Ruby trunk - Bug #7214

Ruby 2.0 breaks support for some debugging tools

10/25/2012 07:46 PM - banister (john mair)

Status:	Third Party's Issue		
Priority:	Normal		
Assignee:	ko1 (Koichi Sasada)		
Target version:	2.0.0		
ruby -v:	ruby 2.0.0dev (2012-10-18 trunk 37260) [x86_64-linux]	**Backport:**	

Description

Notably the "Pry" debugger breaks, and (though i haven't checked) I'm assuming the 'debugger' project as well. The reason for the breakages (as far as i can see) is that the rb_vm_make_env_object function is now hidden. In the comments for 1.9.3's vm.c it was stated an alternative API for rb_vm_make_env_object (see https://github.com/ruby/ruby/blob/ruby_1_9_3/vm.c#L53-60) would be provided, but I have been unable to find one.

Can you please inform me of where I can find the new API (if it exists), or alternatively, provide a work-around so we can get the debuggers working on Ruby 2.0

Thanks

Related issues:

Related to Ruby trunk - Bug #7259: ruby-debug-base19x does not compile on 2.0...	**Closed**	**11/02/2012**

History

#1 - 10/25/2012 08:29 PM - ko1 (Koichi Sasada)

Thank you for your comment.

I want to support debugging feature, but my hands doesn't work on it yet.

Does "debugging feature" guru attend RubyConf2012 next week?
I will attend it.
If there is (are), I want to finish a design (and an implementation I hope) about it for 2.0.

Thanks,
Koichi

(2012/10/25 19:46), banister (john mair) wrote:

> Issue #7214 has been reported by banister (john mair).
>
> ---
>
> Bug #7214: Ruby 2.0 breaks support for some debugging tools
> https://bugs.ruby-lang.org/issues/7214
>
> Author: banister (john mair)
> Status: Open
> Priority: Normal
> Assignee:
> Category:
> Target version:
> ruby -v: ruby 2.0.0dev (2012-10-18 trunk 37260) [x86_64-linux]
>
> Notably the "Pry" debugger breaks, and (though i haven't checked) I'm assuming the 'debugger' project as well. The reason for the breakages (as far as i can see) is that the rb_vm_make_env_object function is now hidden. In the comments for 1.9.3's vm.c it was stated an alternative API for rb_vm_make_env_object (see https://github.com/ruby/ruby/blob/ruby_1_9_3/vm.c#L53-60) would be provided, but I have been unable to find one.
>
> Can you please inform me of where I can find the new API (if it exists), or alternatively, provide a work-around so we can get the debuggers working on Ruby 2.0
>
> Thanks

--

// SASADA Koichi at atdot dot net

#2 - 10/27/2012 09:30 AM - ko1 (Koichi Sasada)

- Category set to core

- Assignee set to ko1 (Koichi Sasada)

- Target version set to 2.0.0

#3 - 10/28/2012 05:14 AM - banister (john mair)

ko1 (Koichi Sasada)

Unfortunately I can't make it to rubconf2012.

What we need for Ruby 2.0 is the binding_of_caller gem working (https://github.com/banister/binding_of_caller/blob/ruby-2.0/ext/binding_of_caller/binding_of_caller.c). We need to generate Binding objects from parent frames further up the call-stack, and it looks like the thing stopping us in Ruby 2.0 is the visibility of the rb_vm_make_env_object function.

We are using this binding_of_caller functionality to great effect in the pry-rescue (https://github.com/conradirwin/pry-rescue) and pry-stack_explorer (https://github.com/pry/pry-stack_explorer) projects, they allow some very powerful smalltalk-style workflows, and it would be a great shame if they can't work in 2.0 as they work brilliantly in Ruby 1.9.2-1.9.3 currently.

I am willing to help in any way possible to make this a reality for Ruby 2.0, just let me know what you need.

Thanks!

#4 - 10/28/2012 05:29 AM - ko1 (Koichi Sasada)

(2012/10/28 5:14), banister (john mair) wrote:

> Unfortunately I can't make it to rubconf2012.
>
> What we need for Ruby 2.0 is the binding_of_caller gem working (https://github.com/banister/binding_of_caller/blob/ruby-2.0/ext/binding_of_caller/binding_of_caller.c). We need the ability to grab bindings from parent frames up the call-stack.
>
> We are using this binding_of_caller functionality to great effect in the pry-rescue (https://github.com/conradirwin/pry-capture) and pry-stack_explorer (https://github.com/pry/pry-stack_explorer) projects, they allow some very powerful workflows, and it would be a real shame if they can't work in 2.0 as they work brilliantly in Ruby 1.9.2-1.9.3 currently.
>
> I am willing to help in any way possible to make this a reality for Ruby 2.0, just let me know what you need.

Thank you. It is great information for me.
Can I find API description?

