Ruby - Bug #19387

ObjectSpace.each_objects only returns shareable objects after starting a Ractor

01/27/2023 07:27 PM - luke-gru (Luke Gruber)

Status:	Assigned		
Priority:	Normal		
Assignee:	ractor		
Target version:			
ruby -v:	3.3.0	**Backport:**	2.7: UNKNOWN, 3.0: UNKNOWN, 3.1: UNKNOWN, 3.2: UNKNOWN

Description

```
r = Ractor.new do
  receive # block, the problem is not the termination of the ractor but the starting
end

ObjectSpace.each_object(IO) { |io|
  p io # we get no objects
}
```

Related issues:

Related to Ruby - Feature #17270: ObjectSpace.each_object should be restricte...	**Closed**
Has duplicate Ruby - Bug #21149: Strange behavior of ObjectSpace.each_object ...	**Rejected**
Has duplicate Ruby - Bug #21401: ObjectSpace can't count Fibers after using a...	**Rejected**

Associated revisions

Revision c08fdc68383ee368c18e15e298502e6ee0089e18 - 09/05/2023 11:19 AM - KJ Tsanaktsidis

Allow waitpid(-1) to be woken if a waitpid(pid) call is pending

If two threads are running, with one calling waitpid(-1), and another calling waitpid($some_pid), and then $some_other_pid exits, we would expect the waitpid(-1) call to retrieve that exit status; however, it cannot actually do so until $some_pid *also* exits.

This patch fixes the issue by unconditionally checking for pending process group waits on SIGCHLD, and then allowing pending pid-only waits to "steal" the notification.

[Fixes #19387]

Revision c08fdc68383ee368c18e15e298502e6ee0089e18 - 09/05/2023 11:19 AM - KJ Tsanaktsidis

Allow waitpid(-1) to be woken if a waitpid(pid) call is pending

If two threads are running, with one calling waitpid(-1), and another calling waitpid($some_pid), and then $some_other_pid exits, we would expect the waitpid(-1) call to retrieve that exit status; however, it cannot actually do so until $some_pid *also* exits.

This patch fixes the issue by unconditionally checking for pending process group waits on SIGCHLD, and then allowing pending pid-only waits to "steal" the notification.

[Fixes #19387]

Revision c08fdc68 - 09/05/2023 11:19 AM - KJ Tsanaktsidis

Allow waitpid(-1) to be woken if a waitpid(pid) call is pending

If two threads are running, with one calling waitpid(-1), and another calling waitpid($some_pid), and then $some_other_pid exits, we would expect the waitpid(-1) call to retrieve that exit status; however, it cannot actually do so until $some_pid *also* exits.

This patch fixes the issue by unconditionally checking for pending process group waits on SIGCHLD, and then allowing pending pid-only waits

to "steal" the notification.

[Fixes #19387]

Revision 0b7a4fbaa9c56d2c67d00d86c69f9e5c71803267 - 09/09/2023 09:51 AM - KJ Tsanaktsidis

Allow waitpid(-1) to be woken if a waitpid(pid) call is pending

If two threads are running, with one calling waitpid(-1), and another
calling waitpid($some_pid), and then $some_other_pid exits, we would
expect the waitpid(-1) call to retrieve that exit status; however, it
cannot actually do so until $some_pid *also* exits.

This patch fixes the issue by unconditionally checking for pending
process group waits on SIGCHLD, and then allowing pending pid-only waits
to "steal" the notification.

[Fixes #19387]

Revision 0b7a4fbaa9c56d2c67d00d86c69f9e5c71803267 - 09/09/2023 09:51 AM - KJ Tsanaktsidis

Allow waitpid(-1) to be woken if a waitpid(pid) call is pending

If two threads are running, with one calling waitpid(-1), and another
calling waitpid($some_pid), and then $some_other_pid exits, we would
expect the waitpid(-1) call to retrieve that exit status; however, it
cannot actually do so until $some_pid *also* exits.

