Ruby master - Feature #4531

[PATCH 0/7] use poll() instead of select() in certain cases

03/28/2011 11:48 AM - normalperson (Eric Wong)

<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>kosaki (Motohiro KOSAKI)</td>
</tr>
<tr>
<td>Target version:</td>
<td>2.0.0</td>
</tr>
</tbody>
</table>

**Description**

```ruby
=endef: [ruby-core:35527]

This adds a new C API function with the following prototype:

```ruby
rb_io_poll_fd(int fd, short events, int timeout);
```

It is emulated using select() for platforms that we do not support poll() for. It is much easier to use than rb_thread_select() and rb_thread_fd_select() for the common case in C extensions[1].

For Linux (and eventually any other platforms where poll() works for all select()-able files), we actually implement rb_io_poll_fd() using the poll() system call which means it is faster for high numbered file descriptors and does not put malloc pressure on the garbage collector.

Lastly, since IO.select() is commonly used with a single IO object in my experience, we will try to use rb_io_poll_fd() in that case.

There is also a new testcase for io/wait since I needed to verify my changes to ext/io/wait.c were correct.

No failures were introduced to test-all and test-rubyspec targets with either the select() or poll()-based implementation of rb_io_poll_fd() on my platform (Linux x86_64)

[1] see patches for changes I made in ext/socket/init.c, ext/io/wait.c, and ext/readline/readline.c:

