Ruby master - Bug #12190

DateTime.strptime and Time.strptime does not have compatibility in terms of parsing timezone

03/17/2016 01:43 PM - sonots (Naotoshi Seo)

Status:	Rejected		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.3.0p0 (2015-12-25 revision 53290) [x86_64-darwin13]	**Backport:**	2.1: REQUIRED, 2.2: REQUIRED, 2.3: REQUIRED

Description

For CET, only DateTime.strptime successfully parsed timezone

```
irb(main):003:0> DateTime.strptime('2015-11-12 CET', '%Y-%m-%d %Z')
=> #<DateTime: 2015-11-12T00:00:00+01:00 ((2457338j,82800s,0n),+3600s,2299161j)>
irb(main):004:0> Time.strptime('2015-11-12 CET', '%Y-%m-%d %Z')
=> 2015-11-12 00:00:00 +0900
```

For JST also, only DateTime worked.

```
irb(main):005:0> ENV['TZ'] = 'UTC'
=> "UTC"
irb(main):006:0> DateTime.strptime('2015-11-12 JST', '%Y-%m-%d %Z')
=> #<DateTime: 2015-11-12T00:00:00+09:00 ((2457338j,54000s,0n),+32400s,2299161j)>
irb(main):007:0> Time.strptime('2015-11-12 JST', '%Y-%m-%d %Z')
=> 2015-11-12 00:00:00 +0000
```

For PST, both worked.

```
irb(main):004:0> DateTime.strptime('2015-11-12 PST', '%Y-%m-%d %Z')
=> #<DateTime: 2015-11-12T00:00:00-08:00 ((2457339j,28800s,0n),-28800s,2299161j)>
irb(main):003:0> Time.strptime('2015-11-12 PST', '%Y-%m-%d %Z')
=> 2015-11-12 00:00:00 -0800
```

I felt DateTime.strptime and Time.strptime should have compatibility.

Associated revisions

Revision 1ba62fa9 - 03/17/2016 05:46 PM - sonots (Naotoshi Seo)

- lib/time.rb (parse, strptime): Fix Time.parse/strptime does not have compatibility with DateTime.parse/strptime in terms of parsing timezone [Bug #12190] [Fix GH-1297]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@54167 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 54167 - 03/17/2016 05:46 PM - sonots (Naotoshi Seo)

- lib/time.rb (parse, strptime): Fix Time.parse/strptime does not have compatibility with DateTime.parse/strptime in terms of parsing timezone [Bug #12190] [Fix GH-1297]

Revision 54167 - 03/17/2016 05:46 PM - sonots (Naotoshi Seo)

- lib/time.rb (parse, strptime): Fix Time.parse/strptime does not have compatibility with DateTime.parse/strptime in terms of parsing timezone [Bug #12190] [Fix GH-1297]

Revision 54167 - 03/17/2016 05:46 PM - sonots (Naotoshi Seo)

- lib/time.rb (parse, strptime): Fix Time.parse/strptime does not have compatibility with DateTime.parse/strptime in terms of parsing timezone [Bug #12190] [Fix GH-1297]

Revision 54167 - 03/17/2016 05:46 PM - sonots (Naotoshi Seo)

- lib/time.rb (parse, strptime): Fix Time.parse/strptime does not have compatibility with DateTime.parse/strptime in terms of parsing timezone [Bug #12190] [Fix GH-1297]

History

#1 - 03/17/2016 01:46 PM - sonots (Naotoshi Seo)

- Subject changed from DateTime.strptime parses timezone, but Time.strptime does not for some timezone to DateTime.strptime and Time.strptime does not have compatibility in terms of parsing timezone

#2 - 03/17/2016 04:35 PM - sonots (Naotoshi Seo)

It seems both DateTime.strptime and Time.strptime uses Date._strptime internally https://github.com/ruby/ruby/blob/087a393fa56c4dcf247588d2fb018c4bd46003cb/lib/time.rb#L428, but Time.strptime is not utilizing :offset information given by it. That was why.

I also found that Time.parse also did not utilize the timezone information.

It looks Date class https://github.com/ruby/ruby/blob/2169bea82e57c3887285b06f36b0f79827de0986/ext/date/date_parse.c#L345-L421 supports more timezones than Time class https://github.com/ruby/ruby/blob/2169bea82e57c3887285b06f36b0f79827de0986/lib/time.rb#L97-L112 , so utilizing :offset given by Date._strptime resolves this issue.

