Ruby master - Bug #13644

Windows - Setting Time.now

06/09/2017 03:00 AM - MSP-Greg (Greg L)

Status:	Open		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.5.0dev (2017-06-04 trunk 59013) [x64-mingw32]	**Backport:**	2.2: UNKNOWN, 2.3: UNKNOWN, 2.4: UNKNOWN

Description

While reviewing a MinGW build test-all failure in TestLogDevice#test_shifting_midnight_exist_file, I determined the cause. The test in question (and other tests that seem to have newer/better exception handling) sets Time.now.

The thread StackOverflow - Change system date programmatically seems to imply that in some instances/configurations, changing the system time on Windows systems requires elevated permissions.

I tested on both a mswin build (ruby 2.5.0dev (2017-05-27 trunk 58922) [x64-mswin64_140]) and a MinGW build (ruby 2.5.0dev (2017-06-08 trunk 59046) [x64-mingw32]), and both responded to:

```
Time.now = Time.mktime(2017, 1, 3, 1, 1, 1)
```

with the following eror:

```
undefined method `now=' for Time:Class (NoMethodError)
```

Both also had false = Time.respond_to?(:now=).

So, I'm confused as to how this test passes on mswin. Regardless, the test requires a skip to bypass it and allow MinGW test-all to pass.

I thought I'd file an issue before doing a PR. I'm building and testing on Win7. Finally, if anyone has time, where is the method now= defined?

History

#1 - 06/09/2017 10:32 PM - MSP-Greg (Greg L)

- ruby -v set to ruby 2.5.0dev (2017-06-04 trunk 59013) [x64-mingw32]

Submitted PR #1645

Re tests, this PR changes a file used in test-all, which passes on both appveyor and travis.

Spec tests failed on appveyor with a failure that I have intermittently seen on MinGW builds.

#2 - 06/10/2017 10:35 AM - nobu (Nobuyoshi Nakada)

It's defined in FakeTime module which is prepended to Time.
I haven't seen that failure on mingw.

#3 - 06/10/2017 02:09 PM - MSP-Greg (Greg L)

Nobuyoshi,

> It's defined in FakeTime module which is prepended to Time.

Thank you for being so polite and taking the time to respond. That was a very dumb mistake on my part. I reached that conclusion when I was fighting a poor, old, MinGW embedded ruby 2.2.4 build in a commercial app. It was nice when I swapped in my build of 2.3 stable, and everything worked...

Anyway, I believe I've found the real issue. It may be occurring because I'm using Win7, or possibly because I have UAC enabled.

On my system, File.utime only sets atime, not mtime. I suspect that the OS may not allow setting mtime. As you probably know, LogDevice.new

uses mtime to determine whether it's time to create a new log file.

Normally, I can't see how that would be an issue, but re this test trying to mock things, it is.

If you have a minute, could you check the following code? For a MinGW build, it changes LogDevice to use atime to determine whether to create a new log.

```
require 'logger'

module PLogDevice
  def initialize(log = nil, shift_age: nil, shift_size: nil, shift_period_suffix: nil)
    @dev = @filename = @shift_age = @shift_size = @shift_period_suffix = nil
    mon_initialize
    set_dev(log)
    if @filename
      @shift_age = shift_age || 7
      @shift_size = shift_size || 1048576
      @shift_period_suffix = shift_period_suffix || '%Y%m%d'

      unless @shift_age.is_a?(Integer)
        base_time = @dev.respond_to?(:stat) ?
          (RUBY_PLATFORM !~ /mingw/ ? @dev.stat.mtime : @dev.stat.atime) :
#          @dev.stat.mtime :
          Time.now
        @next_rotate_time = next_rotate_time(base_time, @shift_age)
      end
    end
  end
end
class Logger::LogDevice ; prepend PLogDevice ; end

module FakeTime ; attr_accessor :now ; end
class << Time   ; prepend FakeTime   ; end

log = "log"
File.open(log, "w") {}
File.utime(*[Time.mktime(2014, 1, 2, 0, 0, 0)]*2, log)

Time.now = Time.mktime(2014, 1, 2, 23, 59, 59, 999000)
dev = Logger::LogDevice.new(log, shift_age: 'daily')
dev.write("#{Time.now} hello-1\n")
dev.close

puts "\nTime.now  #{Time.now}\n" \
       "atime     #{File.atime(log)}\n" \
       "mtime     #{File.mtime(log)}\n"

Time.now = Time.mktime(2014, 1, 3, 1, 1, 1)
dev = Logger::LogDevice.new(log, shift_age: 'daily')
dev.write("#{Time.now} hello-2\n")
dev.close

puts "\nTime.now  #{Time.now}\n" \
       "atime     #{File.atime(log)}\n" \
       "mtime     #{File.mtime(log)}\n"
```

As listed, the logs are written correctly to the same folder the script is in. If you uncomment

```
#          @dev.stat.mtime :
```

and comment

```
          (RUBY_PLATFORM !~ /mingw/ ? @dev.stat.mtime : @dev.stat.atime) :
```

it behaves as normal (using mtime) and will fail (appending to the log file, not creating another one).

Hence, I don't believe there's an issue with logger, it's an issue with trying to mock this.

If adding the prepend to LogDevice looks okay, someone should do a PR or commit.

Again, thanks for all your work, both Ruby in general and windows issues.

#4 - 06/11/2017 03:43 AM - MSP-Greg (Greg L)

- File test-logger-test_logdevice.rb.patch added

nobu,

Attached is the patch I'm using in my MinGW build system for the test in question. It passes, and my test-all is now at 3 failures.

I ran into three issues with the current test mock.

1. Once I patched logger to use atime, the test wouldn't pass if the FakeTime.now DST setting was different that the current system DST setting. So, I moved the mock FakeTime.now to today.
2. Temp Directories using a block. I had issues with readline tests and temp files. With this, I could not get all the asserts to pass with a block. As to files, I believe Windows may have issues with - if process A does not close a file, process B can neither close nor delete the file.
3. As I understand Logger, there was code that would not be used in a normal Logger application. Removed/changed code that opened the log file first, code that closed and then opened (with .new) the log file, etc.

The current test uses a midnight delta of -1mS and +01:01:01. The plus delta seems high, and from the GH 539, the author (megayu) states

> In my real situation, if there is some events in 23:59:59, it will never shift the log file.

Seems to imply a minus delta of around a second. So, I set up the test with a symmetrical delta. I tested three deltas, 1000mS, 500mS, and 10mS, no failures with three processes running 50 tests each.

I can't test any other platform without pushing a PR. I learned a bit about the logger library.

Aside - found an interesting line in the file #729, just after the test in question.

```
env_tz_works = /linux|darwin|freebsd/ =~ RUBY_PLATFORM # borrow from test/ruby/test_time_tz.rb
```

Files

test-logger-test_logdevice.rb.patch	3.6 KB	06/11/2017	MSP-Greg (Greg L)