Ruby master - Bug #4925

Infinite recursion allowed in rescue clause

06/25/2011 01:01 PM - wuputah (Jonathan Dance)

Status:	Rejected		
Priority:	Normal		
Assignee:	ko1 (Koichi Sasada)		
Target version:	1.9.3		
ruby -v:	ruby 1.9.3dev (2011-06-24 trunk 32222) [i686-linux]	**Backport:**	

Description

The issue pertains to the following example:

```
def a
b
rescue NameError
a
end
a
```

The type of exception raised and rescued is not relevant; the bug occurs with any exception type.

Expected behavior: A SystemStackError should be raised.

Current behavior: The Ruby process will consume cpu and memory until killed.

This is a regression in 1.9.x; the bug does not occur in 1.8-head. The bug is reproducible on all versions of 1.9 I have tested:

```
ruby 1.9.2p180 (2011-02-18 revision 30909) [i686-linux]
ruby 1.9.2p274 (2011-06-06 revision 31932) [i686-linux]
ruby 1.9.3dev (2011-06-24 trunk 32222) [i686-linux]
```

History

#1 - 06/26/2011 07:15 PM - naruse (Yui NARUSE)

- Status changed from Open to Assigned

- Assignee set to ko1 (Koichi Sasada)

#2 - 06/26/2011 07:22 PM - nahi (Hiroshi Nakamura)

- Target version set to 1.9.3

#3 - 07/10/2011 12:59 AM - mame (Yusuke Endoh)

- Status changed from Assigned to Rejected

Hello,

> Expected behavior: A SystemStackError should be raised.
>
> Current behavior: The Ruby process will consume cpu and memory until killed.
>
> This is a regression in 1.9.x

No, unfortunately. This is caused by an intended spec change.

The root spec change is that 1.9 retains the full backtrace.
The change itself is useful, I think.

```
$ cat t.rb
def b
p caller
end
```

```
def a(n)
if n == 0
b
else
a(n-1)
end
end

a(10)

$ /usr/bin/ruby -v t.rb
ruby 1.8.7 (2010-08-16 patchlevel 302) [i686-linux]
["t.rb:7:in a'", "t.rb:9:ina'", "t.rb:13"]

$ ruby -v t.rb
ruby 1.9.2p180 (2011-02-18 revision 30909) [i686-linux]
["t.rb:7:in a'", "t.rb:9:ina'", "t.rb:9:in a'", "t.rb:9:ina'", "t.rb:9:in a'", "t.rb:9:ina'", "t.rb:9:in a'", "t.rb:9:ina'", "t.rb:9:in a'", "t.rb:9:ina'", "t.rb:9:in a'", "t.rb:13:in'"]
```

But this change impacts your code maybe unintentionally.

Whenever NameError occurs, the exception object creates a backtrace.
In 1.9, the time is O(n) where n is the length of stack.
So, your code runs in $O(n^{2)}$ until the stack overflows.

In addition, all backtrace objects are not collected because the
exception object can be accessible as "$!" in the rescue clause.
So, your code also uses memory in $O(n^{2),}$ which will cause thrashing.

You can confirm this behavior by clearing backtrace manually:

```
$ cat t.rb
def a
b
rescue NameError
$!.backtrace.clear
a
end

$ time ruby -v t.rb
ruby 1.9.2p180 (2011-02-18 revision 30909) [i686-linux]
t.rb:2: stack level too deep (SystemStackError)

real    0m22.361s
user    0m22.185s
sys     0m0.084s
```

The code works, but is still slow. I think that it runs in $O(n^{2).}$

Anyway, this is not a bug, very unfortunately. Closing.

This may be solved by tail call optimization, but it should be
discussed in another thread. I personally dislike the optimization
in Ruby because it makes backtrace hard to understand.

--
Yusuke Endoh mame@tsg.ne.jp