: eval backtrace and source_location. If changing the backtrace is inappropriate then a separate bug for source_location should be filed.

#2 - 02/02/2011 12:46 PM - mame (Yusuke Endoh)

Hi,

2011/2/1 James M. Lawrence redmine@ruby-lang.org:

> Knowing the line of an error inside eval is useful. Passing a binding
> shouldn't discard that information.

I understand you, but the behavior is intended.

A binding also has its own information of filename and lineno.
Some people ([ruby-core:28307] [ruby-dev:38767]) think that binding's
lineno information is more important than eval's information, and that
eval shouldn't discard the binding's informantion.

```
# foo.rb
eval("p [__FILE__, __LINE__]", binding)  #=> expected: ["foo.rb", 2]
```

In addition, the behavior is compatible to 1.8.

> Present behavior is even wrong:
> there's no line 10 in this file.

```
eval %{

  # .. code ...
  raise


}, binding
```

Without patch:

```
/Users/jlawrence/tmp/raiser.rb:10:in `<main>': unhandled exception
       from /Users/jlawrence/tmp/raiser.rb:7:in `eval'
       from /Users/jlawrence/tmp/raiser.rb:7:in `<main>'
```

With patch:

```
/Users/jlawrence/tmp/raiser.rb:7:in `eval': (eval):4:in `<main>': ?(RuntimeError)
       from /Users/jlawrence/tmp/raiser.rb:7:in `eval'
       from /Users/jlawrence/tmp/raiser.rb:7:in `<main>'
```

It is indeed confusing, but it can be understood as follows:

1) Here are actual linenos. The binding has its own lineno at which the method is called:

```
1: eval %{
2:
3:   # .. code ...
4:   raise
5:
6:
7: }, binding
```

2) binding virtually overwrites linenos so that the eval'ing code starts with the binding's own lineno (that is, Line 7):

```
1: eval %{           # ( 7)
2:                   # ( 8)
3:   # .. code ...   # ( 9)
4:   raise           # (10)
5:                   # (11)
6:                   # (12)
7: }, binding        # (13)
```

3) an exception is raised at (virtual) Line 10.

You can exploit this behavior to know the lineno more directly:

```
 1: # foo.rb
 2: b, s = binding, %{
 3:
 4:
 5:   # .. code ...
 6:   raise           # <- HERE!!
 7:
 8:
 9: }, b
10: eval s, b #=> foo.rb:6:in `<main>': unhandled exception
```

I guess eval should have received binding as the first argument ;-)

```
eval binding, %{
  ...
}
```

--
Yusuke Endoh mame@tsg.ne.jp

#3 - 02/02/2011 02:36 PM - quix (James M. Lawrence)

Thank you for that detailed explanation. The problem for me is the connection to source_location, which should be usable by tools.

Shouldn't source_location give the file and line of a method or block

definition? If so then source_location is bugged when the binding argument is passed.

The seemingly simplest solution is to make the backtrace and source_location consistent. It only takes a 4-line patch.

I might have thought otherwise if virtual line numbers served some human purpose, but I just see them as confusing. Tools are even more confused.

Should a new bug for source_location be created?

#4 - 02/03/2011 12:48 AM - mame (Yusuke Endoh)

Hi,

2011/2/2 James M. Lawrence redmine@ruby-lang.org:

> Thank you for that detailed explanation. The problem for me is the connection to source_location, which should be usable by tools.

What kind of tools are you talking about?
Even if a binding location is discarded, we can still fake __FILE__ and __LINE__ without using a binding:

```
eval <<-END, nil, "/etc/passwd", 1
  def foo
  end
END
p method(:foo).source_location  #=> ["/etc/passwd", 1]
```

So, source_location user should know and accept the fact that the information is not trustable.
Why do you think only a binding location as a problem?

--
Yusuke Endoh mame@tsg.ne.jp

#5 - 02/03/2011 11:23 AM - quix (James M. Lawrence)

> Why do you think only a binding location as a problem?

