

## Ruby master - Bug #7445

### strftime('%s %z') doesn't work

11/27/2012 05:02 PM - felipec (Felipe Contreras)

<b>Status:</b> Rejected	
<b>Priority:</b> Normal	
<b>Assignee:</b> tadf (tadayoshi funaba)	
<b>Target version:</b> 3.0	
<b>ruby -v:</b> ruby 1.9.3p327	<b>Backport:</b>
<b>Description</b>	
See:	
<pre>ruby -e "require 'date'; p DateTime.strptime('0 +0100', '%s %z').strftime('%s %z')"</pre>	
The timezone is ignored.	
<b>Related issues:</b>	
Related to Ruby master - Feature #9794: DateTime.strptime() doesn't work corr...	<b>Closed</b> <b>05/02/2014</b>

### History

#### #1 - 11/27/2012 10:33 PM - charliesome (Charlie Somerville)

- File *bug-7445.patch* added

I've attached a patch that fixes this.

#### #2 - 11/27/2012 11:36 PM - usa (Usaku NAKAMURA)

- Category set to *lib*

- Status changed from *Open* to *Assigned*

- Assignee set to *tadf (tadayoshi funaba)*

- Target version set to *2.0.0*

#### #3 - 11/28/2012 08:01 PM - tadf (tadayoshi funaba)

- Priority changed from *Normal* to *3*

- Target version changed from *2.0.0* to *3.0*

this is intentional behaviour.

%s cannot represent localtime with offset.

%s implies utc.

this is just odd as a date.

but, it's an option.

we may be able to adopt this feature on next major.

i'm not sure at the present time.

#### #4 - 05/27/2013 12:59 PM - felipec (Felipe Contreras)

It works perfectly fine in C:

```
./test '%s' '%s %z'
1369626774
1369626774 -0500
```

```
#include <stdio.h>
#include <time.h>
```

```
int main(int argc, char *argv[]) {
    struct tm *tm;
    time_t t;
    char buf[0x100];
    t = time(NULL);
    tm = localtime(&t);
```

```

    for (int i = 0; i < argc; i++) {
        strftime(buf, sizeof(buf), argv[i], tm);
        printf("%s\n", buf);
    }
    return 0;
}

```

#### #5 - 05/27/2013 07:43 PM - tadf (tadayoshi funaba)

DateTime also does.

```

$ ruby -r date -e 'dt=DateTime.now; ARGV.each{|f| p dt.strftime(f)}' '%s' '%s %z'
"1369651261"
"1369651261 +0900"

```

so what?

#### #6 - 06/08/2013 06:32 AM - felipec (Felipe Contreras)

tadf (tadayoshi funaba) wrote:

DateTime also does.

```

$ ruby -r date -e 'dt=DateTime.now; ARGV.each{|f| p dt.strftime(f)}' '%s' '%s %z'
"1369651261"
"1369651261 +0900"

```

so what?

```

./test '0 +0100' '%s' '%s %z'
0
0 +0100

```

```

#define _XOPEN_SOURCE
#include <stdio.h>
#include <time.h>

```

```

int main(int argc, char *argv[]) {
    struct tm tm;
    char buf[0x100];
    for (int i = 2; i < argc; i++) {
        strptime(argv[1], argv[i], &tm);
        strftime(buf, sizeof(buf), argv[i], &tm);
        printf("%s\n", buf);
    }
    return 0;
}

```

#### #7 - 06/08/2013 07:42 AM - tadf (tadayoshi funaba)

but you can't preserve offset with mktime(3).

if you want substring which denote %z, you can call `_strptime`.

```

DateTime._strptime('0 +0100', '%s %z') #=> {:seconds=>0, :zone=>"+0100", :offset=>3600}

```

you told about C's stdlib.

why not talk about ruby's Time?

Time is based on stdlib.

why not want it with Time?

#### #8 - 06/29/2013 06:19 AM - tadf (tadayoshi funaba)

- Status changed from Assigned to Rejected

there is no evidence of bug.

#### #9 - 09/04/2013 11:21 AM - felipec (Felipe Contreras)

tadf (tadayoshi funaba) wrote:

but you can't preserve offset with mktime(3).

We are not talking about mktime(3), we are talking about strptime(). This is a red herring argument.

The documentation of `Date.strptime()` clearly mentions C's `strptime(3)` and `strftime`:

"See also `strptime(3)` and `strftime`."