```bash
$ git diff --stat origin/trunk -- ext
ext/io/wait/wait.c |   34 +++-------------------
ext/readline/readline.c |    6 +---
ext/socket/init.c |   72 ++++++++---------------------------------------
3 files changed, 18 insertions(+), 94 deletions(-)
```

=end

**Associated revisions**

Revision b48537cd - 04/29/2011 07:09 PM - kosaki (Motohiro KOSAKI)

- test/io/wait/test_io_wait.rb: New. for testing ext/io/wait. the patch was written by Eric Wong. [Feature #4531]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@31389 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 31389 - 04/29/2011 07:09 PM - kosaki (Motohiro KOSAKI)

- test/io/wait/test_io_wait.rb: New. for testing ext/io/wait. the patch was written by Eric Wong. [Feature #4531]

Revision 31389 - 04/29/2011 07:09 PM - kosaki (Motohiro KOSAKI)

- test/io/wait/test_io_wait.rb: New. for testing ext/io/wait. the patch was written by Eric Wong. [Feature #4531]
revision 31389 - 04/29/2011 07:09 PM - kosaki (Motohiro KOSAKI)

* test/io/wait/test_io_wait.rb: New. for testing ext/io/wait. the patch was written by Eric Wong. [Feature #4531]

revision 31389 - 04/29/2011 07:09 PM - kosaki (Motohiro KOSAKI)

* test/io/wait/test_io_wait.rb: New. for testing ext/io/wait. the patch was written by Eric Wong. [Feature #4531]

revision 31389 - 04/29/2011 07:09 PM - kosaki (Motohiro KOSAKI)

* test/io/wait/test_io_wait.rb: New. for testing ext/io/wait. the patch was written by Eric Wong. [Feature #4531]

revision 31389 - 04/29/2011 07:09 PM - kosaki (Motohiro KOSAKI)

* test/io/wait/test_io_wait.rb: New. for testing ext/io/wait. the patch was written by Eric Wong. [Feature #4531]

revision 568c9343 - 04/30/2011 07:16 AM - kosaki (Motohiro KOSAKI)

* benchmark/bm_io_select.rb, benchmark/bm_io_select2.rb: New. based on a patch from Eric Wong at [Feature #4531]
git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@31390 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

revision 31390 - 04/30/2011 07:16 AM - kosaki (Motohiro KOSAKI)

* benchmark/bm_io_select.rb, benchmark/bm_io_select2.rb: New. based on a patch from Eric Wong at [Feature #4531]

revision 31390 - 04/30/2011 07:16 AM - kosaki (Motohiro KOSAKI)

* benchmark/bm_io_select.rb, benchmark/bm_io_select2.rb: New. based on a patch from Eric Wong at [Feature #4531]

revision 31390 - 04/30/2011 07:16 AM - kosaki (Motohiro KOSAKI)

* benchmark/bm_io_select.rb, benchmark/bm_io_select2.rb: New. based on a patch from Eric Wong at [Feature #4531]

revision 31390 - 04/30/2011 07:16 AM - kosaki (Motohiro KOSAKI)

* benchmark/bm_io_select.rb, benchmark/bm_io_select2.rb: New. based on a patch from Eric Wong at [Feature #4531]

revision 31390 - 04/30/2011 07:16 AM - kosaki (Motohiro KOSAKI)

* benchmark/bm_io_select.rb, benchmark/bm_io_select2.rb: New. based on a patch from Eric Wong at [Feature #4531]

revision 31390 - 04/30/2011 07:16 AM - kosaki (Motohiro KOSAKI)

* benchmark/bm_io_select.rb, benchmark/bm_io_select2.rb: New. based on a patch from Eric Wong at [Feature #4531]

revision 31390 - 04/30/2011 07:16 AM - kosaki (Motohiro KOSAKI)

* benchmark/bm_io_select.rb, benchmark/bm_io_select2.rb: New. based on a patch from Eric Wong at [Feature #4531]

revision 31390 - 04/30/2011 07:16 AM - kosaki (Motohiro KOSAKI)

* benchmark/bm_io_select.rb, benchmark/bm_io_select2.rb: New. based on a patch from Eric Wong at [Feature #4531]

revision 9be37ca7 - 05/04/2011 12:59 AM - kosaki (Motohiro KOSAKI)

* thread.c (rb_wait_for_single_fd): new.

* thread.c (select_single): select(2) based backend for rb_wait_for_single_fd().

* io.c (make_writeconv): use rb_wait_for_single_fd() instead of rb_thread_fd_select().

* io.c (rb_io_wait_readable): ditto.

* thread.c (rb_thread_wait_fd_rw): ditto.

* io.c (wait_readable): removed.

* thread.c (init_set_fd): new helper function.
include/ruby/io.h (RB_WAITFD_IN, RB_WAITFD_PRI, RB_WAITFD_OUT): new constant for rb_single_wait_fd().

The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@31419 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 31419 - 05/04/2011 12:59 AM - kosaki (Motohiro KOSAKI)

- thread.c (rb_wait_for_single_fd): new.
- thread.c (select_single): select(2) based backend for rb_wait_for_single_fd().
- io.c (make_writeconv): use rb_wait_for_single_fd() instead of rb_thread_fd_select().
- io.c (rb_io_wait_readable): ditto.
- io.c (rb_thread_wait_fd_rw): ditto.
- io.c (wait_readable): removed.
- thread.c (init_set_fd): new helper function.
- include/ruby/io.h (RB_WAITFD_IN, RB_WAITFD_PRI, RB_WAITFD_OUT): new constant for rb_single_wait_fd().

The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]

Revision 31419 - 05/04/2011 12:59 AM - kosaki (Motohiro KOSAKI)

- thread.c (rb_wait_for_single_fd): new.
- thread.c (select_single): select(2) based backend for rb_wait_for_single_fd().
- io.c (make_writeconv): use rb_wait_for_single_fd() instead of rb_thread_fd_select().
- io.c (rb_io_wait_readable): ditto.
- thread.c (rb_thread_wait_fd_rw): ditto.
- io.c (wait_readable): removed.
- thread.c (init_set_fd): new helper function.
- include/ruby/io.h (RB_WAITFD_IN, RB_WAITFD_PRI, RB_WAITFD_OUT): new constant for rb_single_wait_fd().

The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]

Revision 31419 - 05/04/2011 12:59 AM - kosaki (Motohiro KOSAKI)

- thread.c (rb_wait_for_single_fd): new.
- thread.c (select_single): select(2) based backend for rb_wait_for_single_fd().
- io.c (make_writeconv): use rb_wait_for_single_fd() instead of rb_thread_fd_select().
- io.c (rb_io_wait_readable): ditto.
- thread.c (rb_thread_wait_fd_rw): ditto.
- io.c (wait_readable): removed.
- thread.c (init_set_fd): new helper function.
- include/ruby/io.h (RB_WAITFD_IN, RB_WAITFD_PRI, RB_WAITFD_OUT): new constant for rb_single_wait_fd().

The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]

Revision 31419 - 05/04/2011 12:59 AM - kosaki (Motohiro KOSAKI)

- thread.c (rb_wait_for_single_fd): new.
- thread.c (select_single): select(2) based backend for rb_wait_for_single_fd().
- io.c (make_writeconv): use rb_wait_for_single_fd() instaed of rb_thread_fd_select().
- io.c (rb_io_wait_readable): ditto.
- thread.c (rb_thread_wait_fd_rw): ditto.
- io.c (wait_readable): removed.
- thread.c (init_set_fd): new helper function.
- include/ruby/io.h (RB_WAITFD_IN, RB_WAITFD_PRI, RB_WAITFD_OUT): new constant for rb_single_wait_fd().

The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]

Revision 31419 - 05/04/2011 12:59 AM - kosaki (Motohiro KOSAKI)

