Hi, I would like to propose the addition of two methods for setting and getting the recursion limit:

recursion_limit(limit)
recursion_limit => returns an Integer value

At this time, I have been experimenting with recursion to compute the nth fibonacci number using ruby 1.9.1 (i.e ruby 1.9.1p0 (2009-01-30 revision 21907)). These tests are being ran on a Apple PowerMac G5 2.5 Ghz Quad 4 GB RAM and a MacBook Pro 2.8 GHz Intel Core 2 Duo 4 GB RAM. Also, I was able to successfully produce a result using an iteration which I'm attaching to verify the correctness of the other two algorithms:

action: ./fibonacci-recursive.rb 10000

result:

./fibonacci-recursive.rb:10:in fibonacci': stack level too deep (SystemStackError)
from ./fibonacci-recursive.rb:10:in fibonacci'
from ./fibonacci-recursive.rb:10:in fibonacci'
from ./fibonacci-recursive.rb:10:in fibonacci'
from ./fibonacci-recursive.rb:10:in fibonacci'
from ./fibonacci-recursive.rb:10:in fibonacci'
from ./fibonacci-recursive.rb:10:in fibonacci'
from ./fibonacci-recursive.rb:10:in fibonacci'
... 8174 levels...
from ./fibonacci-recursive.rb:10:in fibonacci'
from ./fibonacci-recursive.rb:10:in fibonacci'
from ./fibonacci-recursive.rb:10:in fibonacci'
from ./fibonacci-recursive.rb:18:in ''

action: ./fibonacci-tail-recursive.rb 10000

result:

./fibonacci-tail-recursive.rb:10:in fibonacci': stack level too deep (SystemStackError)
from ./fibonacci-tail-recursive.rb:10:in fibonacci Helper'
from ./fibonacci-tail-recursive.rb:10:in fibonacci Helper'
from ./fibonacci-tail-recursive.rb:10:in fibonacci Helper'
from ./fibonacci-tail-recursive.rb:10:in fibonacci Helper'
from ./fibonacci-tail-recursive.rb:10:in fibonacci Helper'
from ./fibonacci-tail-recursive.rb:10:in fibonacci Helper'
from ./fibonacci-tail-recursive.rb:10:in fibonacci Helper'
... 7265 levels...
from ./fibonacci-tail-recursive.rb:10:in fibonacci Helper'
from ./fibonacci-tail-recursive.rb:10:in fibonacci Helper'
from ./fibonacci-tail-recursive.rb:17:in fibonacci'
from ./fibonacci-tail-recursive.rb:24:in ''

action: ./fibonacci-iterative.rb 10000

result:

336447648764317832666216120051075433103021484606800639065647699746800814421666623681555955136337340255820
65326808361593737490483865626823604089246305643188735454436955988724916066020989841839386645267313008888
30269235673613135111315111757929743785441375213052050434771062264758318960527890855154366159582987279828975106
31200575428783453214319252765022737629441540360584402572114349461800203901282870460898293623288354615057765
8327125256073359112520792825285934346290245248929439017062338899108581406518317386043747073790855626317
64325733993712871937578746897479923605837065742301616374089619784263784212383521128205163702989093220
9990570979200646374620238978311470054074998495250360633560399388383192338678305613643535189213273792390811
33733246256339897639227234078928177953580570993691049175470808931841056146323382174646373212482653830
102109277016048547262438374862411145309381220656491403275108664339451751216152655436133111314042436548
501067658434935238369565342808717687753283482343455573667197313927462736910821067928078471980353291117167
78924659099386354592789452377764461922403376386740040213034332947969202832814593341882681768389930720
363347956231171031012191953169794607632737589253530772552375943788435404067715555779056450443016640194625
80972216729758615026968441469520346149322911059706276432851599283470989128470674086200858117530162603120
719031720680940812983215810772820763531866246112782455320853236507575964300725177443150515396090951686
323220341632226408858248582433518051515469426234848298390905050704838424932745732624567557589089187198036
6205800959474315005240253270974969531877072437682590741939936322659841474981936092852239450397071654431564
213281576899080587831834049174345562705202235648464951961124602643139709750693824687066113264507665075411
1287572274826198064253071192844118262266105716351506962002986170494542504791437811151541399415506712562711
97133252763631939606902895602828686038622418028505624307017949767117121230660763310059947366875
=end