#3 - 03/17/2016 04:38 PM - sonots (Naotoshi Seo)

PRed https://github.com/ruby/ruby/pull/1297

#4 - 03/17/2016 05:46 PM - sonots (Naotoshi Seo)

- Status changed from Open to Closed

Applied in changeset r54167.

- lib/time.rb (parse, strptime): Fix Time.parse/strptime does not have compatibility with DateTime.parse/strptime in terms of parsing timezone [Bug #12190] [Fix GH-1297]

#5 - 03/24/2016 07:51 AM - usa (Usaku NAKAMURA)

- Backport changed from 2.1: UNKNOWN, 2.2: UNKNOWN, 2.3: UNKNOWN to 2.1: REQUIRED, 2.2: REQUIRED, 2.3: REQUIRED

#6 - 04/18/2016 05:26 PM - nagachika (Tomoyuki Chikanaga)

I wonder if this change could break existing codes.
Any thoughts? If there's no real demand, I'll marks this as WONTFIX in 2.3 branch.

#7 - 04/18/2016 05:38 PM - usa (Usaku NAKAMURA)

I also thought in the same way.
So, I did not backport this into 2.1.

#8 - 04/19/2016 01:25 AM - sonots (Naotoshi Seo)

I verified that new codes cover all ZoneOffset which covered before with test codes
https://github.com/ruby/ruby/pull/1297/files#diff-0bc90ed869793fffc539dfe705fa8facR443
This change should not break existing codes.

(Of course, if someone expects that Time.parse('2000-01-01T00:00:00 CET') returns UTC timezone, such codes will be broken. But, I think such codes are already broken)

#9 - 04/19/2016 02:35 AM - akr (Akira Tanaka)

I think current Time.parse doesn't match the document.

```
# Since there are numerous conflicts among locally defined time zone
# abbreviations all over the world, this method is not intended to
# understand all of them.  For example, the abbreviation "CST" is
# used variously as:
#
#     -06:00 in America/Chicago,
#     -05:00 in America/Havana,
#     +08:00 in Asia/Harbin,
#     +09:30 in Australia/Darwin,
#     +10:30 in Australia/Adelaide,
#     etc.
```

```
#
# Based on this fact, this method only understands the time zone
# abbreviations described in RFC 822 and the system time zone, in the
# order named. (i.e. a definition in RFC 822 overrides the system
# time zone definition.)  The system time zone is taken from
# <tt>Time.local(year, 1, 1).zone</tt> and
# <tt>Time.local(year, 7, 1).zone</tt>.
# If the extracted time zone abbreviation does not match any of them,
# it is ignored and the given time is regarded as a local time.
```

"this method only understands the time zone abbreviations described in RFC 822 and the system time zone" is invalid now.

This causes that more system time zones are overriden.

For example, AST is used by America/Anguilla as -14400 (-04:00) and
Asia/Aden as 10800 (+03:00).

AST is not described in RFC 822.

But ext/date/date_parse.c defines AST as -4*3600 (-04:00).
This means date_parse.c ignores Asia/Aden.

So, people live in Asia/Aden will see following behavior difference
between Ruby 2.3 and Ruby 2.4.

```
% TZ=Asia/Aden ruby-2.3.0 -v -rtime -e 'p Time.parse("2000-01-01 00:00:00 AST")'
ruby 2.3.0p0 (2015-12-25 revision 53290) [x86_64-linux]
2000-01-01 00:00:00 +0300
% TZ=Asia/Aden ./ruby -v -rtime -e 'p Time.parse("2000-01-01 00:00:00 AST")'
ruby 2.4.0dev (2016-04-16 trunk 54606) [x86_64-linux]
2000-01-01 00:00:00 -0400
```

#10 - 04/19/2016 03:38 AM - sonots (Naotoshi Seo)

Thank you for pointing out behavior differences. I was supposed that this should not break backward compatibilities, but it was actually not.
I am going to revert then.

#11 - 04/19/2016 04:23 AM - sonots (Naotoshi Seo)

- Status changed from Closed to Rejected

Reverted with r54647, and changed status to Rejected.