- thread.c (rb_wait_for_single_fd): new.
- thread.c (select_single): select(2) based backend for rb_wait_for_single_fd().
- io.c (make_writeconv): use rb_wait_for_single_fd() instaed of rb_thread_fd_select().
- io.c (rb_io_wait_readable): ditto.
- thread.c (rb_thread_wait_fd_rw): ditto.
- io.c (wait_readable): removed.
- thread.c (init_set_fd): new helper function.
- include/ruby/io.h (RB_WAITFD_IN, RB_WAITFD_PRI, RB_WAITFD_OUT): new constant for rb_single_wait_fd().

The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]

Revision 31419 - 05/04/2011 12:59 AM - kosaki (Motohiro KOSAKI)

- thread.c (rb_wait_for_single_fd): new.
- thread.c (select_single): select(2) based backend for rb_wait_for_single_fd().
- io.c (make_writeconv): use rb_wait_for_single_fd() instaed of rb_thread_fd_select().
- io.c (rb_io_wait_readable): ditto.
- thread.c (rb_thread_wait_fd_rw): ditto.
- io.c (wait_readable): removed.
- thread.c (init_set_fd): new helper function.
- include/ruby/io.h (RB_WAITFD_IN, RB_WAITFD_PRI, RB_WAITFD_OUT): new constant for rb_single_wait_fd().

The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]

Revision 249fe0e7 - 05/04/2011 01:07 AM - kosaki (Motohiro KOSAKI)
- thread.c (rb_wait_for_single_fd): new. poll(2) based backend for rb_wait_for_single_fd(). Now only Linux uses it.
The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@31420 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 31420 - 05/04/2011 01:07 AM - kosaki (Motohiro KOSAKI)
- thread.c (rb_wait_for_single_fd): new. poll(2) based backend for rb_wait_for_single_fd(). Now only Linux uses it.
The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]

Revision 31420 - 05/04/2011 01:07 AM - kosaki (Motohiro KOSAKI)
- thread.c (rb_wait_for_single_fd): new. poll(2) based backend for rb_wait_for_single_fd(). Now only Linux uses it.
The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]

Revision 31420 - 05/04/2011 01:07 AM - kosaki (Motohiro KOSAKI)
- thread.c (rb_wait_for_single_fd): new. poll(2) based backend for rb_wait_for_single_fd(). Now only Linux uses it.
The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]

Revision 31420 - 05/04/2011 01:07 AM - kosaki (Motohiro KOSAKI)
- thread.c (rb_wait_for_single_fd): new. poll(2) based backend for rb_wait_for_single_fd(). Now only Linux uses it.
The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]

Revision 31420 - 05/04/2011 01:07 AM - kosaki (Motohiro KOSAKI)
- thread.c (rb_wait_for_single_fd): new. poll(2) based backend for rb_wait_for_single_fd(). Now only Linux uses it.
The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]

Revision 31420 - 05/04/2011 01:07 AM - kosaki (Motohiro KOSAKI)
- thread.c (rb_wait_for_single_fd): new. poll(2) based backend for rb_wait_for_single_fd(). Now only Linux uses it.
The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]

Revision 31420 - 05/04/2011 01:07 AM - kosaki (Motohiro KOSAKI)
- thread.c (rb_wait_for_single_fd): new. poll(2) based backend for rb_wait_for_single_fd(). Now only Linux uses it.
The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]

Revision cd796c60 - 05/04/2011 01:09 AM - kosaki (Motohiro KOSAKI)
- ext/io/wait/wait.c (io_wait): use rb_wait_for_single_fd(). The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]

Revision 31421 - 05/04/2011 01:09 AM - kosaki (Motohiro KOSAKI)
- ext/io/wait/wait.c (io_wait): use rb_wait_for_single_fd(). The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]

Revision 31421 - 05/04/2011 01:09 AM - kosaki (Motohiro KOSAKI)
- ext/io/wait/wait.c (io_wait): use rb_wait_for_single_fd(). The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]
Revision 31421 - 05/04/2011 01:09 AM - kosaki (Motohiro KOSAKI)

• ext/io/wait/wait.c (io_wait): use rb_wait_for_single_fd(). The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]

Revision 31421 - 05/04/2011 01:09 AM - kosaki (Motohiro KOSAKI)

• ext/io/wait/wait.c (io_wait): use rb_wait_for_single_fd(). The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]

Revision 31421 - 05/04/2011 01:09 AM - kosaki (Motohiro KOSAKI)

• ext/io/wait/wait.c (io_wait): use rb_wait_for_single_fd(). The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]

Revision 31421 - 05/04/2011 01:09 AM - kosaki (Motohiro KOSAKI)

• ext/io/wait/wait.c (io_wait): use rb_wait_for_single_fd(). The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]

Revision 82b2e801 - 05/04/2011 01:12 AM - kosaki (Motohiro KOSAKI)

• ext/socket/init.c (wait_connectable): use rb_wait_for_single_fd().
The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]

• ext/socket/init.c (try_wait_connectable, wait_connectable_ensure): removed.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@31422 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 31422 - 05/04/2011 01:12 AM - kosaki (Motohiro KOSAKI)

• ext/socket/init.c (wait_connectable): use rb_wait_for_single_fd().
The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]

• ext/socket/init.c (try_wait_connectable, wait_connectable_ensure): removed.

Revision 31422 - 05/04/2011 01:12 AM - kosaki (Motohiro KOSAKI)

• ext/socket/init.c (wait_connectable): use rb_wait_for_single_fd().
The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]

• ext/socket/init.c (try_wait_connectable, wait_connectable_ensure): removed.

Revision 31422 - 05/04/2011 01:12 AM - kosaki (Motohiro KOSAKI)

• ext/socket/init.c (wait_connectable): use rb_wait_for_single_fd().
The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]

• ext/socket/init.c (try_wait_connectable, wait_connectable_ensure): removed.

Revision 31422 - 05/04/2011 01:12 AM - kosaki (Motohiro KOSAKI)

• ext/socket/init.c (wait_connectable): use rb_wait_for_single_fd().
The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]

• ext/socket/init.c (try_wait_connectable, wait_connectable_ensure): removed.
- ext/socket/init.c (wait_connectable): use rb_wait_for_single_fd(). The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]
- ext/socket/init.c (try_wait_connectable, wait_connectable_ensure): removed.

Revision 31422 - 05/04/2011 01:12 AM - kosaki (Motohiro KOSAKI)

- ext/socket/init.c (wait_connectable): use rb_wait_for_single_fd(). The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]
- ext/socket/init.c (try_wait_connectable, wait_connectable_ensure): removed.

Revision 31422 - 05/04/2011 01:12 AM - kosaki (Motohiro KOSAKI)

- ext/socket/init.c (wait_connectable): use rb_wait_for_single_fd(). The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]