# History

#1 - 02/27/2009 12:33 AM - daz (Dave B)

=begin
See if this post helps:

http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-talk/89931
=end

#2 - 02/27/2009 03:32 AM - jredville (Jim Deville)

=begin
-----Original Message-----
From: Dave B [mailto:redmine@ruby-lang.org]
Sent: Thursday, February 26, 2009 7:33 AM
To: ruby-core@ruby-lang.org
Subject: [ruby-core:22545] [Feature #1218] New method needed to set and get the current recursion limit

Issue #1218 has been updated by Dave B.

See if this post helps:

http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-talk/89931
=end

That wouldn't help for Ruby on Windows. It would also like to have something tied to the code, not the system.

JD

=end

#3 - 02/28/2009 09:27 PM - Anonymous

=begin
Hi Dave, thanks for responding to the post. However, this doesn't resolve the issue that I'm seeing after increasing the stack size. There should be some class methods for obtaining and setting the recursion limit.

-Conrad

=end

#4 - 03/01/2009 11:23 PM - austin (Austin Ziegler)

=begin
On Sat, Feb 28, 2009 at 7:26 AM, Conrad Taylor redmine@ruby-lang.org wrote:

Issue #1218 has been updated by Conrad Taylor.

=end

08/06/2021 2/6
Hi Dave, thanks for responding to the post. However, this does resolve the issue that I'm seeing after increasing the stack size. There should be some class methods for obtaining and setting the recursion limit.

You can't change the recursion limit from within the program on all operating systems.

-austin

Austin Ziegler * halostatue@gmail.com * http://www.halostatue.ca/
* austin@halostatue.ca * http://www.halostatue.ca/feed/
* austin@zieglers.ca

---

#5 - 03/02/2009 05:42 PM - brent (Brent Roman)

setrlimit(RLIMIT_STACK, size_in_bytes)

should do what you want under a POSIX compliant OS.

See the manpage for setrlimit and getrlimit

This could easily be made accessible to Ruby code via a ‘C’ extension.

I have found that 1.8 Ruby is very inefficient in its use of stack-space and made some patches that improve this.

With ulimit -s 8192, the attached little script on gives up after:

```
1.6.8-mbari8B: 14374 levels
1.8.6-p287:  7198 levels
1.8.7-p72:  7197 levels
1.8.7-p72-mbari8B: 14750 levels
1.8.6-p287-mbari8B: 14750 levels
1.9.0: 7704 levels (but, see note below)
```

Ruby 1.9 seems completely unaffected by stack size configured via setrlimit (via ulimit -s from the bash shell)

Perhaps 1.9 is allocating the stack on the heap.
If this is the case, you will need some support in the 1.9 core for changing the size of this allocation.

  ● brent

Usaku NAKAMURA wrote:

Issue #1218 has been updated by Conrad Taylor.

Hi Dave, thanks for responding to the post. However, this does resolve the issue that I'm seeing after increasing the stack size. There should be some class methods for obtaining and setting the recursion limit.

-Conrad

http://redmine.ruby-lang.org/issues/show/1218

http://redmine.ruby-lang.org

---

View this message in context:
Sent from the ruby-core mailing list archive at Nabble.com.

#6 - 03/02/2009 06:01 PM - nobu (Nobuyoshi Nakada)
Hi,

At Mon, 2 Mar 2009 17:41:03 +0900, Brent Roman wrote in [ruby-core:22619]:

Ruby 1.9 seems completely unaffected by stack size configured via setrlimit (via ulimit -s from the bash shell)

Perhaps 1.9 is allocating the stack on the heap.

1.9 uses the default of pthread.

--
Nobu Nakada

#7 - 03/02/2009 10:08 PM - Anonymous

austin (Austin Ziegler) Ziegler - I was able to change these limits within Python using FreeBSD, Linux, and Mac OS X operating systems. The first two are production machines and the last is a development machine. Otherwise, the Python examples would have crapped out as did the Ruby recursive examples. Thus, I'm sure that it's possible on most modern platforms to change these limits within the Ruby language.