--
// SASADA Koichi at atdot dot net

#5 - 10/28/2012 09:08 PM - banister (john mair)

The API is this:

class methods

return the binding for the nth caller, where Binding.of_caller(0) == binding

Binding.of_caller(n)

return an array of all the caller bindings (this is useful when you want to take a snap-shot of the entire call stack for later inspection, as in pry-rescue)

Binding.callers

The number of frames currently on the stack

Binding.frame_count

instance methods

the frame type, e.g :block, :class, :top, :lambda, :method i.e VM_FRAME_MAGIC_METHOD

Binding#frame_type

The frame description, i.e cfp->iseq->name on 1.9.3, returns stuff like "block in my_method"

Binding#frame_description

Note that we skip some frames (such as VM_FRAME_MAGIC_IFUNC) as introspecting on them appears to return junk data.

#6 - 10/28/2012 09:23 PM - ko1 (Koichi Sasada)

Thank you for your explanation.

(2012/10/28 21:08), banister (john mair) wrote:

> # return the binding for the nth caller, where Binding.of_caller(0) == binding
>
> Binding.of_caller(n)

`Kernel.binding' can't return binding of CFUNC.
Is it enough?

--
// SASADA Koichi at atdot dot net

#7 - 10/28/2012 10:59 PM - banister (john mair)

No problem, our current code skips CFUNC frames too :)

Are you thinking of exposing this API to Ruby (in core or stdlib) or just as a C API ?

#8 - 10/29/2012 12:23 AM - ko1 (Koichi Sasada)

(2012/10/28 22:59), banister (john mair) wrote:

> No problem, our current code skips CFUNC frames too :)

Okay.

> Are you thinking of exposing this API to Ruby (in core or stdlib) or just as a C API ?

I think it should be C API at Ruby 2.0.

or exposed on RubyVM::... (MRI, ruby 2.0 specific)?

--
// SASADA Koichi at atdot dot net

#9 - 11/15/2012 09:31 PM - ko1 (Koichi Sasada)

=begin
[PLEASE REVIEW!!]

= Abstract

I made debugger support interface.
https://github.com/ko1/ruby/compare/debugger_api

Currently, no docs, no tests.

Sorry for my laziness.

Please review it.

= Background

Generally, debugger needs two features.

(1) Flow (execution) control API
(2) Inpsection API

For (1), inserting breakpoints, watch points and so on.
However, I don't touch these features because of no time to discuss.
(set_trace_func and TracePoint will help it)

For (2), (2-1) inspecting current thread's frames, (2-2) thread frames and (2-3) global environment.
We can access (2-3) using Ruby's powerful reflecting features (such as global_variables and so on).
We lacks (2-1) and (2-2). Currently, we don't have flow control features ((1)'s feature) to stop other threads.
So I decide to support only (2-1) "inspecting current thread's frames".

To make (2-1), Binding.of_caller' (which is supported by 'binding_of_caller') gem is almost enough for debugger.
However,Binding.of_caller' is too powerful and it can break Ruby's semantics.
I feel that we need to restrict for debugging purpose.

= API

With above consideration, I made a new C/Ruby API.

https://github.com/ko1/ruby/compare/debugger_api

== C-Level APIs

Types:

- typedef struct rb_debug_inspector_struct rb_debug_inspector_t;
- typedef VALUE (*rb_debug_inspector_func_t)(const rb_debug_inspector_t *, void *);

Functions:

- VALUE rb_debug_inspector_open(rb_debug_inspector_func_t func, void *data);
- VALUE rb_debug_inspector_frame_binding_get(const rb_debug_inspector_t *dc, int index);
- VALUE rb_debug_inspector_frame_class_get(const rb_debug_inspector_t *dc, int index);
- VALUE rb_debug_inspector_backtrace_locations(const rb_debug_inspector_t *dc);

== Ruby-level APIs (RubVM::DebugInspector)

You can use the following APIs after "require 'rubyvm/debug_inspector'".

- RubVM::DebugInspector.open{|inspector| ...}
- RubVM::DebugInspector#backtrace_locations #=> locations array
- RubVM::DebugInspector#frame_binding(i) #=> i-th binding. Generated bindings can be broken' outside ofRubVM::DebugInspector.open' block to avoid abusing this powerful feature (now, they are not broken)
- RubVM::DebugInspector#frame_class(i) #=> i-th method class

Ruby level API is sample code of C-level APIs.

== Sample code

The following sample code is very simple debugger (breakpoint).

```
require 'rubyvm/debug_inspector'

def breakpoint
RubyVM::DebugInspector.open{|inspector|
$inspector = inspector
inspector.backtrace_locations.each_with_index{|location, i|
b = inspector.frame_binding(i) # binding is nil if it is for when C's context
vars = b ? b.eval('local_variables') : []
puts [i, location.to_s, vars, b, inspector.frame_class(i)].inspect
}
}
end

def foo
foo_a = foo_b = nil
breakpoint
end

hello = 1
foo
```

```
begin
# DebugInspector object is not active outside block of
# RubyVM::DebugInspector.open
p $inspector.backtrace_locations
rescue ArgumentError => e
p [:ok, e]
end
```

= Consideration

== Satisfaction

I'm sorry I don't make surveys about debugger APIs for current CRuby's debugger
and debuggers for alternative Ruby implementations.
Please point out that it is enough or not enough, misunderstood and so on.

