Ruby 1.8 - Bug #993

obsolete "nonblocking IO#read" used in system calls

01/09/2009 06:48 PM - JWuttke (Joachim Wuttke)

Status:	Closed
Priority:	Normal
Assignee:	shyouhei (Shyouhei Urabe)
Target version:	
ruby -v:	ruby 1.8.7

Description

=begin
Running ruby 1.8.7 under linux, from time to time I get the message
"warning: nonblocking IO#read is obsolete; use IO#readpartial or IO#sysread".

It comes from a line of code like
my_file_list += ls *.myextension

Hence I guess the invalid read is located somewhere inside the implementation of the .. system call.
=end

History

#1 - 01/09/2009 07:18 PM - shyouhei (Shyouhei Urabe)

=begin
Sorry but I've not been able to reproduce this.

Can you shou us a short ruby script that shows that error message?
=end

#2 - 01/09/2009 07:48 PM - JWuttke (Joachim Wuttke)

=begin
It is not that easy to reproduce the warning. I am issuing the ls .. command every ten seconds, and I get only a couple of warning messages per day, at irregular intervals. I should also explain that the directory I am running the ls against is mounted via NFS. Probably the warning only appears when a network congestion causes some delay in the execution of ls.

I would approach the problem in a very different way: Since IO#read is obsolete, I would search for obsolete lines in the Ruby source. Unfortunately, my knowledge of CRuby (the program) is basically zero so that I have no idea where to start searching.
=end

#3 - 01/09/2009 08:55 PM - JWuttke (Joachim Wuttke)

=begin
Well, I searched in the source. The "nonblocking" warning comes from io.c:io_fread. It is issued only if
a C library function (getc) raises EAGAIN or EWOULDBLOCK. This is consistent with my observation that
the warning appears only at irregular intervals - namely, if some anomaly occurs during execution of the
.. statement.

I think this is a very unsatisfactory state of affairs: If "IO#read" should not be used, (1) a warning should be given at "compile time", not at run time, and (2) it should not be used internally by the implementation of .. (which I am still not able to locate in the source code).

=end

#4 - 01/09/2009 10:32 PM - shyouhei (Shyouhei Urabe)

=begin
(1) "compile time" is not obvious for this language (as is a dynamic interpreter).

(2) On calling io_fread a backtick does *not* set its pipe a O_NONBLOCK flag, so
io_fread call should block until piped subprocess says anything (at least
according to Linux man page).

(3) A quick grance at the ls implementation of GNU coreutils does not find any
use of O_NONBLOCK.

So approaching from the source it is suspicious for a backtick to result in
EWOURLDBLOCK.

Of course there can be bugs and that can perhaps happen on occasion, that is another reason why we want to reproduce that on our environment.
=end

#5 - 02/02/2009 12:52 PM - ko1 (Koichi Sasada)

- Status changed from Open to Feedback

- Assignee set to shyouhei (Shyouhei Urabe)

- ruby -v set to ruby 1.8.7

=begin

=end

#6 - 12/21/2016 05:14 AM - shyouhei (Shyouhei Urabe)

- Status changed from Feedback to Closed