Ruby trunk - Bug #2774

add explicit constraints for WONTFIX IO bug

02/22/2010 12:15 AM - mame (Yusuke Endoh)

Status:	Rejected	
Priority:	Normal	
Assignee:		
Target version:		
ruby -v:	ruby 1.9.2dev (2010-03-24 trunk 27033) [i686-linux]	**Backport:**

Description

=begin
Hi, all

I propose writing two constraints into rdoc of IO:

- after running IO#dup, both an original IO and generated IO may cause "bezarre behavior" except IO#close
- after running IO#reopen, an original IO that passed to reopen may cause "bezarre behavior" except IO#close

In short, it means "IO#dup should not be used" and "IO that was once passed to IO#reopen should be just closed."
"bezarre behavior" does not mean undefined behavior (such as SEGV), but means wrong order of reading and writing, wrong value from IO#pos, etc. (see Appendix)

These behaviors are caused by bug of core's wrong buffering handing. But to fix these, we must change the structure `rb_io_t', public API in include/ruby/io.h, resulting in binary incompatibility.
We are currently thinking that these behaviors won't cause so serious issue to ought to be fixed with even binary incompatibility.
So I'm proposing writing the above constraints to assert WONTFIX in the immediate future.

Please say your rebuttal if the above constraints make trouble in "real world" example. If valid and convincing rebuttal is come, the issue will be fixed with breaking binary compatibility. If not, the above constraint statement will be added.

Answers for anticipated rebuttal:

- how to replace stdout and stderr temporarily to invoke subprocess

Use Kernel#spawn's option.

- my existing code will not work

Please fix your code.
Fixing this, *all users* will be forced to reinstall ext library (even worse, the code of the library may need to be modified).
We are expecting rebuttals such as examples that cannot be absorbed by Ruby-level fix or that requires all users to do more cumbersome things.

If anyone says no objection in three days, I'll add the constraints.

Thanks,

Appendix: current bezarre behaviors of IO#dup and reopen

foo.txt

A
B
C

example: first-come-first-served gets after IO#reopen (1)

```
f1 = File.new("foo.txt")
f2 = File.new("foo.txt")
f1.reopen(f2)
p f1.gets #=> "A\n"
p f2.gets #=> nil (cannot read)
```

example: first-come-first-served gets after IO#reopen (2)

```
f1 = File.new("foo.txt")
f2 = File.new("foo.txt")
f1.reopen(f2)
p f2.gets #=> "A\n"
p f1.gets #=> nil (cannot read)
```

example: negative value of IO#pos after IO#reopen

```
f1 = File.new("foo.txt")
f2 = File.new("foo.txt")
f2.gets
f1.reopen(f2)
f2.gets
f1.rewind
p f2.pos #=> -2
```

example: wrong value of IO#pos after IO#dup

```
f1 = File.new("foo.txt")
f2 = f1.dup()
p f1.pos  #=> 0
p f2.gets #=> "A\n"
p f1.pos  #=> 6 (neither 0 or 2)
```

example: IO#pos with side-effect after IO#reopen

```
f1 = File.new("foo.txt")
f2 = File.new("foo.txt")
f2.gets
f1.reopen(f2)
f2.gets
p f1.pos  #=> 6
f2.pos
p f1.pos  #=> 4 (changed)
```

example: wrong order of reading after IO#dup

```
r, w = IO.pipe
Thread.new do
w.print "Foo\nBar"
sleep 1
w.print "Baz\n"
sleep 1
w.print "Qux\n"
```

```
end
p r.gets   #=> "Foo\n"
r2 = r.dup
p r2.gets  #=> "Baz\n" (not "BarBaz\n")
p r.gets   #=> "BarQux\n" (not "Qux\n")

example: wrong order of writing after IO#dup

---

f = File.new("out.txt", "w")
f1 = File.new("foo.txt")
f2 = File.new("foo.txt")
f1.reopen(f)
f2.reopen(f)
f2.puts("foo")
f1.puts("bar")
#=> resulting "bar\nfoo\n" in out.txt, not "foo\nbar\n"

--
Yusuke ENDOH mame@tsg.ne.jp
=end
```

Related issues:

Has duplicate Ruby trunk - Bug #2775: add explicit constraints for WONTFIX IO...	**Rejected**

History

#1 - 02/25/2010 12:08 AM - mame (Yusuke Endoh)

=begin
Hi,

2010/2/24 Tanaka Akira akr@fsij.org:

> > These behaviors are caused by bug of core's wrong buffering handing.
>
> I think it is difficult to fix.
>
> *snip*
>
> I think we should live with this behavior because
> buffering is important for performance.

OK, I agree.

Then, what should user care to avoid bizarre behavior? IOW, what behavior can be guaranteed? How about:

User must not read/write at the time multiple IOs that share the same file descriptor or duplicated file descriptor.

Anyway, I think we agreed these bizarre behaviors are WONTFIX.
IO#reopen's spec in rubyspec fails because of this. I'll soon quarantine! the spec.

--
Yusuke ENDOH mame@tsg.ne.jp

=end

#2 - 03/25/2010 02:03 AM - mame (Yusuke Endoh)

- Status changed from Open to Rejected

=begin
This ticket was rejected. These won't be fixed.
The current spec is explained in [ruby-core:28335]

--
Yusuke Endoh mame@tsg.ne.jp

=end