The initial problem I encountered was

```
eval %{def g ; end}, nil, "(eval)", 99
p method(:g).source_location #=> ["(eval)", 99]

eval %{def f ; end}, binding, "(eval)", 99
p method(:f).source_location #=> ["prob.rb", 1]
```

I needed source_location to be ["(eval)", 99] in the latter case, which is solved by the patch. This could be also be done by making eval recognize the difference between an empty file argument and an "(eval)" file argument, however the problem seemed more fundamental.

eval should not implicitly slurp file/line info from the binding. Creating a "dishonest" backtrace is something that should be done explicitly by the user, as in

```
eval(s, b, *b.source_location)
```

In the last the example in the original post, the existence of raiser.rb:10 is an outright falsehood. Pointing to a nonexistent line number should not be the default behavior of eval(s, b).

Since source_location claims to be "the ruby source filename and line number containing this proc", I was thinking that source_location could give the "true" location, ignoring the file/line "lies" passed to eval. But this was wrong--indeed I want the "lies" to be reflected in source_location. My problem is with eval implicitly grabbing file/line from the binding, which is just what the patch solves.

As mentioned above, this would be nice too:

```
class Binding
  def source_location
    eval "[__FILE__, __LINE__]"
  end
end

diff --git a/proc.c b/proc.c
index 7b1d147..0042caf 100644
--- a/proc.c
+++ b/proc.c
@@ -305,6 +305,24 @@ binding_clone(VALUE self)
     return bindval;
 }

+/*
+ * call-seq:
+ *    binding.source_location  -> [String, Fixnum]
+ *
+ * Returns the ruby source filename and line number associated with
+ * this binding.
+ */
+static VALUE
+binding_source_location(VALUE self)
+{
+    rb_binding_t *src;
+    VALUE loc[2];
+    GetBindingPtr(self, src);
+    loc[0] = rb_str_dup_frozen(src->filename);
+    loc[1] = INT2FIX(src->line_no);
+    return rb_ary_new4(2, loc);
+}
+
 VALUE
 rb_binding_new(void)
 {
@@ -2227,6 +2245,7 @@ Init_Binding(void)
     rb_undef_method(CLASS_OF(rb_cBinding), "new");
     rb_define_method(rb_cBinding, "clone", binding_clone, 0);
     rb_define_method(rb_cBinding, "dup", binding_dup, 0);
+    rb_define_method(rb_cBinding, "source_location", binding_source_location, 0);
     rb_define_method(rb_cBinding, "eval", bind_eval, -1);
     rb_define_global_function("binding", rb_f_binding, 0);
 }
```

#6 - 02/03/2011 01:19 PM - mame (Yusuke Endoh)

Hi,

2011/2/3 Rocky Bernstein rockyb@rubyforge.org:

> See also
> http://ola-bini.blogspot.com/2008/01/ruby-antipattern-using-eval-without.html.
> It is called an "anti-pattern" there which I guess is used in a derogatory
> fashion.

I'd like to positively call it a "best practice" :-)

> A place where setting the file and line is used is in template systems like
> merb or erb where the file the user works on is not a Ruby file but a
> template file that expands to a Ruby file. In that situation, disregarding
> the expanded Ruby file is convenient since there can only be one location
> attached in the Ruby implementation.
> However I don't see why a templating system couldn't also provide an
> expanded Ruby file for debugging problems much as one can get the
> C-preprocessed output for a C program when one wants.

I'm not sure that I could understand you.
Are you saying that erb users should debug their erb code by looking
the erb-generated Ruby code? I don't think that it is user-friendly.

FYI, erb offers ERB#src which provides a generated Ruby code. However,

I don't want to encourage users to read and debug the generated code.

> But shouldn't we try to address the location problem head on in the Ruby implementation? I suspect it too late to try to change Ruby MRI 1.x, but perhaps in Ruby 2.x some thought could be given to how to address.?

Yes, it is good to improve the spec in 2.0.
Before that, we must first check the use case.
For creating what kind of tools, what kind of information do you need?

> If the fact that source_location is not trustable is a a concern, then perhaps setting the file and line parameters on eval can be disallowed at some level greater than 0 and less than 4 which is where eval() is disallowed totally.?