This patch fixes the issue by unconditionally checking for pending
process group waits on SIGCHLD, and then allowing pending pid-only waits
to "steal" the notification.

[Fixes #19387]

Revision 0b7a4fba - 09/09/2023 09:51 AM - KJ Tsanaktsidis

Allow waitpid(-1) to be woken if a waitpid(pid) call is pending

If two threads are running, with one calling waitpid(-1), and another
calling waitpid($some_pid), and then $some_other_pid exits, we would
expect the waitpid(-1) call to retrieve that exit status; however, it
cannot actually do so until $some_pid *also* exits.

This patch fixes the issue by unconditionally checking for pending
process group waits on SIGCHLD, and then allowing pending pid-only waits
to "steal" the notification.

[Fixes #19387]

Revision 76a8c963c7ad975b7bbfc1c4979bf7a2de15af27 - 01/11/2024 02:12 AM - KJ Tsanaktsidis

Add a test for what happens with concurent calls to waitpid

Ruby 3.1 and 3.2 have a bug in their *implementation*, for which I'm
backporting a fix. However, the current development branch doesn't have
the issue (because the MJIT -> RJIT change refactored how waitpid worked
substantially). I do however want to commit the test which verifies
that waitpid works properly on master.

[Fixes #19387]

Revision 76a8c963c7ad975b7bbfc1c4979bf7a2de15af27 - 01/11/2024 02:12 AM - KJ Tsanaktsidis

Add a test for what happens with concurent calls to waitpid

Ruby 3.1 and 3.2 have a bug in their *implementation*, for which I'm
backporting a fix. However, the current development branch doesn't have
the issue (because the MJIT -> RJIT change refactored how waitpid worked
substantially). I do however want to commit the test which verifies
that waitpid works properly on master.

[Fixes #19387]

Revision 76a8c963 - 01/11/2024 02:12 AM - KJ Tsanaktsidis

Add a test for what happens with concurent calls to waitpid

Ruby 3.1 and 3.2 have a bug in their *implementation*, for which I'm
backporting a fix. However, the current development branch doesn't have
the issue (because the MJIT -> RJIT change refactored how waitpid worked
substantially). I do however want to commit the test which verifies
that waitpid works properly on master.

[Fixes #19387]

History

#1 - 01/27/2023 07:32 PM - luke-gru (Luke Gruber)

- Subject changed from Issue with ObjectSpace.each_objects not returning IO objects after starting a ractor to Issue with ObjectSpace.each_objects not returning objects after starting a ractor

The problem is actually for any objects, not just IO.

PR here: https://github.com/ruby/ruby/pull/7191

Edit: I took down the PR because I realized it doesn't properly fix the issue. The only way to get this working, and also to get ObjectSpace._id2ref working, is to keep the RACTOR_BELONGING_ID in all builds, and use that info when letting ractors access the objects.

I'll create another PR with that in mind. But I'm wondering why was this only in debug builds? Does it have to do with speed?

#2 - 02/07/2023 06:05 AM - hsbt (Hiroshi SHIBATA)

- Status changed from Open to Assigned

- Assignee set to ko1 (Koichi Sasada)

#3 - 09/05/2023 11:20 AM - Anonymous

- Status changed from Assigned to Closed

Applied in changeset git|c08fdc68383ee368c18e15e298502e6ee0089e18.

Allow waitpid(-1) to be woken if a waitpid(pid) call is pending

If two threads are running, with one calling waitpid(-1), and another
calling waitpid($some_pid), and then $some_other_pid exits, we would
expect the waitpid(-1) call to retrieve that exit status; however, it
cannot actually do so until $some_pid *also* exits.

This patch fixes the issue by unconditionally checking for pending
process group waits on SIGCHLD, and then allowing pending pid-only waits
to "steal" the notification.

