Ruby master - Bug #17813

rb_funcall() may reset the latest socket error unexpectedly since Ruby 3.0.0

04/19/2021 09:40 AM - kenhys (Kentaro Hayashi)

Status: Open
Priority: Normal
Assignee: 
Target version: ruby -v:
ruby 3.0.0p0 (2020-12-25 revision 95aff21468) [x64-mingw32] 
Backport: 2.6: UNKNOWN, 2.7: UNKNOWN, 3.0: UNKNOWN

Description

The problem

It seems that when rb_funcall() is called, the socket error is reset. This behavior is introduced since Ruby 3.0.0 on Windows.

With this incompatible change, it fails to get WSAGetLastError correctly even though the previous function call fails.

Does it intentional change?

The actual situation

This incompatible behavior is found with the following Fiddle example.

```ruby
module WinSock
  require 'fiddle/import'
  extend Fiddle::Importer
  dlload 'ws2_32.dll'
  extern 'int bind(int, void *, int)'
  extern 'int WSAGetLastError(void)'
end

p WinSock.bind(0, nil, 0)
p WinSock.WSAGetLastError
```

With Ruby 3.0 (ruby 3.0.0p0 (2020-12-25 revision 95aff21468) [x64-mingw32]), WinSock.bind(0, nil, 0) should be failed, thus it returns -1. (expected) WinSock.WSAGetLastError should return 10038 (WSAENOTSOCK) in this case, but returns 0. (unexpected)

With Ruby 2.7 (ruby 2.7.2p137 (2020-10-01 revision 5445e04352) [x64-mingw32]), WinSock.bind(0, nil, 0) returns -1. (expected) WinSock.WSAGetLastError return 10038 (WSAENOTSOCK). (expected)

Thus, the use case such as checking bind result and report socket related error via (WSAGetLastError) doesn't work as expected with Ruby 3.0 (Windows).

The expected status

If this change is introduced by intendedly, no need to change ruby itself. If not, it may be better to be fixed.

Additional Information

I've reported this issue to Fiddle, then since Fiddle 1.0.8, Fiddle.win32_last_socket_error is implemented as a workaround.

Unexpected error code with WSAGetLastError even though the previous function call was failed

https://github.com/ruby/fiddle/issues/72
Ruby cannot guarantee that the state of WSA does not change between two method calls, because select is called internally when thread switch occurs.
So, the solution of fiddle (adding Fiddle.win32_last_socket_error) is unavoidable in this case, I guess.

Aaron, the maintainer of Ruby's fiddle, can you backport fiddle from upstream?