#8 - 03/03/2009 01:20 AM - brent (Brent Roman)

I don't do much pthreads programming. The call you want seems to be:

int pthread_attr_setstacksize(pthread_attr_t *attr, size_t stacksize)
Set the stack size attribute in a thread attributes object.
See:
http://developer.apple.com/DOCUMENTATION/Darwin/Reference/ManPages/man3/pthread.3.html (The prospective 'C' extension would have to get at the relevant pthread_attr_t)

Here's the little test script I forgot to attach yesterday:

$a=0
def x
  $a += 1
  x
end
begin
  x
rescue SystemStackError
  puts "After $#{$a} levels"
  puts $!
end

Conrad Taylor-3 wrote:

Issue #1218 has been updated by Conrad Taylor.

austin (Austin Ziegler) Ziegler - I was able to change these limits within Python using FreeBSD, Linux, and Mac OS X operating systems. The first two are production machines and the last is a development machine. Otherwise, the Python examples would have crapped out as did the Ruby recursive examples. Thus, I'm sure that it's possible on most modern platforms to change these limits within

the Ruby language.

http://redmine.ruby-lang.org/issues/show/1218
http://redmine.ruby-lang.org

--
View this message in context:
Sent from the ruby-core mailing list archive at Nabble.com.

#9 - 03/04/2009 01:44 AM - austin (Austin Ziegler)

=begin
On Mon, Mar 2, 2009 at 8:07 AM, Conrad Taylor redmine@ruby-lang.org wrote:

Issue #1218 has been updated by Conrad Taylor.

austin (Austin Ziegler) Ziegler - I was able to change these limits within Python using FreeBSD, Linux, and Mac OS X operating systems. The first two are production machines and the last is a development machine. Otherwise, the Python examples would have crapped out as did the Ruby recursive examples. Thus, I'm sure that it's possible on most modern platforms to change these limits within the Ruby language.

Did you try this on Windows?

I'm *not* sure it's possible on most modern platforms (and despite what people think and my own personal prejudices, Windows is a modern platform that Rubyists can't afford to ignore).

-austin

---
Austin Ziegler * halostatue@gmail.com * http://www.halostatue.ca/
* austin@halostatue.ca * http://www.halostatue.ca/feed/
* austin@zieglers.ca

=end

#10 - 03/05/2009 07:12 PM - Anonymous

=begin
Hi, I do not have access to a Windows machine because I do all my development using Unix environments.

-Conrad

=end

#11 - 09/14/2010 04:37 PM - shyouhei (Shyouhei Urabe)

- Status changed from Open to Assigned

=begin

=end

#12 - 02/13/2012 11:38 PM - mame (Yusuke Endoh)

- Status changed from Assigned to Rejected

I'm rejecting this feature ticket because no progress has been made for a long time. See [ruby-core:42391].

I think it is impossible to implement recursion_limit exactly because we cannot know how much stack each C function consumes.

I expected Python's recursion_limit to fail conservatively (i.e., it will raise an exception for big limit even when it is actually possible). But it was worse than I expected. Python caused SEGV when big limit is specified:

```python
import sys
sys.setrecursionlimit(1000000)
```

08/06/2021
def foo():
  foo()
  foo()  #=> SEGV

I think that this is not a correct way, at least, in Ruby.

--
Yusuke Endoh mame@tsg.ne.jp

<table>
<thead>
<tr>
<th>Files</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>fibonacci-iterative.rb</td>
<td>373 Bytes</td>
<td>02/27/2009</td>
<td>Anonymous</td>
</tr>
<tr>
<td>fibonacci-recursive.rb</td>
<td>408 Bytes</td>
<td>02/27/2009</td>
<td>Anonymous</td>
</tr>
<tr>
<td>fibonacci-tail-recursive.rb</td>
<td>478 Bytes</td>
<td>02/27/2009</td>
<td>Anonymous</td>
</tr>
</tbody>
</table>