# Ruby master - Feature #11139

## [PATCH] socket: support accept `sock_nonblock: (true|false)'

05/12/2015 12:14 AM - normalperson (Eric Wong)

| | | |
|---|---|---|
| **Status:** | Feedback | |
| **Priority:** | Normal | |
| **Assignee:** | akr (Akira Tanaka) | |
| **Target version:** | | |

**Description**

An application wanting to do non-blocking accept may want to
create a blocking accepted socket, allow it with a kwarg while
preserving default behavior.

This is analogous to the SOCK_NONBLOCK flag in the Linux `accept4'
syscall.

While this has little obvious effect for Ruby API users (which
can emulate blocking behavior) this will reduce syscalls made
internally by Ruby.  Forcing blocking will preserve "wake-one"
behavior in the OS kernel to avoid a "thundering herd" problem.

In all cases, existing Ruby 2.2 behavior is preserved by default
to maximize compatibility, especially when sharing sockets with
non-Ruby processes:

accept' andsysaccept' calls will create sockets which are
blocking by default.

`accept_nonblock', calls will create sockets which are non-blocking
by default.

### History

#### #1 - 05/12/2015 12:25 AM - djberg96 (Daniel Berger)

How about just :block ?

```
a.accept(block: false)
a.accept_nonblock(block: true)
```

#### #2 - 05/12/2015 12:58 AM - normalperson (Eric Wong)

djberg96@gmail.com wrote:

> How about just :block ?
>
> ```
> a.accept(block: false)
> a.accept_nonblock(block: true)
> ```

I don't think that helps convey it affects the newly-accepted socket,
not the socket performing the accept.  Perhaps :sock_nonblock is better,
at least it maps to SOCK_NONBLOCK in Linux.

#### #3 - 06/14/2015 10:09 AM - akr (Akira Tanaka)

*- Status changed from Open to Feedback*

The status of nonblocking flag is not so important in Ruby
because most methods works regardless of the flag.
(Nonblocking methods sets the flag internally.  Blocking methods retry on EAGAIN/EWOULDBLOCK internally.)

So the proposed kwarg is almost no-op at Ruby level.
It may reduce system calls, though.

I'm not sure the kwarg is worth enough to provide at Ruby level.

But, if it is provided, it should have descriptive name.
"nonblock" and "block" is too short.

#### #4 - 06/15/2015 11:48 PM - normalperson (Eric Wong)

akr@fsij.org wrote:

> So the proposed kwarg is almost no-op at Ruby level.

Agree.

> It may reduce system calls, though.

Yes.  Also, real blocking send/recv family calls without select/ppoll
allows for exclusive "wake-one" behavior to avoid thundering herds on
wakeup.

Because there is no thundering herd, exclusive wakeup also improves
fairness if multiple threads/processes share the socket for packetized
I/O

> I'm not sure the kwarg is worth enough to provide at Ruby level.
> But, if it is provided, it should have descriptive name.
> "nonblock" and "block" is too short.

Perhaps "sock_nonblock" to match the Linux flag name? (SOCK_NONBLOCK)

#### #5 - 06/16/2015 01:53 AM - akr (Akira Tanaka)

"sock_nonblock kwarg is similar to SOCK_NONBLOCK flag in (future) POSIX" seems intuitive.

Note that accept4 (and its behavior with SOCK_NONBLOCK) will be defined in POSIX (or SUS):
http://austingroupbugs.net/view.php?id=411

It means that O_NONBLOCK flag of the created FD is set if sock_nonblock is true and
O_NONBLOCK flag is unset if sock_nonblock is false.

The behavior should work on the platforms which doesn't have accept4().
Be careful that O_NONBLOCK behavior of accept() is not portable.
http://cr.yp.to/docs/unixport.html

It is not clear that the behavior when sock_nonblock kwarg is not specified.
I feel that it should be defined anyway.
It may reduce the portability problem a bit.

#### #6 - 06/16/2015 03:35 AM - akr (Akira Tanaka)

Akira Tanaka wrote:

> It is not clear that the behavior when sock_nonblock kwarg is not specified.
> I feel that it should be defined anyway.
> It may reduce the portability problem a bit.

"sock_nonblock: false" would be better for the default behavior.
It will reduce system calls: it avoids poll() for each blocking read operations and it adds only one fcntl(F_SETFL) for the first nonblocking operation.
It is consistent with accept4().

#### #7 - 06/22/2015 07:38 PM - normalperson (Eric Wong)

I don't think the flag matters for second-tier OSes, as the underlying
descriptor does not matter with our current APIs.

So better to leave the OS default if sock_nonblock kwarg is not
specified.

#### #8 - 06/25/2015 02:29 AM - akr (Akira Tanaka)

The default behavior of accept4() is fine.

Although most Ruby-level API is nonblocking flag insensitive,
there is nonblocking flag sensitive API such as IO#syswrite.

Also, the flag affects other (non-Ruby) programs when the FD is inherited to them.
Especially stdio doesn't work well on FD with nonblocking flag.

I think setting or clearing nonblocking flag after accept() reduces unexpected behaviors.

#### #9 - 07/02/2015 01:30 AM - normalperson (Eric Wong)

*- File socket-support-accept-sock_nonblock-true-false-v2.patch added*

*- Subject changed from [PATCH] socket: accept_nonblock supports "nonblock: false" kwarg to [PATCH] socket: support accept `sock_nonblock: (true|false)'*

*- Description updated*

I think we need to preserve existing behavior with accept_nonblock in case
there is code which shares accepted FDs with non-Ruby processes (or even
passes it to a C extension).

### Files

| | | | |
|---|---|---|---|
| 0002-socket-accept_nonblock-supports-nonblock-false-kwarg.patch | 2.94 KB | 05/12/2015 | normalperson (Eric Wong) |
| socket-support-accept-sock_nonblock-true-false-v2.patch | 20.8 KB | 07/02/2015 | normalperson (Eric Wong) |