- ext/socket/init.c (try_wait_connectable, wait_connectable_ensure): removed.

Revision 4da4cb04 - 05/04/2011 01:13 AM - kosaki (Motohiro KOSAKI)

- ext/readline/readline.c (readline_event): use rb_wait_for_single_fd(). The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@31423 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 31423 - 05/04/2011 01:13 AM - kosaki (Motohiro KOSAKI)

- ext/readline/readline.c (readline_event): use rb_wait_for_single_fd(). The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]

Revision 31423 - 05/04/2011 01:13 AM - kosaki (Motohiro KOSAKI)

- ext/readline/readline.c (readline_event): use rb_wait_for_single_fd(). The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]

Revision 31423 - 05/04/2011 01:13 AM - kosaki (Motohiro KOSAKI)

- ext/readline/readline.c (readline_event): use rb_wait_for_single_fd(). The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]

Revision 31423 - 05/04/2011 01:13 AM - kosaki (Motohiro KOSAKI)

- ext/readline/readline.c (readline_event): use rb_wait_for_single_fd(). The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]

Revision 31423 - 05/04/2011 01:13 AM - kosaki (Motohiro KOSAKI)

- ext/readline/readline.c (readline_event): use rb_wait_for_single_fd(). The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]

Revision 31423 - 05/04/2011 01:13 AM - kosaki (Motohiro KOSAKI)

- ext/readline/readline.c (readline_event): use rb_wait_for_single_fd(). The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]

Revision 31423 - 05/04/2011 01:13 AM - kosaki (Motohiro KOSAKI)

- ext/readline/readline.c (readline_event): use rb_wait_for_single_fd(). The patch was written by Eric Wong. [Ruby 1.9 - Feature #4531]

Revision 766c32ad - 05/30/2011 04:45 AM - yugui (Yuki Sonoda)

03/16/2020
merges r31389 and r31398 from trunk into ruby_1_9_2.

- test/io/wait/test_io_wait.rb: New. for testing ext/io/wait. the patch was written by Eric Wong. [Feature #4531] -- fix commit mistake of r31389.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_1_9_2@31821 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 03/28/2011 09:23 PM - kosaki (Motohiro KOSAKI)

=begin
ref: [ruby-core:35527]
This adds a new C API function with the following prototype:

Â rb_io_poll_fd(int fd, short events, int timeout);

It is emulated using select() for platforms that we do not support poll() for. Â It is much easier to use than rb_thread_select() and rb_thread_fd_select() for the common case in C extensions[1].

For Linux (and eventually any other platforms where poll() works for all select()-able files), we actually implement rb_io_poll_fd() using the poll() system call which means it is faster for high numbered file descriptors and does not put malloc pressure on the garbage collector.

Meta review comment. All performance patches should have measured performance number.

And, if you really want to care C10K scalability issue, why don't you use epoll?

Lastly, since IO.select() is commonly used with a single IO object in my experience, we will try to use rb_io_poll_fd() in that case.

There is also a new testcase for io/wait since I needed to verify my changes to ext/io/wait.c were correct.

No failures were introduced to test-all and test-rubyspec targets with either the select() or poll()-based implementation of rb_io_poll_fd() on my platform (Linux x86_64)
=begin

#2 - 03/29/2011 01:59 AM - normalperson (Eric Wong)

- File io_select_using_poll_test.rb added

=begin
Added a contrived test case, using a high RLIMIT_NOFILE (e.g. ulimit -n 16384) is recommended to make the results more apparent.
=begin

#3 - 03/29/2011 02:23 AM - normalperson (Eric Wong)

=begin
KOSAKI Motohiro kosaki.motohiro@gmail.com wrote:

ref: [ruby-core:35527]
This adds a new C API function with the following prototype:

rb_io_poll_fd(int fd, short events, int timeout);

It is emulated using select() for platforms that we do not support poll() for. It is much easier to use than rb_thread_select() and rb_thread_fd_select() for the common case in C extensions[1].

For Linux (and eventually any other platforms where poll() works for all select()-able files), we actually implement rb_io_poll_fd() using the poll() system call which means it is faster for high numbered file descriptors and does not put malloc pressure on the garbage collector.

=begin
Meta review comment. All performance patches should have measured performance number.

Based on a contrived test case: 
http://redmine.ruby-lang.org/attachments/1562/io_select_using_poll_test.rb

I get the following results:

---- before NOFILE=1024 ----
max fd: 1024 (results not apparent with <= 1024 max fd)
last IO: #
  2.050000   1.440000   3.490000 ( 3.476264)

GC.count: 386

---- after NOFILE=1024 ----
max fd: 1024 (results not apparent with <= 1024 max fd)
last IO: #
  2.430000   0.320000   2.750000 ( 2.734643)

GC.count: 343

---- before NOFILE=16384 ----
max fd: 16384 (results not apparent with <= 1024 max fd)
last IO: #
  3.610000   5.680000   9.290000 ( 9.266017)

GC.count: 643

---- after NOFILE=16384 ----
max fd: 16384 (results not apparent with <= 1024 max fd)
last IO: #
  2.970000   0.450000   3.420000 ( 3.415426)

GC.count: 394

I don't have real world results/test case but I think small bits like this count for the future.

  And, if you really want to care C10K scalability issue, why don't you use epoll?

I assume you mean Ruby programming in general and not making MRI itself use epoll().

I find synchronous programming easier especially when dealing with existing libraries (I use Ruby to make programming life easier :)

For me, thousands of native threads (NPTL + 64-bit) often makes more sense than epoll() + non-blocking I/O.

Concurrent connections isn't always an issue, either, but also sometimes have many open file handles to support other tasks[1].

I mainly think select() is a horrible interface and having extension authors to deal with HAVE_RB_FD_INIT and remembering rb_fd_term() is dangerous. Small bits of pressure from Rubyists can hopefully steer other OSes towards better poll() support.

Thank you for your time!

[1] DB connection pool, large memcached pool, regular files, various message queues, etc...

--
Eric Wong
=end

#4 - 03/29/2011 09:07 PM - kosaki (Motohiro KOSAKI)
- Assignee set to kosaki (Motohiro KOSAKI)

=end
KOSAKI Motohiro wrote:

ref: [ruby-core:35527]

This adds a new C API function with the following prototype:

```c
rb_io_poll_fd(int fd, short events, int timeout);
```

It is emulated using `select()` for platforms that we do not support `poll()` for. It is much easier to use than `rb_thread_select()` and `rb_thread_fd_select()` for the common case in C extensions[1].

For Linux (and eventually any other platforms where `poll()` works for all `select()`-able files), we actually implement `rb_io_poll_fd()` using the `poll()` system call which means it is faster for high numbered file descriptors and does not put malloc pressure on the garbage collector.

Meta review comment. All performance patches should have measured performance number.

Based on a contrived test case:

http://redmine.ruby-lang.org/attachments/1562/io_select_using_poll_test.rb

I get the following results:

---- before NOFILE24 ----
max fd: 1024 (results not apparent with <

I mainly think `select()` is a horrible interface and having extension authors to deal with HAVE_RB_FD_INIT and remembering rb_fd_term() is dangerous. Small bits of pressure from Rubyists can hopefully steer other OSes towards better `poll()` support.

I'm sorry. I haven't catch this. can you please explain more detail?

rb_fd_init() uses malloc() to create fdsets instead of traditional stack allocation. This also requires using rb_fd_term() to free() memory, so rb_ensure() must be used[1]. Often this is for waiting on a single fd, so the `rb_io_poll_fd()` interface is better for extension authors all around even if it internally uses select().

As for `poll()` support, Evan Phoenix pointed out `poll()` doesn't work on OS X with ttys/pipes (ref: [ruby-core:35529]). I don't know about other kernels. Ideally, `poll()` should work everywhere `select()` does (and `select()` should be deprecated). I would like to encourage other OSes to get where Linux is (or get more people using Linux :).

On a related note:

Ruby also still needlessly checks for read/writability with `select()`[2] even though though we have support for blocking regions. I think that is safe to remove if we check return values with `rb_io_wait_{read,writable}()`. One part of
http://redmine.ruby-lang.org/issues/4535 does exactly this but I have another patch to kill more unnecessary `select()` calls that aren't bugs, just extra code/syscalls.

[1] though from reading through `do_select()` in thread.c, I don't think `do_select()` can raise..

[2] via `rb_thread_wait_fd()/rb_thread_fd_writable()`

--
Eric Wong
=end

#7 - 03/30/2011 03:29 AM - normalperson (Eric Wong)
=begin
Eric Wong normalperson@yhbt.net wrote:

On a related note:

Ruby also still needlessly checks for read/writability with `select()` even though though we have support for blocking regions. I think that is safe to remove if we check return values with `rb_io_wait_(read,writable)`. One part of http://redmine.ruby-lang.org/issues/4535 does exactly this but I have another patch to kill more unnecessary `select()` calls that aren't bugs, just extra code/syscalls.

Just opened this one: http://redmine.ruby-lang.org/issues/4538

--
Eric Wong
=end

#8 - 04/30/2011 03:46 AM - kosaki (Motohiro KOSAKI)
=begin
commnet for patch 0001.

begin
require 'io/wait'
 rescue LoadError
 end

should be

begin
require 'io/wait'
 rescue LoadError
 # skip this test.
 return
 end

If we can't load io/wait, why do we need to run the tests for it?
Anyway, I'll commit it.
=end

#9 - 04/30/2011 08:23 AM - normalperson (Eric Wong)
=begin
Motohiro KOSAKI kosaki.motohiro@gmail.com wrote:

Issue #4531 has been updated by Motohiro KOSAKI.

commnet for patch 0001.

begin
require 'io/wait'
 rescue LoadError
 end

should be

begin
require 'io/wait'
rescue LoadError
  # skip this test.
  return
end

If we can't load io/wait, why do we need to run the tests for it?
Anyway, I'll commit it.

I used a condition at the end of the class definition:

end if IO.method_defined?(:wait)

I copied the style from test/io/nonblock/test_flush.rb

--
Eric Wong
=end

#10 - 04/30/2011 04:22 PM - kosaki (Motohiro KOSAKI)
=begin
I also committed io_select_using_poll_test.rb (but modified a little) at r31390.
=end

#11 - 05/01/2011 02:23 AM - normalperson (Eric Wong)
KOSAKI Motohiro kosaki.motohiro@gmail.com wrote:

I'd suggest to make generic single fd waiting abstract function, like below.
  rb_wait_for_single_fd(int fd, int events, struct timeval *tv);

Also, RB_POLLIN should be avoided.

Huh? What would I use for events instead?

Thanks for taking your time to review this!

--
Eric Wong

#12 - 05/01/2011 12:23 PM - kosaki (Motohiro KOSAKI)

Also, RB_POLLIN should be avoided.

Huh? What would I use for events instead?

Thanks for taking your time to review this!

I meant RB_POLLIN is not backend independ name.

thanks.

#13 - 05/03/2011 07:45 AM - normalperson (Eric Wong)
- File rb_wait_for_single_fd.mbox added

I've updated the series with your suggestions and rebased against trunk r31412 and uploaded a new
mbox with the 6 following patches:

1. rb_wait_for_single_fd: initial implementation
2. io/wait: switch to rb_wait_for_single_fd()