We should not ignore erb-like use case. Just prohibiting the file and line parameters is too strict. Unless Ruby provides an alternative feature, like #line of C-preprocessor.

BTW, is it ok for you that source_location returns ["(eval)", 1]?
It does not allow to distinguish whether it is executed in Kernel#eval or the source code file is named "(eval)".
"-" (for stdin) and "-e" (for -e option) seem to have the same problem.

--
Yusuke Endoh mame@tsg.ne.jp

#7 - 02/03/2011 01:19 PM - mame (Yusuke Endoh)

Hi,

2011/2/3 James M. Lawrence redmine@ruby-lang.org:

> The initial problem I encountered was
>
> ```
> eval %{def g ; end}, nil, "(eval)", 99
> p method(:g).source_location #=> ["(eval)", 99]
>
> eval %{def f ; end}, binding, "(eval)", 99
> p method(:f).source_location #=> ["prob.rb", 1]
> ```

Hmm. It is indeed irritating for Kernel#eval to prefer implicit filename of a binding to explicitly-specified filename.
This is because the current implementation is uncool, like:

```
def eval(src, binding, filename = "(eval)", lineno = 1)
  filename = binding.filename if filename == "(eval)"
  ...
end
```

Kernel#eval uses a string "(eval)" when the filename is not specified explicitly. And then, it replaces it with implicit filename of binding when the given filename is equal to "(eval)" with string comparison.

Even so, I hesitate to change the behavior in 1.9.x for compatibility reason.

> Since source_location claims to be "the ruby source filename and line number containing this proc", I was thinking that source_location could give the "true" location, ignoring the file/line "lies" passed to eval.

Honestly, I understand your expectation. It is of course acceptable to fix it in 2.0. But it would require a major modification because the current implementation itself does NOT know the "true" location. The information is discarded at the parse time.

--
Yusuke Endoh mame@tsg.ne.jp

#8 - 02/03/2011 01:57 PM - quix (James M. Lawrence)

Yusuke Endoh:

> > Since source_location claims to be "the ruby source filename and line
> > number containing this proc", I was thinking that source_location
> > could give the "true" location, ignoring the file/line "lies" passed
> > to eval.
>
> Honestly, I understand your expectation. It is of course acceptable to
> fix it in 2.0. But it would require a major modification because the
> current implementation itself does NOT know the "true" location. The
> information is discarded at the parse time.

Did you misunderstand? It's not my expectation--the next sentence said
it was wrong, and indeed I rely on the current behavior.

My expectation is that overriding file/line for eval can only be done
explicitly, never implicitly through the binding. That's what the
patch does. What did you think of my argument for that?

#9 - 02/04/2011 09:21 AM - mame (Yusuke Endoh)

Hi,

2011/2/3 James M. Lawrence redmine@ruby-lang.org:

> Yusuke Endoh:
>
> > > Since source_location claims to be "the ruby source filename and line
> > > number containing this proc", I was thinking that source_location
> > > could give the "true" location, ignoring the file/line "lies" passed
> > > to eval.
> >
> > Honestly, I understand your expectation. It is of course acceptable to
> > fix it in 2.0. But it would require a major modification because the
> > current implementation itself does NOT know the "true" location. The
> > information is discarded at the parse time.
>
> Did you misunderstand? It's not my expectation--the next sentence said
> it was wrong, and indeed I rely on the current behavior.
>
> My expectation is that overriding file/line for eval can only be done
> explicitly, never implicitly through the binding.

I see. Sorry for my misunderstanding.
But anyway, it is difficult to meet your expectation in 1.9.
Inheriting file/line from a binding is actually intended, not a
bug, even though it is confusing.
We cannot change any spec in 1.9, unless it is considered a bug.

However, it might be considered a bug for Kernel#eval to ignore
an explicitly specified "(eval)", I think.

Does that answer you?

--
Yusuke Endoh mame@tsg.ne.jp

#10 - 02/08/2011 03:22 AM - quix (James M. Lawrence)

Yusuke Endoh:

> However, it might be considered a bug for Kernel#eval to ignore
> an explicitly specified "(eval)", I think.
>
> Does that answer you?