[Fixes #19387]

#4 - 09/05/2023 02:12 PM - ko1 (Koichi Sasada)

- Status changed from Closed to Assigned

c08fdc68383ee368c18e15e298502e6ee0089e18 is not related to this ticket.

#5 - 09/09/2023 09:51 AM - Anonymous

- Status changed from Assigned to Closed

Applied in changeset git|0b7a4fbaa9c56d2c67d00d86c69f9e5c71803267.

Allow waitpid(-1) to be woken if a waitpid(pid) call is pending

If two threads are running, with one calling waitpid(-1), and another
calling waitpid($some_pid), and then $some_other_pid exits, we would
expect the waitpid(-1) call to retrieve that exit status; however, it
cannot actually do so until $some_pid *also* exits.

This patch fixes the issue by unconditionally checking for pending
process group waits on SIGCHLD, and then allowing pending pid-only waits

to "steal" the notification.

[Fixes #19387]

#6 - 01/04/2024 10:24 PM - luke-gru (Luke Gruber)

- ruby -v set to 3.3.0

I believe this was closed prematurely due to an unrelated git commit that wrongly tagged it. Please reopen :)

#7 - 01/04/2024 11:43 PM - jeremyevans0 (Jeremy Evans)

- Status changed from Closed to Open

#8 - 01/11/2024 02:12 AM - Anonymous

- Status changed from Open to Closed

Applied in changeset git|76a8c963c7ad975b7bbfc1c4979bf7a2de15af27.

Add a test for what happens with concurent calls to waitpid

Ruby 3.1 and 3.2 have a bug in their *implementation*, for which I'm
backporting a fix. However, the current development branch doesn't have
the issue (because the MJIT -> RJIT change refactored how waitpid worked
substantially). I do however want to commit the test which verifies
that waitpid works properly on master.

[Fixes #19387]

#9 - 01/11/2024 03:34 AM - luke-gru (Luke Gruber)

It happened again :(I notified the committer of the mistake. Please reopen, thanks!

#10 - 01/11/2024 04:38 AM - jeremyevans0 (Jeremy Evans)

- Status changed from Closed to Open

#11 - 04/03/2024 03:50 AM - hsbt (Hiroshi SHIBATA)

- Status changed from Open to Assigned

#12 - 02/19/2025 11:07 PM - wanabe (_ wanabe)

- Has duplicate Bug #21149: Strange behavior of ObjectSpace.each_object after Ractor.new added

#13 - 03/12/2025 08:12 PM - Eregon (Benoit Daloze)

One idea to solve this would be to implement ObjectSpace.each_object like TruffleRuby does it (at least when there are multiple Ractors):
basically it's just the transitive ObjectSpace.reachable_objects_from(roots).
I.e. it uses the same mechanism as GC traversal/marking to find every object reachable from some roots objects (like Object class, the stack frames, etc).

Context: the latest breakage due to this bug: https://github.com/ruby/prism/pull/3491
And I know of many before, it's a very common issue that Ractor.new can't be used in make test-all.

#14 - 03/25/2025 11:21 PM - tenderlovemaking (Aaron Patterson)

- Assignee changed from ko1 (Koichi Sasada) to ractor

#15 - 05/09/2025 04:50 AM - osyoyu (Daisuke Aritomo)

I also ran into this, so I opened an pull request document this behavior.
https://github.com/ruby/ruby/pull/13278

#16 - 05/09/2025 06:41 PM - Eregon (Benoit Daloze)

- Related to Feature #17270: ObjectSpace.each_object should be restricted on multi-Ractors added

#17 - 06/09/2025 04:32 PM - jhawthorn (John Hawthorn)

- Has duplicate Bug #21401: ObjectSpace can't count Fibers after using a Ractor added

#18 - 06/09/2025 04:35 PM - jhawthorn (John Hawthorn)

- Subject changed from Issue with ObjectSpace.each_objects not returning objects after starting a ractor to ObjectSpace.each_objects only returns shareable objects after starting a Ractor