Ruby - Feature #9145

Queue#pop(true) return nil if empty instead of raising ThreadError

11/24/2013 12:42 PM - jsc (Justin Collins)

Status:	Open	
Priority:	Normal	
Assignee:		
Target version:		

Description

I propose the non-blocking form of Queue#pop behave like Array#pop and return nil when empty.

Current behavior is to raise a ThreadError, with a message indicating the queue is empty.

For example:

q = Queue.new
begin
loop do
next_item = q.pop(true)
end
rescue ThreadError

queue is empty...or maybe something bad happened

end

Instead, this could be

q = Queue.new
while next_item = q.pop(true)
end

Alternatively, raise an exception that is a subclass of ThreadError with a more specific name, such as "QueueEmpty". This would be a small improvement while remaining compatible with existing code.

History

#1 - 11/24/2013 01:30 PM - Glass_saga (Masaki Matsushita)

- Category changed from lib to ext

- Status changed from Open to Feedback

I think we can't change default behavior of Queue#pop(true) because some code expects ThreadError to be raised.
However, it may be possible to introduce new keyword argument like following:

q = Queue.new
while next_item = q.pop(true, exception: false) # it doesn't raise ThreadError and returns nil.

do something

end

#2 - 11/24/2013 01:53 PM - normalperson (Eric Wong)

"Glass_saga (Masaki Matsushita)" glass.saga@gmail.com wrote:

> I think we can't change default behavior of Queue#pop(true) because some code expects ThreadError to be raised.
> However, it may be possible to introduce new keyword argument like following:
>
> q = Queue.new
> while next_item = q.pop(true, exception: false) # it doesn't raise ThreadError and returns nil.
>
> # do something

end

+1 to that. All non-blocking methods (I/O or not) should support
exception: false to avoid (expensive/noisy-on-$DEBUG=$true) exceptions.

#3 - 11/25/2013 09:13 AM - jsc (Justin Collins)

Glass_saga (Masaki Matsushita) wrote:

> I think we can't change default behavior of Queue#pop(true) because some code expects ThreadError to be raised.
> However, it may be possible to introduce new keyword argument like following:
>
> q = Queue.new
> while next_item = q.pop(true, exception: false) # it doesn't raise ThreadError and returns nil.
>
> # do something
>
> end

That would work for me.

#4 - 11/25/2013 11:59 AM - Anonymous

On 11/23/2013 08:30 PM, Glass_saga (Masaki Matsushita) wrote:

> I think we can't change default behavior of Queue#pop(true) because some code expects ThreadError to be raised.
> However, it may be possible to introduce new keyword argument like following:
>
> q = Queue.new
> while next_item = q.pop(true, exception: false) # it doesn't raise ThreadError and returns nil.
>
> # do something
>
> end

Or what about a new method, Queue#pop?, which is always non-blocking and
non-raising. It would behave like:

```
class Queue
def pop?
pop(true)
rescue ThreadError
nil
end
end

q = Queue.new

q << 1
q << 2
q << 3

while x=q.pop?
p x
end
```

END
output:
1
2
3

#5 - 11/25/2013 03:17 PM - drbrain (Eric Hodel)

Note that the current behavior allows you to distinguish between a nil in the queue (returns nil) and no value in the queue (raises ThreadError)

#6 - 06/22/2017 07:23 AM - Glass_saga (Masaki Matsushita)

- Status changed from Feedback to Closed

Currently, Queue#pop takes non_block flag.

#7 - 06/22/2017 07:51 AM - normalperson (Eric Wong)

glass.saga@gmail.com wrote:

> Issue #9145 has been updated by Glass_saga (Masaki Matsushita).
>
> Status changed from Feedback to Closed
>
> Currently, Queue#pop takes non_block flag.

No, I don't think this should be closed.

I think Justin's point was:

Currently, it is impossible to know if a queue is closed
(permanent condition) or if it is empty (temporary condition).
So at the very least, a different exception should be raised:

Justin Collins wrote:
> Alternatively, raise an exception that is a subclass of
> ThreadError with a more specific name, such as "QueueEmpty".
> This would be a small improvement while remaining compatible
> with existing code.

On a side note, relying on exceptions for flow control has all
the same performance and $DEBUG noise problems it did with
IO#*_nonblock [ruby-core:38666] [Feature #5138]

But thinking of an efficient API for that is tricky :<

#8 - 06/28/2017 07:05 AM - Glass_saga (Masaki Matsushita)

- Status changed from Closed to Open

#9 - 01/01/2018 07:41 PM - uwe@kubosch.no (Uwe Kubosch)

How about a block form where the block is called with the popped element? The method would return false if called with non_block set to true if the queue is empty.

```
q = Queue.new
q << 1
q << 2
q << 3

while q.pop(true){|x| p x}
```