Ruby master - Bug #4400

nested at_exit hooks run in strange order

02/15/2011 03:49 PM - sunaku (Suraj Kurapati)

<table>
<thead>
<tr>
<th>Status:</th>
<th>Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td>ko1 (Koichi Sasada)</td>
</tr>
<tr>
<td>Target version:</td>
<td>2.0.0</td>
</tr>
<tr>
<td>ruby -v:</td>
<td>ruby 1.9.2p136 (2010-12-25 revision 30365) [x86_64-linux]</td>
</tr>
</tbody>
</table>

Description

=begin
Hello,

The documentation for Kernel#at_exit says "If multiple [at_exit] handlers are registered, they are executed in reverse order of registration". However, does not seem to be true for nested at_exit hooks (registering an at_exit hook inside another at_exit hook). For example consider this code:

at_exit { puts :outer0 }
at_exit { puts :outer1_begin; at_exit { puts :inner1 }; puts :outer1_end }
at_exit { puts :outer2_begin; at_exit { puts :inner2 }; puts :outer2_end }
at_exit { puts :outer3 }

Here is the output of running this code with two Rubies:

ruby 1.9.2p136 (2010-12-25 revision 30365) [x86_64-linux]
outer3
outer2_begin
outer2_end
outer1_begin
outer1_end
outer0
inner1
inner2

ruby 1.8.7 (2010-08-16 patchlevel 302) [x86_64-linux]
outer3
outer2_begin
outer2_end
outer1_begin
outer1_end
outer0
inner1
inner2

Observe how inner1 and inner2 are executed in registration order after all non-nested hooks are executed in reverse registration order. This seems very strange to me; I would expect nested at_exit hooks to be executed immediately (as follows) because we are already inside the at_exit phase of the program:

outer3
outer2_begin
inner2
outer2_end
outer1_begin
inner1
outer1_end
outer0

What do you think? Thanks for your consideration.
=end
eval_jump.c (rb_exec_end_proc): changed at_exit and END proc evaluation order. [Bug #4400] [ruby-core:35237]

eval_jump.c (rb_mark_end_proc): ditto.

test/ruby/test_beginendblock.rb (TestBeginEndBlock#test_nested_at_exit): added a test for nested at_exit.

test/ruby/test_beginendblock.rb (TestBeginEndBlock#test_beginendblock): changed the test to adopt new spec.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@30888 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 30888 - 02/16/2011 11:42 AM - kosaki (Motohiro KOSAKI)

eval_jump.c (rb_exec_end_proc): changed at_exit and END proc evaluation order. [Bug #4400] [ruby-core:35237]

eval_jump.c (rb_mark_end_proc): ditto.

test/ruby/test_beginendblock.rb (TestBeginEndBlock#test_nested_at_exit): added a test for nested at_exit.

test/ruby/test_beginendblock.rb (TestBeginEndBlock#test_beginendblock): changed the test to adopt new spec.

Revision 30888 - 02/16/2011 11:42 AM - kosaki (Motohiro KOSAKI)

eval_jump.c (rb_exec_end_proc): changed at_exit and END proc evaluation order. [Bug #4400] [ruby-core:35237]

eval_jump.c (rb_mark_end_proc): ditto.

test/ruby/test_beginendblock.rb (TestBeginEndBlock#test_nested_at_exit): added a test for nested at_exit.

test/ruby/test_beginendblock.rb (TestBeginEndBlock#test_beginendblock): changed the test to adopt new spec.

Revision 30888 - 02/16/2011 11:42 AM - kosaki (Motohiro KOSAKI)

eval_jump.c (rb_exec_end_proc): changed at_exit and END proc evaluation order. [Bug #4400] [ruby-core:35237]

eval_jump.c (rb_mark_end_proc): ditto.

test/ruby/test_beginendblock.rb (TestBeginEndBlock#test_nested_at_exit): added a test for nested at_exit.

test/ruby/test_beginendblock.rb (TestBeginEndBlock#test_beginendblock): changed the test to adopt new spec.

Revision 30888 - 02/16/2011 11:42 AM - kosaki (Motohiro KOSAKI)

eval_jump.c (rb_exec_end_proc): changed at_exit and END proc evaluation order. [Bug #4400] [ruby-core:35237]

eval_jump.c (rb_mark_end_proc): ditto.
History

#1 - 02/15/2011 03:56 PM - sunaku (Suraj Kurapati)

=begin
By the way, this issue is not contrived. It prevents propagation of a
proper exit status when using Test::Unit with Capybara (Selenium driver)
where a unit test (run from Test::Unit's at_exit hook) loads the Capybara
library which registers an at_exit hook of its own.

