io.c: do not close inherited FDs by default

While I fully agree Ruby should create FDs with close-on-exec by default (as it has since 2.0.0); I don't believe Ruby should be arbitrarily closing file descriptors it does not know about.

The following is an example (using mwrap[1]) where closing inherited FDs is harmful.

```
MWRAP=dump_fd:99 mwrap make -j4 exam 99>>mwrap.out
```

I only found one minor regression from this change in test/lib/test/unit.rb as IO.new does not set close-on-exec when using jobserver FDs from make. A possible change is to make IO.new set FD_CLOEXEC by default.

[1] https://80x24.org/mwrap/README.html

Associated revisions

Revision b53fadfd - 08/16/2018 11:56 PM - normal
process.c: defaults to close_others false

Arbitrarily closing file descriptors on exec breaks use cases where a Ruby process sets up a descriptor for non-Ruby children to use. For example, the "rake foo" target may spawn any number of subprocesses (Ruby or not) which depends on parsing the "FOO" environment variable for out_fd:99 and writing to foo.out

```
FOO=out_fd:99 rake foo 99>>foo.out
```

Unfortunately, this introduced one incompatibility in test/lib/test/unit.rb and it now requires explicitly setting IO#close_on_exec=true

[ruby-core:88007] [Misc #14907]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@64399 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 64399 - 08/16/2018 11:56 PM - normalperson (Eric Wong)
process.c: defaults to close_others false

Arbitrarily closing file descriptors on exec breaks use cases where a Ruby process sets up a descriptor for non-Ruby children to use. For example, the "rake foo" target may spawn any number of subprocesses (Ruby or not) which depends on parsing the "FOO" environment variable for out_fd:99 and writing to foo.out

```
FOO=out_fd:99 rake foo 99>>foo.out
```

Unfortunately, this introduced one incompatibility in test/lib/test/unit.rb and it now requires explicitly setting IO#close_on_exec=true

[ruby-core:88007] [Misc #14907]

Revision 64399 - 08/16/2018 11:56 PM - normal
Arbitrarily closing file descriptors on exec breaks use cases where a Ruby process sets up a descriptor for non-Ruby children to use. For example, the "rake foo" target may spawn any number of subprocesses (Ruby or not) which depends on parsing the "FOO" environment variable for out_fd:99 and writing to foo.out

```
FOO=out_fd:99 rake foo 99>>foo.out
```

Unfortunately, this introduced one incompatibility in test/lib/test/unit.rb and it now requires explicitly setting IO#close_on_exec=true

[ruby-core:88007] [Misc #14907]

**Revision 30ad3429 - 08/17/2018 07:47 AM - normal**

NEWS: clarify that we still use FD_CLOEXEC [ci skip]

[Misc #14907]

Revision 64408 - 08/17/2018 07:47 AM - normalperson (Eric Wong)

NEWS: clarify that we still use FD_CLOEXEC [ci skip]

[Misc #14907]

Revision 64408 - 08/17/2018 07:47 AM - normal

NEWS: clarify that we still use FD_CLOEXEC [ci skip]

[Misc #14907]

**History**

#1 - 07/11/2018 11:12 AM - normalperson (Eric Wong)

[https://bugs.ruby-lang.org/issues/14907](https://bugs.ruby-lang.org/issues/14907)

I only found one minor regression from this change in test/lib/test/unit.rb as IO.new does not set close-on-exec when using jobserver FDs from make. A possible change is to make IO.new set FD_CLOEXEC by default.

Maybe this is a good complimentary change:

```diff
diff --git a/ext/socket/basicsocket.c b/ext/socket/basicsocket.c
index 2641b4410b..0418c022cf 100644
--- a/ext/socket/basicsocket.c
+++ b/ext/socket/basicsocket.c
@@ -28,6 +28,7 @@
     sock = rsock_init_sock(rb_obj_alloc(klass), NUM2INT(fd));
+    rb_maygvl_fd_fix_cloexec(fptr->fd);
     return sock;
 }

diff --git a/io.c b/io.c
index b041acdef0..ddebd09e65 100644
--- a/io.c
+++ b/io.c
@@ -8085,6 +8085,7 @@
     sock = rsock_init_sock(rb_obj_alloc(klass), NUM2INT(fd));
+    rb_maygvl_fd_fix_cloexec(fd);
     return sock;
 }
```

[03/12/2022]
diff --git a/test/ruby/test_io.rb b/test/ruby/test_io.rb
index a6a8e78209..97ea857c3e 100644
--- a/test/ruby/test_io.rb
+++ b/test/ruby/test_io.rb
@@ -3128,6 +3128,15 @@
   end

     def test_for_fd_close_on_exec
     IO.pipe do |r, w|
+       w.close_on_exec = false
+       w2 = IO.new(w.fileno, autoclose: false)
+       assert_predicate w, :close_on_exec?
+       assert_predicate w2, :close_on_exec?
+     end
+     end if have_close_on_exec?
+
     def test_ioctl_linux
       # Alpha, mips, sparc and ppc have an another ioctl request number scheme.
       # So, hardcoded 0x80045200 may fail.
     diff --git a/test/socket/test_basicsocket.rb b/test/socket/test_basicsocket.rb
index d388b4f0dd..6ea8bcfdf3 100644
--- a/test/socket/test_basicsocket.rb
+++ b/test/socket/test_basicsocket.rb
@@ -150,6 +150,13 @@
   s2 = BasicSocket.for_fd(sock.fileno)
   s2.autoclose = false
   sock.close_on_exec = false
+   s2 = BasicSocket.for_fd(sock.fileno)
+   assert_instance_of BasicSocket, s
+   s.autoclose = false
+
   sock.close
 end

#2 - 07/11/2018 03:29 PM - nobu (Nobuyoshi Nakada)
+
+     r.close_on_exec = true
+     r.close_on_exec = true

The second is w?

