Ruby - Feature #977

caller for all threads patch

01/04/2009 03:46 AM - rogerdpack (Roger Pack)

Status:	Closed	
Priority:	Normal	
Assignee:	ko1 (Koichi Sasada)	
Target version:	1.9.2	

Description

=begin
Here is a patch which provides backtrace for all current threads, instead of just the current one.
http://ph7spot.com/articles/caller_for_all_threads
Author said it would be great to have it accepted upstream.
Thoughts?
-=r
=end

History

#1 - 01/04/2009 03:48 AM - rogerdpack (Roger Pack)

=begin
oops that's a feature request not a bug--for some reason I thought it would default to a feature request since that's the view from whence I clicked "Submit issue"
My bad.
-=r
=end

#2 - 02/02/2009 12:59 PM - ko1 (Koichi Sasada)

- Assignee set to ko1 (Koichi Sasada)

- Target version set to 1.9.2

=begin

=end

#3 - 06/09/2009 08:06 AM - ko1 (Koichi Sasada)

=begin
I made a patch to Thread#caller(lev=1). It may be more flexible than
fetching "all" backtrace.
How about it? (not tested enough)

Index: vm_eval.c

--- vm_eval.c (□□□□□ 23650)
+++ vm_eval.c (□□□□□)
@@ -1342,6 +1342,19 @@ rb_make_backtrace(void)
}

VALUE
+rb_thread_backtrace(VALUE thval, int lev)
+{

- rb_thread_t *th;
- GetThreadPtr(thval, th);

- if (th->status != THREAD_KILLED && GET_THREAD() != th) {
- lev--;
- }

- return vm_backtrace(th, lev);
+}

+VALUE
rb_backtrace_each(rb_backtrace_iter_func *iter, void *arg)
{
return vm_backtrace_each(GET_THREAD(), -1, iter, arg);
Index: thread.c

--- thread.c (□□□□□ 23651)
+++ thread.c (□□□□□)
@@ -3817,6 +3817,26 @@ ruby_suppress_tracing(VALUE (*func)(VALU
return result;
}

+VALUE rb_thread_backtrace(VALUE thval, int lev);
+
+static VALUE
+rb_thread_caller_m(int argc, VALUE *argv, VALUE thval)
+{

- VALUE level;
- int lev;

- rb_scan_args(argc, argv, "01", &level);

- if (NIL_P(level))
- lev = 1;
- else
- lev = NUM2INT(level);
- if (lev < 0)
- rb_raise(rb_eArgError, "negative level (%d)", lev);

- return rb_thread_backtrace(thval, lev);
  +}

/*

- +Thread+ encapsulates the behavior of a thread of
- execution, including the main thread of the Ruby script.
  @@ -3873,6 +3893,7 @@ Init_Thread(void)
  rb_define_method(rb_cThread, "abort_on_exception=", rb_thread_abort_exc_set, 1);
  rb_define_method(rb_cThread, "safe_level", rb_thread_safe_level, 0);
  rb_define_method(rb_cThread, "group", rb_thread_group, 0);

- rb_define_method(rb_cThread, "caller", rb_thread_caller_m, -1);

  rb_define_method(rb_cThread, "inspect", rb_thread_inspect, 0);

Roger Pack wrote::

> Bug #977: caller for all threads patch
> http://redmine.ruby-lang.org/issues/show/977
>
> Author: Roger Pack
> Status: Open, Priority: Normal
>
> Here is a patch which provides backtrace for all current threads, instead of just the current one.
> http://ph7spot.com/articles/caller_for_all_threads
> Author said it would be great to have it accepted upstream.
> Thoughts?
> -=r
>
> ---
>
> http://redmine.ruby-lang.org

--
// SASADA Koichi at atdot dot net

=end

#4 - 06/09/2009 08:33 AM - ko1 (Koichi Sasada)

=begin
Hongli Lai wrote::

> SASADA Koichi wrote:
>
> > I made a patch to Thread#caller(lev=1). It may be more flexible than
> > fetching "all" backtrace.
> > How about it? (not tested enough)
>
> The ability to see all running threads' backtraces, without needing a
> reference to each one of those threads, is caller_for_all_thread's main
> advantage. It's very useful for debugging a live application. Replacing
> it with Thread#caller would require one to maintain references to all
> threads that one wants to inspect. Does Ruby already provide some way to
> obtain a list of all running threads?

def caller_for_all_thread
Thread.list.map{|t| t.caller}
end

--
// SASADA Koichi at atdot dot net

=end

#5 - 06/09/2009 09:33 AM - ko1 (Koichi Sasada)

=begin
Rocky Bernstein wrote::

> One thing I think might be cool is rather than raising an error for a
> negative Fixnum value is to count from the other end. So caller(-1) is the
> least-recent call.
>
> If you want me to try my hand at extending the below, let me know.

Check the following code.

VALUE
+rb_thread_backtrace(VALUE thval, int lev)
+{

- rb_thread_t *th;
- GetThreadPtr(thval, th);

- if (th->status != THREAD_KILLED && GET_THREAD() != th) {

- [redacted]

- }

- return vm_backtrace(th, lev);
  +}

--
// SASADA Koichi at atdot dot net

=end

#6 - 06/10/2009 05:24 AM - ko1 (Koichi Sasada)

=begin
Rocky Bernstein wrote::

> I was suggesting that rather than raise an error here, treat this like array
> indexes do and basically use size - level. (By the way, also suggests it
> might be cool to add some sort of length or size function.)

> I guess I missed something, but what?

I had misunderstood your suggestion. At first, you should suggest the "Kernel.caller" specification, not the Thread#caller spec.

Regards,

// SASADA Koichi at atdot dot net

=end

#7 - 06/10/2009 11:39 AM - ko1 (Koichi Sasada)

=begin
Roger Pack wrote::

> I really like it.
> Appears that it wants default to be level 0 [?]

Thank you for your notice.

I've change my thought. Thread#backtrace() is more proffered name.

- On Thread#caller(lev), nobody may use lev (!= 0)
- Deciding the semantics of lev except zero may be difficult

How about it?

--
// SASADA Koichi at atdot dot net

=end

#8 - 06/12/2009 08:29 AM - ko1 (Koichi Sasada)

=begin
Roger Pack wrote::

> > Roger Pack wrote::
> >
> > > I really like it.
> > > Appears that it wants default to be level 0 [?]
> > > Thank you for your notice.
> >
> > I've change my thought. Thread#backtrace() is more proffered name.
> >
> > - On Thread#caller(lev), nobody may use lev (!= 0)
> > - Deciding the semantics of lev except zero may be difficult
>
> That sounds better. Then the semantics for caller never change. So
> this would be Thread#backtrace can have lev > 0?
> Either way's good for me.

No. Same as Exception#backtrace.

How about it, matz?

--
// SASADA Koichi at atdot dot net

=end

#9 - 06/17/2009 08:07 AM - matz (Yukihiro Matsumoto)

=begin
Hi,

In message "Re: [ruby-core:23812] Re: [Bug #977] caller for all threads patch"
on Fri, 12 Jun 2009 08:28:53 +0900, SASADA Koichi ko1@atdot.net writes:

|No. Same as Exception#backtrace.
|How about it, matz?

I see no problem.

```
        matz.
```

=end

#10 - 07/23/2009 11:03 PM - rogerdpack (Roger Pack)

=begin
You can close this one--thank you to Ko1 for implementing it for me.
=end

#11 - 07/23/2009 11:38 PM - shyouhei (Shyouhei Urabe)

- Status changed from Open to Closed

=begin

=end