Ruby master - Bug #5237

IO.copy_stream calls #read on an object infinitely many times

08/27/2011 11:58 AM - brixen (Brian Shirai)

<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></td>
</tr>
<tr>
<td>Target version:</td>
<td></td>
</tr>
<tr>
<td>ruby -v:</td>
<td></td>
</tr>
<tr>
<td>Backport:</td>
<td></td>
</tr>
</tbody>
</table>

Description
The code for IO.copy_stream appears to support passing any object that implements #readpartial or #read. For example, passing an instance of StringIO works just fine:

```ruby
sasha:rubinius2.0 brian$ irb
ruby-1.9.2-p290 :001 > puts RUBY_DESCRIPTION
ruby 1.9.2p290 (2011-07-09 revision 32553) [x86_64-darwin10.8.0]
=> nil
ruby-1.9.2-p290 :002 > require 'stringio'
=> true
ruby-1.9.2-p290 :003 > s = StringIO.new "this is a test"
=> StringIO:0x00000100851190
ruby-1.9.2-p290 :004 > IO.copy_stream s, "copy_stream_stringio"
=> 14
ruby-1.9.2-p290 :005 > IO.read "copy_stream_stringio"
=> "this is a test"
ruby-1.9.2-p290 :006 >
```

However, passing an object that implements eg #read according to the specification of IO#read, causes IO.copy_stream to endlessly call #read on the object:

```ruby
class A
  def initialize
    @count = 0
  end

  def read(bytes, buffer)
    print "read: #{@count += 1}\r"
    buffer.replace "this is a test"
    return nil
  end
end

a = A.new
IO.copy_stream a, "copy_stream_test"
```

```
ruby 1.9.2p290 (2011-07-09 revision 32553) [x86_64-darwin10.8.0]
copy_stream_bug.rb:7:in write': Interrupt
  from copy_stream_bug.rb:7:in print'
  from copy_stream_bug.rb:7:in read'
  from copy_stream_bug.rb:14:in copy_stream'
  from copy_stream_bug.rb:14:in ''
```

Thanks,
Brian

Associated revisions
Revision a28fe36c - 10/23/2011 11:19 PM - akr (Akira Tanaka)

- io.c (copy_stream_fallback_body): check nil for EOF of read method. patch by Eric Wong. [ruby-core:39134] [Bug #5237]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@33515 b2dd03c8-39d4-4d8f-98ff-823fe69b080e
Revision 33515 - 10/23/2011 11:19 PM - akr (Akira Tanaka)

io.c (copy_stream_fallback_body): check nil for EOF of read method. patch by Eric Wong. [ruby-core:39134] [Bug #5237]

Revision 33515 - 10/23/2011 11:19 PM - akr (Akira Tanaka)

io.c (copy_stream_fallback_body): check nil for EOF of read method. patch by Eric Wong. [ruby-core:39134] [Bug #5237]

Revision 33515 - 10/23/2011 11:19 PM - akr (Akira Tanaka)

io.c (copy_stream_fallback_body): check nil for EOF of read method. patch by Eric Wong. [ruby-core:39134] [Bug #5237]

Revision 33515 - 10/23/2011 11:19 PM - akr (Akira Tanaka)

io.c (copy_stream_fallback_body): check nil for EOF of read method. patch by Eric Wong. [ruby-core:39134] [Bug #5237]

Revision 33515 - 10/23/2011 11:19 PM - akr (Akira Tanaka)

io.c (copy_stream_fallback_body): check nil for EOF of read method. patch by Eric Wong. [ruby-core:39134] [Bug #5237]

Revision 33515 - 10/23/2011 11:19 PM - akr (Akira Tanaka)

io.c (copy_stream_fallback_body): check nil for EOF of read method. patch by Eric Wong. [ruby-core:39134] [Bug #5237]

Revision 33515 - 10/23/2011 11:19 PM - akr (Akira Tanaka)

io.c (copy_stream_fallback_body): check nil for EOF of read method. patch by Eric Wong. [ruby-core:39134] [Bug #5237]

Revision 33515 - 10/23/2011 11:19 PM - akr (Akira Tanaka)

io.c (copy_stream_fallback_body): check nil for EOF of read method. patch by Eric Wong. [ruby-core:39134] [Bug #5237]

Revision 33515 - 10/23/2011 11:19 PM - akr (Akira Tanaka)

io.c (copy_stream_fallback_body): check nil for EOF of read method. patch by Eric Wong. [ruby-core:39134] [Bug #5237]

History

#1 - 08/27/2011 07:54 PM - normalperson (Eric Wong)
- File 0001-avoid-infinite-loop-of-IO.copy_stream.patch added

The class that implements an IO#read-like method without clearing the destination buffer on EOF is arguably broken, but infinite looping is bad, I think.