As a result, Test::Unit always exits with 0 status, even if there were
assertion failures, because Capybara's at_exit hook runs after Test::Unit's
at_exit hook and overrides its exit status setting.

See this bug report for full details:

https://github.com/jnicklas/capybara/issues#issue/178/comment/658647
=end

#2 - 02/15/2011 07:38 PM - kosaki (Motohiro KOSAKI)

=begin
2011/2/15 Suraj Kurapati redmine@ruby-lang.org:

Bug #4400: nested at_exit hooks run in strange order
http://redmine.ruby-lang.org/issues/show/4400

Author: Suraj Kurapati
Status: Open, Priority: Normal
Category: core
ruby -v: ruby 1.9.2p136 (2010-12-25 revision 30365) [x86_64-linux]

Hello,

The documentation for Kernel#at_exit says "If multiple at_exit handlers are
registered, they are executed in reverse order of registration". However, does
not seem to be true for nested at_exit hooks (registering an at_exit hook inside another at_exit hook). For example consider this code:

```ruby
at_exit { puts :outer0 }
at_exit { puts :outer1_begin; at_exit { puts :inner1 }; puts :outer1_end }
at_exit { puts :outer2_begin; at_exit { puts :inner2 }; puts :outer2_end }
at_exit { puts :outer3 }
```

Here is the output of running this code with two Rubies:

```
ruby 1.9.2p136 (2010-12-25 revision 30365) [x86_64-linux]
outer3
outer2_begin
outer2_end
outer1_begin
outer1_end
outer0
inner1
inner2

inner1
inner2
```

```
ruby 1.8.7 (2010-08-16 patchlevel 302) [x86_64-linux]
outer3
outer2_begin
outer2_end
outer1_begin
outer1_end
outer0
inner1
inner2

Observe how inner1 and inner2 are executed in registration order after all non-nested hooks are executed in reverse registration order. This seems very strange to me; I would expect nested at_exit hooks to be executed immediately (as follows) because we are already inside the at_exit phase of the program:

```
outer3
outer2_begin
inner2
outer2_end
outer1_begin
inner1
outer1_end
outer0
```

What do you think? Thanks for your consideration.

btw, C's atexit() has different behavior.

```c
#include
#include

static void func0(void) { printf("outer0\n"); }

static void func1_inner(void) { printf("inner1\n"); }

static void func1(void)
{
    printf("outer1_begin\n");
atexit(func1_inner);
    printf("outer1_end\n");
}