3. ext/socket/init.c: simplify wait_connectable() using rb_wait_for_single_fd
4. readline: use rb_wait_for_single_fd() instead of rb_thread_fd_select()
5. io.c (IO.select): use rb_wait_for_single_fd for single IO case
6. rb_wait_for_single_fd: use poll() on Linux

Also available in my git repo:
git pull git://bogomips.org/ruby rb_wait_for_single_fd

03/16/2020
Please do NOT send untested patch. :-/
% ./ruby-single-wait benchmark/bm_io_select3.rb
max fd: 100000 (results not apparent with <= 1024 max fd)
benchmark/bm_io_select3.rb:15: [BUG] Segmentation fault
ruby 1.9.3dev (2011-05-03 trunk 31412) [x86_64-linux]

-- Control frame information -----------------------------------------------
 benchmark/bm_io_select3.rb:15


c:0005 p:---- s:0014 b:0014 l:000013 d:000013 FINISH

-- Ruby level backtrace information ----------------------------------------
benchmark/bm_io_select3.rb:14:<main>
benchmark/bm_io_select3.rb:14:times'
benchmark/bm_io_select3.rb:15:in block in <main>'
benchmark/bm_io_select3.rb:15:in select'

-- C level backtrace information -------------------------------------------
./ruby-single-wait() [0x528508] vm_dump.c:797
./ruby-single-wait() [0x572226] error.c:249
./ruby-single-wait(rb_bug+0xb1) [0x573631] error.c:266
./ruby-single-wait() [0x4b8420] signal.c:624
lib64/libpthread.so.0() [0x33bf20eeb0]
lib64/libc.so.6() [0x33bea884bb]
./ruby-single-wait(rb_thread_fd_select+0x340) [0x5307e0] thread.c:2393
./ruby-single-wait() [0x4e2e00c] io.c:7298
./ruby-single-wait(rb_ensure+0xb) [0x41758b] eval.c:745
./ruby-single-wait() [0x43d4d72] io.c:7713
./ruby-single-wait() [0x524a2d] vm_insnhelper.c:403
./ruby-single-wait() [0x519224] insns.def:1012
./ruby-single-wait() [0x51eb8e] vm.c:1163
./ruby-single-wait() [0x51f8e5] vm.c:574
./ruby-single-wait() [0x4486b1] numeric.c:3225
./ruby-single-wait() [0x524a2d] vm_insnhelper.c:403
./ruby-single-wait() [0x519224] insns.def:1012
./ruby-single-wait() [0x51eb8e] vm.c:1163
./ruby-single-wait() [0x525e3f] vm.c:1404
./ruby-single-wait() [0x4147d2] eval.c:215
./ruby-single-wait() [0x4147d2] eval.c:262
./ruby-single-wait() [0x414379] main.c:38
/lib64/libc.so.6(__libc_start_main+0xfd) [0x33bea1ee5d]

-- Other runtime information -----------------------------------------------
  Loaded script: benchmark/bm_io_select3.rb
  Loaded features:
  0 enumerator.so
  1 /home/kosaki/ruby/lib/ruby/1.9.1/x86_64-linux/enc/encdb.so
  2 /home/kosaki/ruby/lib/ruby/1.9.1/x86_64-linux/enc/trans/transdb.so
  3 /home/kosaki/ruby/lib/ruby/1.9.1/rubygems/defaults.rb
  4 /home/kosaki/ruby/lib/ruby/1.9.1/isort.rb
  5 /home/kosaki/ruby/lib/ruby/1.9.1/rubygems/dependency_list.rb
  6 /home/kosaki/ruby/lib/ruby/1.9.1/x86_64-linux/rbconfig.rb
  7 /home/kosaki/ruby/lib/ruby/1.9.1/rubygems/exceptions.rb
  8 /home/kosaki/ruby/lib/ruby/1.9.1/rubygems/custom_require.rb
  9 /home/kosaki/ruby/lib/ruby/1.9.1/rubygems.rb

  Process memory map:

00400000-00625000 r-xp 00000000 00:00 0
d0:03 26269500
/home/kosaki/linux/ruby/ruby-single-wait
00824000-00827000 rw-p 00224000 00:00 0
d0:03 26269500
/home/kosaki/linux/ruby/ruby-single-wait
00827000-00830000 rw-p 00000000 00:00 0
014b6000-032b0000 rw-p 00000000 00:00 0
[heap]

03/16/2020
ruby 1.9.3dev (2011-05-03 trunk 31412) [x86_64-linux]
benchmark/bm_io_select3.rb:15: [BUG] Segmentation fault
max fd: 100000 (results not apparent with <= 1024 max fd)
Please do NOT send untested patch. :-/

Please do NOT send untested patch. :-/

76992403000-768824e3000 rw-p 00000000 00:00 0
76992509000-768925b0000 r-xp 00000000 fd:03 1725292 /home/kosaki/ruby/lib/ruby/1.9.1/x86_64-linux/enc/trans/transdb.so
7688925b000-768927a0000 --p 00020000 fd:03 1725292 /home/kosaki/ruby/lib/ruby/1.9.1/x86_64-linux/enc/trans/transdb.so
7688927b000-768927b0000 rw-p 00000000 fd:03 1725292 /home/kosaki/ruby/lib/ruby/1.9.1/x86_64-linux/enc/trans/transdb.so
7688927d000-7689290d0000 --p 00020000 fd:03 1725319 /home/kosaki/ruby/lib/ruby/1.9.1/x86_64-linux/enc/encdb.so
76889290000-7689290d0000 rw-p 00000000 fd:03 1725319 /home/kosaki/ruby/lib/ruby/1.9.1/x86_64-linux/enc/encdb.so
76889290e000-7689290e0000 --p 00000000 00:00 0
7688929e000-768928a12000 rw-p 00000000 00:00 0
768898a2000-768988a7000 rw-p 00000000 00:00 0
768898bc1000-768989b3000 rw-p 00000000 00:00 0
7fffffff8000-7fffffff80000000 00:00 0 [stack]
7fffffff552000 r-xp 00000000 00:00 0 [vsyscall]

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

zsh: abort (core dumped) ./ruby-single-wait benchmark/bm_io_select3.rb

#15 - 05/04/2011 12:23 AM - normalperson (Eric Wong)
Motohiro KOSAKI kosaki.motohiro@gmail.com wrote:

Issue #4531 has been updated by Motohiro KOSAKI.

Please do NOT send untested patch. :-/

% ./ruby-single-wait benchmark/bm_io_select3.rb
max fd: 100000 (results not apparent with <= 1024 max fd)
benchmark/bm_io_select3.rb:15: [BUG] Segmentation fault
ruby 1.9.3dev (2011-05-03 trunk 31412) [x86_64-linux]
I can't reproduce on my end, same architecture and base revision, too...
I just rebased cleanly off r31415 and still can't reproduce.