The attached simple patch should fix the issue. Also available via git:

git pull git://bogomips.org/ruby.git copy-stream-infinite-loop

#2 - 08/28/2011 02:53 AM - brixen (Brian Shirai)

On Sat, Aug 27, 2011 at 3:54 AM, Eric Wong normalperson@yhlb.net wrote:

Issue #5237 has been updated by Eric Wong.

File 0001-avoid-infinite-loop-of-IO.copy_stream.patch added

The class that implements an IO#read-like method without clearing the destination buffer on EOF is arguably broken, but infinite looping is bad, I think.

Ah, you are right, it really should replace buffer with "" (according to the code) and return nil according to the docs. Seems more than a little weird, however, to empty the buffer to signal EOF, especially since the buffer argument is optional.

Cheers,
Brian

The attached simple patch should fix the issue. Also available via git:

git pull git://bogomips.org/ruby.git copy-stream-infinite-loop

Bug #5237: IO.copy_stream calls #read on an object infinitely many times
http://redmine.ruby-lang.org/issues/5237

Author: Brian Ford
Status: Open
Priority: Normal
Assignee:
Category: core
The code for IO.copy_stream appears to support passing any object that implements #readpartial or #read. For example, passing an instance of StringIO works just fine:

```
sasha:rubinius2.0 brian$ irb
ruby-1.9.2-p290 :001 > puts RUBY_DESCRIPTION
```

---

**#3 - 08/28/2011 02:59 PM - kosaki (Motohiro KOSAKI)**

- ruby -v changed from ruby 1.9.2p290 (2011-07-09 revision 32553) [x86_64-darwin10.8.0] to -

The class that implements an IO#read-like method without clearing the destination buffer on EOF is arguably broken, but infinite looping is bad, I think.

Ah, you are right, it really should replace buffer with "" (according to the code) and return nil according to the docs. Seems more than a little weird, however, to empty the buffer to signal EOF, especially since the buffer argument is optional.

I'm curious. Which doc do you talk about?

---

**#4 - 08/30/2011 07:53 AM - brixen (Brian Shirai)**

On Sat, Aug 27, 2011 at 10:55 PM, KOSAKI Motohiro <kosaki.motohiro@gmail.com> wrote:

The class that implements an IO#read-like method without clearing the destination buffer on EOF is arguably broken, but infinite looping is bad, I think.

Ah, you are right, it really should replace buffer with "" (according to the code) and return nil according to the docs. Seems more than a little weird, however, to empty the buffer to signal EOF, especially since the buffer argument is optional.

I'm curious. Which doc do you talk about?

The RDoc, which refers to returning nil or "" depending on what arguments are passed. For the case here, it does not matter what value is returned. The only way to not loop infinitely is to set the buffer to "".

---

**#5 - 08/31/2011 12:48 PM - kosaki (Motohiro KOSAKI)**

If you are reading following paragraph,

- At end of file, it returns nil or ""
- depend on length.
- ios.read() and
- ios.read(nil) returns "".
- ios.read(positive-integer) returns nil.

you have to read following paragraph too, I think.

- If length is a positive integer, it try to read length bytes without any conversion (binary mode).
- It returns nil or a string whose length is 1 to length bytes.
- nil means it met EOF at beginning.

---

3/5
Thus, positive length destination buffer and returning nil are exclusive. Now, I don't think we need doc fix.

but I'm not talking about a behavior change. I have no strong opinion. It's another story. I'm waiting akr-san's response.

Hi,

On Tue, Aug 30, 2011 at 8:48 PM, Motohiro KOSAKI

wrote:

---

Issue #5237 has been updated by Motohiro KOSAKI.

If you are reading following paragraph,

- At end of file, it returns nil or ""
- depend on length.
- ios.read() and
- ios.read(nil) returns ""
- ios.read(positive-integer) returns nil.

you have to read following paragraph too, I think.

- If length is a positive integer,
- it try to read length bytes without any conversion (binary mode).
- It returns nil or a string whose length is 1 to length bytes.
- nil means it met EOF at beginning.

---

I'm not following your explanation. The return value, whether nil or anything else, including an empty string or any string (one byte or not) will NOT cause the copy_stream method to return. It will loop infinitely unless the buffer that is passed in is set to "". In fact, I can return anything from #read as long as I set the buffer to "". This is bizarre.

At the least, the rdoc should note that to participate in this API, #read must set the buffer to "" to signal EOF. Still, in my opinion, that is a bad API. The return value should be the sole way to signal EOF given that the arguments to #read are optional.

---

This issue was solved with changeset r33515.

Brian, thank you for reporting this issue.

Your contribution to Ruby is greatly appreciated.

May Ruby be with you.

---

- io.c (copy_stream_fallback_body): check nil for EOF of read method. patch by Eric Wong. [ruby-core:39134] [Bug #5237]