Ruby master - Bug #3454
Segfault with syscall

<table>
<thead>
<tr>
<th>Status:</th>
<th>Rejected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td></td>
</tr>
<tr>
<td>Target version:</td>
<td>2.0.0</td>
</tr>
<tr>
<td>ruby -v:</td>
<td>ruby 1.9.3dev (2010-06-13 trunk 28312) [i686-linux]</td>
</tr>
</tbody>
</table>

Description

IRB segfaults consistently with the following (reduced from something sensible):

buf="",syscall(106, '/etc/passwd', buf)
106 is stat(2) on this system.

/usr/local/lib/ruby/1.9.1/irb/workspace.rb:81: [BUG] Segmentation fault
ruby 1.9.3dev (2010-06-13 trunk 2832) [i686-linux]

-- control frame ---------
c:0022 p---- s:0081 b:0081 l:000080 d:000080 CFUNC  :eval
c:0020 p:0023 s:0067 b:000066 d:000066 METHOD /usr/local/lib/ruby/1.9.1/irb/context.rb:167
c:0019 p:0039 s:0063 b:000063 d:000062 METHOD /usr/local/lib/ruby/1.9.1/irb/context.rb:254
c:0018 p:0031 s:0058 b:0058 d:000057 BLOCK /usr/local/lib/ruby/1.9.1/irb.rb:159
c:0015 p:0144 s:0041 b:0041 d:000040 BLOCK /usr/local/lib/ruby/1.9.1/irb/ruby-lex.rb:243
c:0014 p---- s:0038 b:0038 l:000037 d:000037 FINISH
c:0013 p---- s:0036 b:0036 l:000035 d:000035 CFUNC  :loop
c:0012 p:0009 s:0033 b:000024 d:000023 BLOCK /usr/local/lib/ruby/1.9.1/irb/ruby-lex.rb:229
c:0011 p---- s:0031 b:0031 l:000030 d:000030 FINISH
c:0009 p:0023 s:0025 b:0025 l:000024 d:000024 METHOD /usr/local/lib/ruby/1.9.1/irb/ruby-lex.rb:228
c:0008 p:0046 s:0022 b:001e3c d:001e3c METHOD /usr/local/lib/ruby/1.9.1/irb.rb:155
c:0006 p---- s:0017 b:0017 l:000016 d:000016 FINISH
c:0005 p---- s:0015 b:0015 l:000014 d:000014 CFUNC  :catch
c:0004 p:0183 s:0011 b:0011 l:00076c d:00076c METHOD /usr/local/lib/ruby/1.9.1/irb.rb:69
c:0003 p:0039 s:0006 b:0006 l:0000dc d:0000dc EVAL  /usr/local/bin/irb:12
c:0002 p---- s:0004 b:0004 l:00003 d:00003 FINISH
c:0001 p:0000 s:0002 b:0002 l:0000dc d:0000dc TOP

-- Ruby level backtrace information ------------------------------------------
/usr/local/bin/irb:12:in <main>
/usr/local/lib/ruby/1.9.1/irb.rb:69:in start'
/usr/local/lib/ruby/1.9.1/irb.rb:69:in catch'
/usr/local/lib/ruby/1.9.1/irb.rb:70:in block in start'
/usr/local/lib/ruby/1.9.1/irb.rb:155:in eval_input'
/usr/local/lib/ruby/1.9.1/irb/ruby-lex.rb:228:in reach_top_level_statement'
/usr/local/lib/ruby/1.9.1/irb/ruby-lex.rb:228:in catch'
/usr/local/lib/ruby/1.9.1/irb/ruby-lex.rb:229:inblock in each_top_level_statement'
/usr/local/lib/ruby/1.9.1/irb/ruby-lex.rb:229:in loop'
/usr/local/lib/ruby/1.9.1/irb/ruby-lex.rb:243:inblock (2 levels) in each_top_level_statement'
/usr/local/lib/ruby/1.9.1/irb.rb:156:inblock in eval_input'
/usr/local/lib/ruby/1.9.1/irb.rb:273:in signal_status'
/usr/local/lib/ruby/1.9.1/irb.rb:159:inblock (2 levels) in eval_input'
/usr/local/lib/ruby/1.9.1/irb/context.rb:254:in evaluate'
/usr/local/lib/ruby/1.9.1/irb/context.rb:167:in set_last_value'

08/07/2021 1/5
You may have encountered a bug in the Ruby interpreter or extension libraries. Bug reports are welcome. For details: http://www.ruby-lang.org/bugreport.html

Aborted

The closest I've come to reproducing outside of IRB is with:

```
buf=""
GC.stress=true
p syscall(106, '/etc/hosts', buf)
```

The GC.stress call isn't necessary, but increases the frequency of the crash. The above gives:

[BUG] Segmentation fault
ruby 1.9.3dev (2010-06-13 trunk 28312) [i686-linux]
Aborted

On another run of a similar script, without GC.stress, I got:

*** glibc detected *** ruby: free(): invalid next size (normal): 0x097d00d0 ***

======= Backtrace: ========
/lib/tls/i686/cmov/libc.so.6(+0x6b591)[0x17b591]
/lib/tls/i686/cmov/libc.so.6(+0x6cde8)[0x17cde8]
/lib/tls/i686/cmov/libc.so.6(cfree+0x6d)[0x17fecc]
/ruby(rb_objspace_free+0x7c)[0x806792c]
ruby(ruby_vm_destruct+0xcf)[0x815150f]
ruby(ruby_cleanup+0x1df)[0x805facf]
ruby(ruby_run_node+0x3a)[0x805fca]
ruby(main+0x5a)[0x805d6ca]
/lib/tls/i686/cmov/libc.so.6(__libc_start_main+0xe6)[0xc84bd6]
ruby([0x805d6d1]

[NOTE]
You may have encountered a bug in the Ruby interpreter or extension libraries.
Bug reports are welcome.
For details: http://www.ruby-lang.org/bugreport.html
Ruby does not stop you to shoot your foot. I believe this is not a bug.

Indeed. syscall expected buffer of sizeof(struct stat), but you gave it an empty string, so as natural consequence syscall overwrote memory region, and segmentation fault was happened. When you use syscall,
it's fundamentally C programming with Ruby syntax, so you can do anything as bad as C can. Same rule applied to dl and ffi.

I will restrict syscall to require $SAFE=0, just for fool proof.

matz.

#3 - 06/20/2010 12:05 AM - matz (Yukihiro Matsumoto)
=begin
Hi,

In message "Re: [ruby-core:30826] Re: [Bug #3454] Segfault with syscall" on Sun, 20 Jun 2010 00:01:32 +0900, Yukihiro Matsumoto matz@ruby-lang.org writes:

I will restrict syscall to require $SAFE=0, just for fool proof.

Oops, it has already restricted to $SAFE<2.

matz.
=end

#4 - 06/20/2010 03:37 AM - shyouhei (Shyouhei Urabe)
- Status changed from Open to Rejected
=begin
=end