```
-- C level backtrace information -------------------------------------------
./ruby-single-wait() [0x528508] vm_dump.c:797
./ruby-single-wait() [0x572226] error.c:249
./ruby-single-wait(ruby_bug+0xb1) [0x573631] error.c:266
./ruby-single-wait() [0x4b8420] signal.c:624
/lib64/libpthread.so.0() [0x33bf20eeb0]
/lib64/libc.so.6() [0x33bea884bb]
./ruby-single-wait(ruby_thread_fd_select+0x340) [0x5307e0] thread.c:2393
```

My patches don't touch the rb_thread_fd_select() code. thread.c:2393
is in rb_fd_copy(), not rb_thread_fd_select(), and is the following:

```c
memcpy(dst->fdset, src, size);
```

Shouldn't that be:

```c
memcpy(dst->fdset, src->fdset, size);
```

since r31395?

#16 - 05/04/2011 08:23 AM - kosaki (Motohiro KOSAKI)

My patches don't touch the rb_thread_fd_select() code. thread.c:2393
is in rb_fd_copy(), not rb_thread_fd_select(), and is the following:

```c
memcpy(dst->fdset, src, size);
```

Shouldn't that be:

```c
memcpy(dst->fdset, src->fdset, size);
```

since r31395?

Right. Thanks for good catching.
I'll test your patch again.

#17 - 05/04/2011 08:30 AM - kosaki (Motohiro KOSAKI)

Your patch improve single fd case. but slightly decrease multiple fd case.
The most important usecase of IO.select is web server application (ie multiple fd case).
It's no good trade-off.

```
name    before  ruby 1.9.3dev (2011-05-03 trunk 31415) [x86_64-linux]
io_select       2.282   1.596
io_select2      56.223  3.097
io_select3      13.470  13.628
```

#18 - 05/04/2011 09:29 AM - kosaki (Motohiro KOSAKI)

```c
int
rb_wait_for_single_fd(int fd, int events, struct timeval *tv)
{
    (snip)
    BLOCKING_REGION({
        result = poll(&fds, 1, timeout);
        if (result < 0) lerrno = errno;
    }, ubf_select, GET_THREAD());

    if (result > 0) {
        /* remain compatible with select(2)-based implementation */
        result = (int)(fds.events & fds.events);
        return result == 0 ? events : result;
    }
```

03/16/2020
It's really unclear comment. select_internal() doesn't have such quirk. Can you please clarify this?

```
static VALUE
select_internal(VALUE read, VALUE write, VALUE except, struct timeval tp, rb_fdset_t *fds)
|
| (snip)
| n = rb_thread_fd_select(max, rp, wp, ep, tp);
| if (n < 0) {
| rb_sys_fail(0);
| }
| if (!pending && n == 0) return Qnil; /* returns nil on timeout */
```

#19 - 05/04/2011 10:16 AM - kosaki (Motohiro KOSAKI)
I've committed following five patches.

0001-rb_wait_for_single_fd-initial-implementation.patch
0002-io-wait-switch-to-rb_wait_for_single_fd.patch
0003-ext-socket-init.c-simplify-wait_connectable-using-rb.patch
0004-readline-use-rb_wait_for_single_fd-instead-of-rb_thr.patch
0006-rb_wait_for_single_fd-use-poll-on-Linux.patch

But I haven't committed below one patch.

0005-io.c-IO.select-use-rb_wait_for_single_fd-for-single-.patch

I'm waiting performance fix or explanation why should we commit it.

#20 - 05/04/2011 10:23 AM - kosaki (Motohiro KOSAKI)
+int
+rb_wait_for_single_fd(int fd, int events, struct timeval *tv)
+{
| struct pollfd fds;
| int result, lerrno;
| double start;
| int timeout = tv ? tv->tv_sec * 1000 + (tv->tv_usec + 500) / 1000 : -1;

no overflow check?

#21 - 05/04/2011 11:23 AM - normalperson (Eric Wong)
Motohiro KOSAKI kosaki.motohiro@gmail.com wrote:

I've committed following five patches.

0001-rb_wait_for_single_fd-initial-implementation.patch
0002-io-wait-switch-to-rb_wait_for_single_fd.patch
0003-ext-socket-init.c-simplify-wait_connectable-using-rb.patch
0004-readline-use-rb_wait_for_single_fd-instead-of-rb_thr.patch
0006-rb_wait_for_single_fd-use-poll-on-Linux.patch

Thanks, I'm preparing some comments + tests in response to your comments on 0006.

But I haven't committed below one patch.

0005-io.c-IO.select-use-rb_wait_for_single_fd-for-single-.patch

I'm waiting performance fix or explanation why should we commit it.

I'm OK with rejecting 0005. It's too ugly in retrospect...

Instead the io/wait extension should be expanded to be able to check for writability, too. Maybe by adding a new method:

```
IO#wait_writable(timeout = 0)
```

--
Eric Wong
3 more safety patches + tests based on your comments in this thread:

1) rb_wait_for_single_fd: explain return value for poll() users
2) rb_wait_for_single_fd: have poll()-using impl set EBADF
3) rb_wait_for_single_fd: check for invalid timeval

Also available at:

git pull git://bogomips.org/ruby rb_wait_for_single_fd-followup

Strongly disagree. Any language change should be passed matz review.

I don't understand this. Why does this behavior help to compatible?
When do we use it?

2) rb_wait_for_single_fd: have poll()-using impl set EBADF is good fix. but a testcase of it introduce ruby level new method. It's unacceptable. I'll commit only code fix.
Strongly disagree. Any language change should be passed matz review.

Huh? ext/* is only loaded during tests and never installed. No users see anything in ext/*.

1) use ppoll(2) if available. and use INT_MAX if unavailable. or
2) fallback select(2)

1) is safe because linux has ppoll(2).

OK, good point about ppoll(), I forgot that exists. I'll work on that later or tomorrow.

if (result > 0) {
    /* remain compatible with select(2)-based implementation */
    /*
    * Remain compatible with the select(2)-based implementation:
    * 1) mask out poll()-only revents such as POLLHUP/POLLERR
    * 2) In case revents only consists of masked-out events, return all
    * requested events in the result and force the caller to make an
    * extra syscall (e.g. read/write/send/recv) to get the error.
    */
    result = (int)(fds.revents & fds.events);
    return result == 0 ? events : result;
}

I don't understand this. Why does this behavior help to compatible? When do we use it?

We need to ensure rb_wait_for_single_fd(fd, events, timeval) returns only a subset of its +events+ argument because that's all select() is capable of.

If poll() returns POLLHUP/POLLERR, we should not expose those flags to callers of rb_wait_for_single_fd() since it would then behave differently if poll() or select() were used.

int events = RB_WAITFD_IN | RB_WAITFD_OUT;
int revents = rb_wait_for_single_fd(fd, events, NULL);
/* poll() itself may return POLLERR, but we prevent it from being in * revents since select() can't return that */
if (revents & RB_WAITFD_IN) {
    /* since we don't know POLLERR, we fall back to fail here */
    if (read(fd, ...) < 0) {
        rb_sys_fail(0);
    }
}
if (revents & RB_WAITFD_OUT) {
    /* since we don't know POLLERR, we fall back to fail here */
    if (write(fd, ...) < 0) {
        rb_sys_fail(0);
    }
    /* user code shouldn't care about anything else since it only * requested RB_WAITFD_IN|RB_WAITFD_OUT */

--
Eric Wong

#25 - 05/04/2011 05:23 PM - kosaki (Motohiro KOSAKI)

Hi
KOSAKI Motohiro kosaki.motohiro@gmail.com wrote:

diff --git a/ext/-test-/wait_for_single_fd/wait_for_single_fd.c b/ext/-test-/wait_for_single_fd/wait_for_single_fd.c
index d406724..6efd1af 100644
--- a/ext/-test-/wait_for_single_fd/wait_for_single_fd.c
+++ b/ext/-test-/wait_for_single_fd/wait_for_single_fd.c
@@ -25,6 +25,7 @@ Init_wait_for_single_fd(void)
 rb_define_const(rb_cObject, "RB_WAITFD_IN", INT2NUM(RB_WAITFD_IN));
 rb_define_const(rb_cObject, "RB_WAITFD_OUT", INT2NUM(RB_WAITFD_OUT));
 rb_define_const(rb_cObject, "RB_WAITFD_PRI", INT2NUM(RB_WAITFD_PRI));

- rb_define_const(rb_cObject, "INT_MAX", INT2NUM(INT_MAX));
+ Â Â rb_define_singleton_method(rb_cIO,
   "wait_for_single_fd", Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Arabic

  Strongly disagree. Any language change should be passed matz review.

Huh? Â ext/-test/* is only loaded during tests and never installed.
No users see anything in ext/-test/*

Oops, my bad. I misunderstood your diff. ok, I'll commit it.

  1) use ppoll(2) if available. and use INT_MAX if unavailable. or
  2) fallback select(2)

  1) is safe because linux has ppoll(2).

OK, good point about ppoll(), I forgot that exists. Â I'll work on that
later or tomorrow.

ok.

Â Â if (result > 0) {
- Â Â /* remain compatible with select(2)-based implementation */
- Â Â /*
- Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Arabic

Instead the io/wait extension should be expanded to be able to check for
writability, too. Maybe by adding a new method:

IO#wait_writable(timeout = 0)
Should've been:

```
IO#wait_writable(timeout = nil)
```

actually. I created a separate issue/patch for it:

```
http://redmine.ruby-lang.org/issues/4647
```

--

Eric Wong

#29 - 05/07/2011 12:02 AM - kosaki (Motohiro KOSAKI)
- Status changed from Open to Closed

#30 - 05/07/2011 12:24 AM - kosaki (Motohiro KOSAKI)

Eric Wong normalperson@yhbt.net wrote:

> Instead the io/wait extension should be expanded to be able to check for writability, too. Maybe by adding a new method:

```
   Â   Â   IO#wait_writable(timeout
```

#31 - 05/07/2011 12:24 AM - kosaki (Motohiro KOSAKI)

Issue #4531 has been updated by Eric Wong.

File 0001-rb_wait_for_single_fd-use-ppoll-instead-of-poll.patch added

I've uploaded a new patch to use ppoll() instead of poll()

Also pullable from git://bogomips.org/ruby rb_wait_for_single_fd-ppoll

I've committed modified patch based on it at r31450.

Files

<table>
<thead>
<tr>
<th>File</th>
<th>Size</th>
<th>Date</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>poll.mbox</td>
<td>19.9 KB</td>
<td>03/28/2011</td>
<td>normalperson (Eric Wong)</td>
</tr>
<tr>
<td>io_select_using_poll_test.rb</td>
<td>366 Bytes</td>
<td>03/29/2011</td>
<td>normalperson (Eric Wong)</td>
</tr>
<tr>
<td>rb_wait_for_single_fd.mbox</td>
<td>18.1 KB</td>
<td>05/03/2011</td>
<td>normalperson (Eric Wong)</td>
</tr>
<tr>
<td>rb_wait_for_single_fd-followup.mbox</td>
<td>9.25 KB</td>
<td>05/04/2011</td>
<td>normalperson (Eric Wong)</td>
</tr>
<tr>
<td>0001-rb_wait_for_single_fd-use-ppoll-instead-of-poll.patch</td>
<td>4.2 KB</td>
<td>05/05/2011</td>
<td>normalperson (Eric Wong)</td>
</tr>
<tr>
<td>0001-add-io-wait-benchmarks.patch</td>
<td>1.27 KB</td>
<td>05/05/2011</td>
<td>normalperson (Eric Wong)</td>
</tr>
</tbody>
</table>