#3 - 07/11/2018 07:52 PM - normalperson (Eric Wong)

nobu@ruby-lang.org wrote:

+     r.close_on_exec = true
+     r.close_on_exec = true

The second is w?

Yes :) But I prefer we drop this hunk and use make IO#new
set FD_CLOEXEC [ruby-core:87905]

#4 - 07/19/2018 09:15 AM - akr (Akira Tanaka)

I understand that FD inheritance dependent program doesn't work well with Ruby.

But I doubt that all external library create FDs with close-on-exec now.
spawn's close_others option would be useful workaround to avoid FD leak
with such (problematic) library.

So, I'm reluctant that removing close_others feature.
It means that we lose the workaround.

However, turn off close_others by default (for spawn and IO.popen) seems possible choice for me.

It is difficult to see how many the problematic libraries actually exists
with closing FDs by default.
#5 - 08/06/2018 07:13 PM - normalperson (Eric Wong)

- File 0001-process.c-close_others-defaults-to-false.patch added

Updated patch to default close_others to false

#6 - 08/09/2018 06:31 AM - akr (Akira Tanaka)

We discuss this issue at today's meeting.

We agree we can challenge to change the default. Then, we can see actual problem (FD leak) exist or not.

#7 - 08/11/2018 01:42 AM - normalperson (Eric Wong)

akr@fsij.org wrote:

   We agree we can challenge to change the default. Then, we can see actual problem (FD leak) exist or not.

OK, I'll commit 0001-process.c-close_others-defaults-to-false.patch early next week (won't have much Internet access soon)

#8 - 08/15/2018 07:03 PM - normalperson (Eric Wong)

I wrote:

   OK, I'll commit 0001-process.c-close_others-defaults-to-false.patch

Nevermind, the version I uploaded here also introduces compatibility problems around make jobserver FD inheritance with rubyspec. So setting FD_CLOEXEC on IO.for_fd still introduces incompatibilities.

So either I need to make changes to test/lib/test/unit.rb or I make changes to spec/default.mspec (for spec/ruby/optional/capi/spec_helper.rb)

Or we continue to suffer with the problem which caused me to proposed this patch in the first place.

Anyways, this version which ONLY defaults "close_others: false" and makes no change to IO.for_fd is likely the safest:

https://80x24.org/spew/20180815182638.8286-1-e@80x24.org/raw

But I still hate having to make the change to test/lib/test/unit.rb, because this has a chance of breaking other 3rd-party code:

```diff
diff --git a/test/lib/test/unit.rb b/test/lib/test/unit.rb
index 51c8960c52..7a09466d1c 100644
--- a/test/lib/test/unit.rb
+++ b/test/lib/test/unit.rb
@@ -145,6 +145,8 @@ def non_options(files, options)
 r.close if r
 nil
 else
+     r.close_on_exec = true
+     w.close_on_exec = true
 @jobserver = [r, w]
 options[:parallel] ||= 1
 end
```  

Damned if we do, damned if we don't...

#9 - 08/16/2018 12:20 PM - Eregon (Benoit Daloze)

normalperson (Eric Wong) FWIW, I'm not fond of the jobserver-related logic in spec/ruby/optional/capi/spec_helper.rb, it's too complex and seems to gain nothing (but does prevent a make warning IIUC).

If there is an easy way to ignore the warning due to having a grandparent make process, I'll gladly take it. There is no point to parallelize building the C-API specs, it's single C files.
#10 - 08/16/2018 11:32 PM - normalperson (Eric Wong) wrote:

normalperson (Eric Wong) FWIW, I'm not fond of the jobserver-related logic in spec/ruby/optional/capi/spec_helper.rb, it's too complex and seems to gain nothing (but does prevent a make warning IIRC).

It's fine, I guess; and it highlights the risk of this change.

The annoying thing about this problem is with my original goal to fix a regression introduced in 2.0; we introduce a new regression for code written after 2.0 (so I patch test/lib/test/unit.rb ...)

#11 - 08/16/2018 11:57 PM - normalperson (Eric Wong)

- Status changed from Open to Closed

Applied in changeset trunk|r64399.

process.c: defaults to close_others false

Arbitrarily closing file descriptors on exec breaks use cases where a Ruby process sets up a descriptor for non-Ruby children to use. For example, the "rake foo" target may spawn any number of subprocesses (Ruby or not) which depends on parsing the "FOO" environment variable for out_fd:99 and writing to foo.out

FOO=out_fd:99 rake foo 99>>foo.out

Unfortunately, this introduced one incompatibility in test/lib/test/unit.rb and it now requires explicitly setting IO#close_on_exec=true

[ruby-core:88007] [Misc #14907]

Files

<table>
<thead>
<tr>
<th>File Name</th>
<th>Size</th>
<th>Date</th>
<th>Author (Eric Wong)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001-io.c-done-inherited-FDs-by-default.patch</td>
<td>4.45 KB</td>
<td>07/11/2018</td>
<td>normalperson</td>
</tr>
<tr>
<td>0001-process.c-close_others-defaults-to-false.patch</td>
<td>4.84 KB</td>
<td>08/06/2018</td>
<td>normalperson</td>
</tr>
</tbody>
</table>