== Toward Ruby 2.0.0

Ruby 2.0.0 spec was already frozen :(
Make debug_inspector gem for 2.0.0?

=end

#10 - 11/15/2012 10:35 PM - ko1 (Koichi Sasada)

I asked Matz about this feature.

His comments were:

(1) Do not need to break bindings at end of block. This is programmer's risk.
(2) Ruby-level API is also okay to contains Ruby 2.0.0.

#11 - 11/16/2012 12:30 AM - ko1 (Koichi Sasada)

I asked mame-san (2.0.0 release manager) about this feature.

His comments is:
DO IT ON A GEM SUCH A BIG FEATURE.

His comment is: it should be experimental just now. We need to make examination with real debugger.

#12 - 11/17/2012 06:35 PM - Conrad.Irwin (Conrad Irwin)

Hey ko1,

Your debugging API looks good :).

It would be great to do this in a gem, but we can't create binding objects anymore due to changes in symbol visibility. (for 1.9 we used rb_vm_make_env_object, but it's now not exported, see [1])

This patch "fixes" it, but there should be a better way: https://gist.github.com/2f19a3cffb1a7bdfaf22

Is there any chance you can make this function callable from C extensions?

Thanks!
Conrad

[1] https://github.com/banister/binding_of_caller/blob/ruby-2.0/ext/binding_of_caller/binding_of_caller.c#L156 (edit, updated link)

#13 - 11/17/2012 07:53 PM - ko1 (Koichi Sasada)

(2012/11/17 18:35), Conrad.Irwin (Conrad Irwin) wrote:

> It would be great to do this in a gem, but we can't create binding objects anymore due to changes in symbol visibility. (for 1.9 we used rb_vm_make_env_object, but it's now not exported, see [1])

Why do you need `rb_vm_make_env_object'? New API is not enough?

--
// SASADA Koichi at atdot dot net

#14 - 11/19/2012 11:43 PM - denofevil (Dennis Ushakov)

I will try to rewrite ruby-debug-base for 2.0 using your fork during this week and will post about results

#15 - 11/20/2012 07:53 AM - ko1 (Koichi Sasada)

(2012/11/19 23:43), denofevil (Dennis Ushakov) wrote:

> I will try to rewrite ruby-debug-base for 2.0 using your fork during this week and will post about results

Can I see ruby-debug-base source code?
I will try it and find out lacked feature.

--
// SASADA Koichi at atdot dot net

#16 - 11/20/2012 05:46 PM - denofevil (Dennis Ushakov)

Yes, sure: https://github.com/ruby-debug/ruby-debug-base19

#17 - 11/20/2012 08:23 PM - ko1 (Koichi Sasada)

(2012/11/20 17:46), denofevil (Dennis Ushakov) wrote:

> Yes, sure: https://github.com/ruby-debug/ruby-debug-base19

Thanks! I'll try it.

BTW, I can't understand relationship between debugger' andruby-debug'.

--
// SASADA Koichi at atdot dot net

#18 - 11/20/2012 08:53 PM - luislavena (Luis Lavena)

On Tue, Nov 20, 2012 at 8:08 AM, SASADA Koichi ko1@atdot.net wrote:

> Thanks! I'll try it.
>
> BTW, I can't understand relationship between debugger' andruby-debug'.

AFAIK: debugger gem was a response to many of the installation issues
ruby-debug-base19 had, specially due the need to download Ruby source
and extract the internal headers during installation.

Reason of fork:

https://github.com/cldwalker/debugger#reason-for-fork

And main differences:

https://github.com/cldwalker/debugger#whats-different-from-ruby-debug19

--
Luis Lavena
AREA 17
-
Perfection in design is achieved not when there is nothing more to add,
but rather when there is nothing more to take away.
Antoine de Saint-Exupéry

#19 - 11/20/2012 09:04 PM - denofevil (Dennis Ushakov)

luislavena (Luis Lavena) wrote:

> AFAIK: debugger gem was a response to many of the installation issues
> ruby-debug-base19 had, specially due the need to download Ruby source
> and extract the internal headers during installation.
> Actually link that I gave is sources of ruby-debug-base19x. These are both forks for the same reasons =)
> ruby-debug-base19x was started earlier and intended to keep native frontend(base gem) and IDE/CLI backends separate.
> Sources of the ruby-debug-base19x and debugger native part share lots of code

#20 - 11/24/2012 08:57 PM - spastorino (Santiago Pastorino)

Hope this https://bugs.ruby-lang.org/issues/6586 could be done any time soon :)

#21 - 12/12/2012 11:42 PM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Third Party's Issue

Debuggers should use new TracePoint feature now.