Yes, thanks. Redmine doesn't have a category for bugs that are fixed
by a feature request :). As Rocky explains there are some deeper

issues involved, but it still amounts to a feature request. I've filed
a new bug for "(eval)" [ruby-core:35139]. Sorry for the confusion.

#11 - 06/26/2011 05:35 PM - naruse (Yui NARUSE)

- Status changed from Open to Assigned

- Assignee set to mame (Yusuke Endoh)

#12 - 07/05/2011 12:50 PM - mame (Yusuke Endoh)

- Assignee changed from mame (Yusuke Endoh) to matz (Yukihiro Matsumoto)

Hi, matz

This ticket includes some design concerns.

The source_location seems important information not only for
human users, but also for tools, such as a debugger, lint,
coverage measurement, etc.
James and Rocky seem to be interested in the aspect of tools,
and dislike eval, which fakes the information.

1) Is the feature that fakes source_location really needed?
I guess that the feature is designed for erb-like application,
but C-style "#line" may be better.

2) What source_location should be used when only binding is
given?

```
eval("__FILE__", TOPLEVEL_BINDING) #=> "<main>" or "(eval)"?
```

Some people expect "<main>" [ruby-dev:38767], and others expect
"(eval)". [ruby-core:35027]

3) Which source_location should be used when binding and
explicit source_location are both given?

```
eval("__FILE__", TOPLEVEL_BINDING, "foo") #=> "<main>" or "foo"?
```

Currently "foo" is returned.

4) Should 3) be applied even if "(eval)" is given explicitly?

```
eval("__FILE__", TOPLEVEL_BINDING, "(eval)") #=> "<main>" or "(eval)"?
```

Currently "<main>" is returned. (7th Dec. 2017: now it returns "(eval)".)

5) Shouldn't ruby have two-type information, one for human and
one for tools? The former is used for backtrace. The latter
is used for tools. eval can fake only the former information.

6) If 5) is accepted, which information should be used by
require_relative? See #4487.

Thanks,

Yusuke Endoh mame@tsg.ne.jp

#13 - 02/17/2013 03:22 PM - ko1 (Koichi Sasada)

- Target version changed from 2.0.0 to 2.1.0

Time up for 2.0.0.

Matz, could you check this ticket?

#14 - 01/30/2014 06:16 AM - hsbt (Hiroshi SHIBATA)

- Target version changed from 2.1.0 to 2.2.0

#15 - 12/12/2017 09:05 AM - matz (Yukihiro Matsumoto)

- Target version changed from 2.2.0 to 2.6

I am sorry I missed this issue for a long time.

I agree with the rationale behind the proposal. I am slightly concerned about incompatibility.
So we need to experiment to measure how big the compatibility issue after changing the behavior.
I expect the impact is small (but my expectation fails often).

Matz.

#16 - 12/12/2017 11:41 PM - mame (Yusuke Endoh)

- Assignee changed from matz (Yukihiro Matsumoto) to mame (Yusuke Endoh)

Thank you, matz.
I'll try this change after 2.5 is released.

#17 - 12/25/2017 09:44 AM - mame (Yusuke Endoh)

- Assignee changed from mame (Yusuke Endoh) to matz (Yukihiro Matsumoto)

matz (Yukihiro Matsumoto) wrote:

> I agree with the rationale behind the proposal. I am slightly concerned about incompatibility.
> So we need to experiment to measure how big the compatibility issue after changing the behavior.
> I expect the impact is small (but my expectation fails often).

I tried the change, and found some projects actually use the idiom, eval("[__FILE__, __LINE__]", binding), to fetch the source location. See #14230 in detail. What should we do?

#18 - 12/25/2017 06:14 PM - naruse (Yui NARUSE)

- Target version deleted (2.6)

#19 - 12/26/2017 05:39 PM - mame (Yusuke Endoh)

- Target version set to 2.7

We discussed this issue at today's Ruby committer's meeting. We can't change the behavior soon because of compatibility issue, so we need a transition path.

