Ruby trunk - Feature #14404

Adding writev support to IO#write_nonblock

01/26/2018 11:12 AM - janko (Janko Marohnić)

Status:	Open	
Priority:	Normal	
Assignee:		
Target version:		

Description

In Ruby 2.5 IO#write received writev support (https://github.com/ruby/ruby/commit/3efa7126e5e853f06cdd78d4d88837aeb72a9a3e), allowing it to accept multiple arguments and utilize writev when available.

Would it be possible to add this feature to IO#write_nonblock as well? IO#write_nonblock is used by the HTTP.rb and Socketry gems to implement their "write timeout" feature (the same way that IO#read_nonblock is used in Net::HTTP to implement "read timeout"). Since IO#write_nonblock doesn't yet support writev, at the moment it's not possible for HTTP.rb and Socketry to utilize writev when the "write timeout" is specified.

History

#1 - 01/27/2018 01:02 AM - normalperson (Eric Wong)

janko.marohnic@gmail.com wrote:

> Would it be possible to add this feature to IO#write_nonblock
> as well? IO#write_nonblock is used by the HTTP.rb and Socketry
> gems to implement their "write timeout" feature (the same way
> that IO#read_nonblock is used in Net::HTTP to implement "read
> timeout"). Since IO#write_nonblock doesn't yet support writev,
> at the moment it's not possible for HTTP.rb and Socketry to
> utilize writev when the "write timeout" is specified.

How ugly/tedious would it be for the users to deal with partial
writes to use write_nonblock?

It's a lot easier with IO#write because of the write-in-full
expectation, so no new strings get created; pointers just get
updated in C.

Fwiw, one longer-term idea is to integrate Timeout into the VM,
so internal rb_io_wait_*able calls can see the timeout and not
rely on being interrupted as with current timeout.rb.

#2 - 01/29/2018 12:29 AM - janko (Janko Marohnić)

> How ugly/tedious would it be for the users to deal with partial
> writes to use write_nonblock?

It does take a bit of work, but I believe the following code would do the job:

```
until chunks.empty?
  length = io.write_nonblock(*chunks)
  break unless chunks.sum(&:bytesize) > length
  while length > 0
    chunk = chunks.shift
    length -= chunk.bytesize
    chunks.unshift string.byteslice(length..-1) if length < 0
  end
end
```

I remembered now that HTTP.rb and Socketry would probably only utilize
writev on "Transfer-Encoding: chunked" requests, which probably aren't used
very often (you'd probably use that only when uploading a file of unknown
length).

> It's a lot easier with IO#write because of the write-in-full

> expectation, so no new strings get created; pointers just get
> updated in C.

I agree, it would be ideal to be able to always use IO#write.

> Fwiw, one longer-term idea is to integrate Timeout into the VM,
> so internal rb_io_wait_*able calls can see the timeout and not
> rely on being interrupted as with current timeout.rb.

That sounds great!