This file passes on 2.7, 3.0, and fails (if you remove the skip line) on 3.1:

```ruby
#!/usr/bin/env ruby -w

require "minitest/autorun"

class TestRegexpCreation < Minitest::Test
  R31 = RUBY_VERSION > "3.1"
  def test_literal_equivalence
    if R31 then
      assert_equal(/\x03/, /\cC/) # wrong! (note the assert)
    else
      refute_equal(/\x03/, /\cC/)
    end
  end

  def test_from_literal
    re = /\cC/
    assert_equal(/\cC/, re)

    if R31 then
      assert_equal "\\\x03", re.source # wrong?
    else
      assert_equal "\\\cC", re.source
    end
  end

  def test_from_source
    re = Regexp.new "\\\cC"
    assert_equal "\\\cC", re.source

    if R31 then
      skip
    else
      assert_equal(/\cC/, re) # can't be written to pass
      assert_equal(/\x03/, re) # can't be written to pass
    end
  end

  # on 3.1:
  #
  # if written as:
  #
  # assert_equal(/\x03/, re)
  #
  # it fails with:
```
1) Failure:
TestRegexpCreation#test_source [regexp31.rb:32]:
Expected: /\x03/
Actual: /\cC/

but if written as:
assert_equal(/\cC/, re)
it ALSO fails with:
1) Failure:
TestRegexpCreation#test_source [regexp31.rb:32]:
Expected: /\x03/
Actual: /\cC/

Related issues:
Related to Ruby master - Bug #14367: Wrong interpretation of backslash C in r...
Closed

History
#1 - 12/29/2021 05:01 AM - mame (Yusuke Endoh)
- Related to Bug #14367: Wrong interpretation of backslash C in regexp literals added

#2 - 12/29/2021 05:04 AM - zenspider (Ryan Davis)
It looks like tokadd_escape has drastically changed and dropped the \c, \M-, and \C- forms...

This isn't mentioned in the release notes, and seems a backwards incompatibility that should be reserved for 4.0:

#3 - 12/29/2021 05:06 AM - mame (Yusuke Endoh)
Looks like \c? in a regexp literal was changed for #14367.

p(/\cC/.source) #=> '\cC' in Ruby 3.0
p(/\cC/.source) #=> '\x03' in Ruby 3.1

jeremyevans0 (Jeremy Evans) What do you think?

#4 - 12/29/2021 05:24 AM - zenspider (Ryan Davis)
I was just coming back to point at:
Jeremy Evans: Fix handling of control/meta escapes in literal regexps [Wed May 12 12:37:55 2021 -0700 (8 months ago)]
found in https://github.com/ruby/ruby/commit/11ae581a4a7f5d5555c6378872eab8f25381b1b9

#5 - 12/29/2021 10:37 PM - janosch-x (Janosch Müller)
regexps with these escapes can still be constructed with the Regexp::new constructor, they are only pre-processed to hex escapes in Regexp literals.

/\cC/.source == Regexp.new('\cC').source # false iff Ruby >= 3.1

as the matched codepoints are the same, i'd say this only affects maintainers of parsers (i came across this in regexp_parser), and isn't much of a breaking change to end-users?

#6 - 12/30/2021 04:21 AM - jeremyevans0 (Jeremy Evans)
mame (Yusuke Endoh) wrote in #note-3:

Looks like \c? in a regexp literal was changed for #14367.

p(/\cC/.source) #=> '\cC' in Ruby 3.0
p(/\cC/.source) #=> '\x03' in Ruby 3.1

jeremyevans0 (Jeremy Evans) What do you think?

As janosch-x (Janosch Müller) mentioned, the matched codepoints are the same. The fact that #source returns a different result does not seem like a bug/regression to me.
#7 - 01/04/2022 08:20 PM - jeremyevans0 (Jeremy Evans)

- Status changed from Open to Rejected