After r61483, a warning is now printed when eval receives only binding and the code includes __FILE__ or __LINE__.

```
$ ./miniruby -w -e 'eval("[__FILE__, __LINE__]", binding)'
-e:1: warning: __FILE__ in eval may not return location in binding; use Binding#source_location instead
-e:1: warning: __LINE__ in eval may not return location in binding; use Binding#source_location instead
```

The return value will be changed from binding's source location to eval's default, i.e., "(eval)" and 1. If you need the traditional behavior, please pass the location information explicitly, like: eval("[__FILE__, __LINE__]", binding, *binding.source_location).

#20 - 05/18/2018 11:28 PM - xuhuan587 ( )

- File deleted (eval.patch)

#21 - 07/31/2019 06:41 PM - jeremyevans0 (Jeremy Evans)

- File eval-binding-file-line-4352.patch added

Attached is a patch to remove the warning and change the behavior, as well as update the tests and specs so that eval("[__FILE__, __LINE__]", binding) returns ["(eval)", 1].

#22 - 08/02/2019 02:17 PM - mame (Yusuke Endoh)

- Status changed from Assigned to Closed

Applied in changeset git|e9e17cbc051e894dfd27eda5feca2939f65552db.

---

parse.y: make a warning for **FILE** in eval by default

[Bug #4352]

#23 - 08/02/2019 02:21 PM - mame (Yusuke Endoh)

- Target version changed from 2.7 to 3.0

- Status changed from Closed to Open

jeremyevans0 (Jeremy Evans) , thank you for reminding this issue.

The warning is printed only on the verbose mode in 2.6. It would be good to enable the warning by default in 2.7.
I tentatively changed at e9e17cbc051e894dfd27eda5feca2939f65552db. How about applying your patch for 3.0?
If you think your patch should be applied soon, I'm not so strongly against it.

#24 - 08/02/2019 03:11 PM - jeremyevans0 (Jeremy Evans)

mame (Yusuke Endoh) wrote:

> jeremyevans0 (Jeremy Evans) , thank you for reminding this issue.
>
> The warning is printed only on the verbose mode in 2.6. It would be good to enable the warning by default in 2.7.
> I tentatively changed at e9e17cbc051e894dfd27eda5feca2939f65552db. How about applying your patch for 3.0?
> If you think your patch should be applied soon, I'm not so strongly against it.

mame (Yusuke Endoh) I think applying my patch for 3.0 is fine. I only suggested it for 2.7 because that is what the issue target was.

#25 - 08/02/2019 10:48 PM - mame (Yusuke Endoh)

- Status changed from Open to Closed

Applied in changeset git|c6837638657429034825d5c9e2a29c340898afb8.

Use source_location instead of eval(**FILE**,binding) in Binding#irb

e9e17cbc051e894dfd27eda5feca2939f65552db (enabling the warning by
default) caused a warning in test-spec:

```
/data/chkbuild/tmp/build/20190802T213005Z/ruby/spec/ruby/core/binding/irb_spec.rb
Binding#irb
- creates an IRB session with the binding in scope/data/chkbuild/tmp/build/20190802T213005Z/ruby/spec/ruby/cor
e/binding/fixtures/irb.rb:3: warning: __FILE__ in eval may not return location in binding; use Binding#source_
location instead
```

https://rubyci.org/logs/rubyci.s3.amazonaws.com/debian/ruby-master/log/20190802T213005Z.log.html.gz

ref: [Bug #4352]

#26 - 08/12/2019 11:06 PM - jeremyevans0 (Jeremy Evans)

- Status changed from Closed to Open

#27 - 01/04/2020 04:13 AM - jeremyevans (Jeremy Evans)

- Status changed from Open to Closed

Applied in changeset git|0eeed5bcc5530edb0af2af2ccff09d067c59e8f9.

Make eval(code, binding) use (eval) as **FILE** and 1 as **LINE**

This removes the warning that was added in
3802fb92ff8c83eed3e867db20f72c53932f542d, and switches the behavior
so that the eval does not use the binding's **FILE** and **LINE**
implicitly.

Fixes [Bug #4352]

Files

eval-binding-file-line-4352.patch	11 KB	07/31/2019	jeremyevans0 (Jeremy Evans)