Ruby master - Misc #10473

Date.to_datetime.to_time != Date.to_time

11/03/2014 05:01 PM - lojack (Jack Lowe)

Status:	Open	
Priority:	Normal	
Assignee:		

Description

Date.new(2014,1,1).to_datetime.to_time.utc.to_s
=> "2014-01-01 00:00:00 UTC"

Date.new(2014,1,1).to_time.utc.to_s
=> "2014-01-01 05:00:00 UTC"

History

#1 - 05/21/2016 04:45 AM - yui-knk (Kaneko Yuichiro)

Date#to_time interprets date as the local time zone.
But Date#to_datetime interprets date as UTC.

I think this incompatibility is a king of bug.
We have two ways to solve this incompatibility, to make both methods to interpret date as the local time zone,
or to interpret date UTC.

It is difficult to judge which is more natural or useful.
But I always interpret date as the local time zone in my daily life...

```
diff --git a/ext/date/date_core.c b/ext/date/date_core.c
index 3a10fcb..4cc3dc9 100644
--- a/ext/date/date_core.c
+++ b/ext/date/date_core.c
@@ -8520,35 +8520,7 @@ date_to_date(VALUE self)
 static VALUE
 date_to_datetime(VALUE self)
 {
-    get_d1a(self);
-
-    if (simple_dat_p(adat)) {
-       VALUE new = d_lite_s_alloc_simple(cDateTime);
-       {
-           get_d1b(new);
-           bdat->s = adat->s;
-           return new;
-       }
-    }
-    else {
-       VALUE new = d_lite_s_alloc_complex(cDateTime);
-       {
-           get_d1b(new);
-           bdat->c = adat->c;
-           bdat->c.df = 0;
-           RB_OBJ_WRITE(new, &bdat->c.sf, INT2FIX(0));
-#ifndef USE_PACK
-           bdat->c.hour = 0;
-           bdat->c.min = 0;
-           bdat->c.sec = 0;
-#else
-           bdat->c.pc = PACK5(EX_MON(adat->c.pc), EX_MDAY(adat->c.pc),
-                              0, 0, 0);
-           bdat->c.flags |= HAVE_DF | HAVE_TIME;
-#endif
-           return new;
-       }
-    }
+    return time_to_datetime(date_to_time(self));
 }

 /*
diff --git a/test/date/test_date_conv.rb b/test/date/test_date_conv.rb
```

```
index 3729476..0feaf2a 100644
--- a/test/date/test_date_conv.rb
+++ b/test/date/test_date_conv.rb
@@ -126,10 +126,13 @@ def test_to_datetime__from_time

   def test_to_datetime__from_date
     d = Date.new(2004, 9, 19) + 1.to_r/2
-    d2 = d.to_datetime
-    assert_equal([2004, 9, 19, 0, 0, 0, 0, 0],
-                 [d2.year, d2.mon, d2.mday, d2.hour, d2.min, d2.sec,
-                  d2.sec_fraction, d2.offset])
+
+    with_tz('Asia/Tokyo') do
+      d2 = d.to_datetime
+      assert_equal([2004, 9, 19, 0, 0, 0, 0, (3.to_r/8)],
+       [d2.year, d2.mon, d2.mday, d2.hour, d2.min, d2.sec,
+        d2.sec_fraction, d2.offset])
+    end
   end

   def test_to_datetime__from_datetime
```

#2 - 06/13/2016 09:12 AM - akr (Akira Tanaka)

The proposed patch seems fine.

However I recommend to add more tests for old dates around transition between Jurian to Geregorian Calendar.

#3 - 06/13/2016 12:39 PM - akr (Akira Tanaka)

Akira Tanaka wrote:

> The proposed patch seems fine.
>
> However I recommend to add more tests for old dates around transition between Jurian to Geregorian Calendar.

I found that there are days that exists on Jurian carendar but not on Gregorian calendar.

1000/2/29 is exist on Jurian calender but it is not exist on Gregorian calendar.

So, Date.new(1000, 2, 29) preveserves the arguments but Time.new(1000, 2, 29) doesn't.

```
% ruby -rdate -e '
p Date.new(1000, 2, 29)
p Time.new(1000, 2, 29)
'
#<Date: 1000-02-29 ((2086367j,0s,0n),+0s,2299161j)>
1000-03-01 00:00:00 +0918
```

So, Date.new(1000, 2, 29).to_time.to_datetime doesn't preserve the arguments
(and Date.new(1000, 2, 29).to_time.to_datetime.to_date doesn't round trip).

```
% ruby -rdate -e '
d = Date.new(1000, 2, 29)
p d
p d.to_time
p d.to_time.to_datetime
p d.to_time.to_datetime.to_date
'
#<Date: 1000-02-29 ((2086367j,0s,0n),+0s,2299161j)>
1000-03-01 00:00:00 +0918
#<DateTime: 1000-03-01T00:00:00+09:18 ((2086367j,52861s,0n),+33539s,2299161j)>
#<Date: 1000-03-01 ((2086368j,0s,0n),+0s,2299161j)>
```

#4 - 06/13/2016 02:09 PM - akr (Akira Tanaka)

Similar problem exists on Samoa (Pacific/Apia).

There is no 2011-12-30 in Pacific/Apia.
http://en.wikipedia.org/wiki/International_Date_Line

So, Date.new(2011,12,30) preserves the arguments but Time.new(2011,12,30) doesn't.

```
% TZ=Pacific/Apia ruby -rdate -e 'p Date.new(2011,12,30), Time.new(2011,12,30)'
```

```
#<Date: 2011-12-30 ((2455926j,0s,0n),+0s,2299161j)>
2011-12-31 00:00:00 +1400
```

Date doesn't depend on Time as much as possible.
In this sense, the current behavior, Date#to_datetime chooses UTC, is reasonable.

But if it is too confusing and Date#to_datetime should respect the local time zone using Time, it is better to use only utc_offset as follows instead of the cascading conversion date.to_time.to_datetime.

```
% TZ=Pacific/Apia ruby -rdate -e '
class Date
  def to_datetime2
    DateTime.new(year, mon, mday, 0, 0, 0, Time.new(year, mon, mday).utc_offset/86400r, start)
  end
end
d = Date.new(2011,12,30)
p d.to_datetime2'
#<DateTime: 2011-12-30T00:00:00+14:00 ((2455925j,36000s,0n),+50400s,2299161j)>
```