static void func2_inner(void) { printf("inner2\n"); }

static void func2(void)
{
    printf("outer2_begin\n");
atexit(func2_inner);
    printf("outer2_end\n");
}
```
static void func3(void) { printf("outer3\n"); }

main()
{
atexit(func0);
atexit(func1);
atexit(func2);
atexit(func3);
}

% gcc at_exit.c; ./a.out
outer3
outer2_begin
outer2_end
inner2
outer1_begin
outer1_end
inner1
outer0

=end

#3 - 02/15/2011 08:25 PM - kosaki (Motohiro KOSAKI)
- File bug4400-atexit.patch added
- Assignee set to ko1 (Koichi Sasada)
- Target version set to 2.0.0

=begin
The attached patch is to adapt C's behavior.
And, Current behavior seems to be introduced by following commit.
Therefore we should hear ko1's opinion. I think.

ko1, what do you think?

commit a3e1b1ce7ed7e7ffac23015fc2fde56511b30681
Author: ko1 ko1@b2dd03c8-39d4-4d8f-98ff-823fe69b080e
Date:   Sun Dec 31 15:02:22 2006 +0000
    * Merge YARV
    
git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@11439 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

=end

#4 - 02/15/2011 10:31 PM - kosaki (Motohiro KOSAKI)
=begin
    btw, C's atexit() has different behavior.
    
(snip)

% gcc at_exit.c; ./a.out
outer3
outer2_begin
outer2_end
inner2
outer1_begin
outer1_end
inner1
outer0

Python has the same behavior with C.

test_atexit.py

import atexit
def func0():
    print "outer0"

def func1_internal():
    print "inner1"

def func1():
    print "outer1_begin"
    atexit.register(func1_internal);
    print "outer1_end"

def func2_internal():
    print "inner2"

def func2():
    print "outer2_begin"
    atexit.register(func2_internal);
    print "outer2_end"

def func3():
    print "outer3"
    atexit.register(func0);
    atexit.register(func1);
    atexit.register(func2);
    atexit.register(func3);
=end

#5 - 02/15/2011 10:32 PM - ko1 (Koichi Sasada)
=begin
(2011/02/15 20:25), Motohiro KOSAKI wrote:
>
ko1, what do you think?

I don't have any idea about it. However, I think it should be a specification issue == Matz issue.

Regards,
Koichi

--
// SASADA Koichi at atdot dot net
=end

#6 - 02/15/2011 11:48 PM - matz (Yukihiro Matsumoto)
=begin
Hi,

In message "Re: [ruby-core:35252] Re: [Ruby 1.9-Bug#4400] nested at_exit hooks run in strange order" on Tue, 15 Feb 2011 22:32:39 +0900, SASADA Koichi ko1@atdot.net writes:
>
| (2011/02/15 20:25), Motohiro KOSAKI wrote:
| > ko1, what do you think?
| |
| I don't have any idea about it. However, I think it should be a specification issue == Matz issue.

OK, I choose C's behavior. Although I don't recommend to rely too much on the atexit order. Motohiro, could you check in?

matz.

=end

#7 - 02/16/2011 04:25 AM - sunaku (Suraj Kurapati)
=begin
Cool! I prefer C's behavior also. Thank you.
=end

09/20/2021
#8 - 02/16/2011 05:21 AM - kosaki (Motohiro KOSAKI)

begin

|> ko1, what do you think?
| I don't have any idea about it. However, I think it should be a specification issue == Matz issue.

OK, I choose C's behavior. Although I don't recommend to rely too much on the atexit order. Motohiro, could you check in?

Yes, sir.~)

=end

#9 - 02/16/2011 08:47 PM - kosaki (Motohiro KOSAKI)

- Status changed from Open to Closed
- % Done changed from 0 to 100

begin

This issue was solved with changeset r30888.
Suraj, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

* eval_jump.c (rb_exec_end_proc): changed at_exit and END proc evaluation order. [Bug #4400] [ruby-core:35237]

* eval_jump.c (rb_mark_end_proc): ditto.

* test/ruby/test_beginendblock.rb (TestBeginEndBlock#test_nested_at_exit): added a test for nested at_exit.

* test/ruby/test_beginendblock.rb (TestBeginEndBlock#test_beginendblock): changed the test to adopt new spec.

=end

#10 - 02/18/2011 12:16 AM - headius (Charles Nutter)

begin

FWIW, JRuby already seems to match the C ordering, though I don't think we did it on purpose:

~/projects/jruby ➔ jruby at_exit.rb
outer3
outer2_begin
outer2_end
inner2
outer1_begin
outer1_end
inner1
outer0

=end

#11 - 02/18/2011 02:30 AM - jballanc (Joshua Ballanco)

begin

On Thu, Feb 17, 2011 at 10:16 AM, Charles Nutter redmine@ruby-lang.org wrote:

Issue #4400 has been updated by Charles Nutter.

FWIW, JRuby already seems to match the C ordering, though I don't think we did it on purpose:

~/projects/jruby ➔ jruby at_exit.rb
outer3
outer2_begin
outer2_end
Seems the same is true of MacRuby:

```ruby
DeepThought: ~/Source/MacRuby > macruby at_exit.rb
outer3
outer2_begin
outer2_end
inner2
outer1_begin
outer1_end
inner1
outer0
```

On Thu, Feb 17, 2011 at 10:16 AM, Charles Nutter <redmine@ruby-lang.org> wrote:

Issue #4400 has been updated by Charles Nutter.

FWIW, JRuby already seems to match the C ordering, thought I don't think we did it on purpose:

```ruby
~/projects/jruby ➔ jruby at_exit.rb
outer3
outer2_begin
outer2_end
inner2
outer1_begin
outer1_end
inner1
outer0
```

Seems the same is true of MacRuby:

```ruby
DeepThought: ~/Source/MacRuby > macruby at_exit.rb
outer3
outer2_begin
outer2_end
inner2
outer1_begin
outer1_end
inner1
outer0
```

### Files

<table>
<thead>
<tr>
<th>File</th>
<th>Size</th>
<th>Date</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>bug4400-atexit.patch</td>
<td>1.99 KB</td>
<td>02/15/2011</td>
<td>kosaki (Motohiro KOSAKI)</td>
</tr>
</tbody>
</table>