And both of them work perfectly fine with `'%s %z'`. I showed in the code above how C's `strptime` works with `'%s %z'`, and here I'm showing Ruby's `strftime` does as well:

```
require 'date'
d = DateTime.new(1970,1,1,1,0,0,'+01:00')
d.strftime('%s %z')
=> "0 +0100"
```

So both `strptime(3)` and `DateTime.strftime()` accept `'%s %z'`, and Charlie Somerville attached a patch that can be applied on top of Ruby's trunk, and `DateTime.strptime()` would work just like `strptime(3)`, and would be consistent with `DateTime.strftime()`.

What reason could you possibly have to not accept this patch? What use-case would it break? How could it possibly affect negatively anybody?

If you truly think `'%s %z'` does not make sense, then throw an error when both `%s` and `%z` are used at the same time, both in `DateTime.strptime()`, `DateTime._strptime()`, and `DateTime.strftime()`. Also, update the documentation to mention that it's not exactly the same as `strptime(3)`, and neither `strptime(3)`. This of course would be totally stupid.

Just accept the patch and everything will be consistent.

if you want substring which denote `%z`, you can call `_strptime`.

No, I want `strptime` to work correctly.

you told about C's `stdlib`.  
why not talk about ruby's `Time`?  
`Time` is based on `stdlib`.  
why not want it with `Time`?

That English does not parse.

#### #10 - 09/04/2013 11:23 AM - felipec (Felipe Contreras)

tadf (tadayoshi funaba) wrote:

`DateTime` also does.

```
$ ruby -r date -e 'dt=DateTime.now; ARGV.each{|f| p dt.strftime(f)}' '%s' '%s %z'
"1369651261"
"1369651261 +0900"
```

so what?

So you accept `strftime()` works correctly with `'%s %z'`? Then why do you not accept that `strptime()` doesn't?

#### #11 - 09/04/2013 11:25 AM - felipec (Felipe Contreras)

tadf (tadayoshi funaba) wrote:

this is intentional behaviour.

Then update the documentation to say so.

`%s` cannot represent `localtime` with offset.

Yes it can. I've shown how it can in C `strptime(3)`, and even you showed that it works with `DateTime.strftime(3)`.

but, it's an option.  
we may be able to adopt this feature on next major.  
i'm not sure at the present time.

The patch is right there and it works perfectly, why aren't you applying it?

If you don't reopen this I'll escalate the issue through different channels.

#### #12 - 09/05/2013 07:38 AM - tadf (tadayoshi funaba)





That is written to the destination in the reference is not necessarily as it is Spoken. References  
It is not just a reference for a better understanding. Is natural. Is common sense.

Totally unparsable.

quoted from more(1)

SEE ALSO  
vi(1), less(1)

or equivalent to less? more I have exceptional.

First of all , you should you have said already , but the machine perfectly '%s z%' as you say  
Nothing to be is! Performance . The syntax level you 're talking about is , read "  
It will only say "he . It has been described as possible with \_strptime. Its  
For C, the members of the time zone is not a standard Somo also . Thailand from the original  
It is not symmetric about the time difference and time zones . What if , built-in classes of ruby  
Show How do you do in Time.

quoted from (3) localtime

```
struct tm {
    int tm_sec;        /* seconds */
    int tm_min;        /* minutes */
    int tm_hour;       /* hours */
    int tm_mday;       /* day of the month */
    int tm_mon;        /* month */
    int tm_year;       /* year */
    int tm_wday;       /* day of the week */
    int tm_yday;       /* day in the year */
    int tm_isdst;     /* daylight saving time */
};
```

I don't know what you are talking about, but we are not talking about Time, nor localtime, which I presume is actually localtime().

Time doesn't even have strptime(), so what's the point of bringing it up? I have no idea.

Well as DateTime, Time of built-in class , library of C also is the same  
But, in order to properly accept , as well as the syntax level and have to interpret the meaning  
Must . Those that do not rely on the first elements and conflicting be ignored  
Will be . You're of sesame and pretend you do not notice it at all .

%s is intended to time\_t, but there is no such a member to struct tm  
In fact , it should be stored in some cases local self-evident is decomposed so . Even if time difference  
I do not go , multiply Dzuna mind it's a little funny thing is , even if it is stored.  
The difference would be good if they match that of the local time , but it is not necessarily so  
Do . It would be implemented as soon as the first place and whether it is stored . mktime (3) is free it  
Coherence will fit if you view. Behavior of this area think that it is undefined in any  
There are . Is not the significance of ours to discuss .

```
#define _XOPEN_SOURCE
#include <stdio.h>
#include <time.h>

int main(int argc, char *argv[]) {
    struct tm tm;
    char buf[100];
    strptime(argv[1], argv[2], &tm);
    strftime(buf, 100, "%F %T %z", &tm);
    printf("%s\n", buf);
    return 0;
}
```

```
$ ./a.out '3 +0100' '%s %z'
1970-01-01 09:00:03 +0100
$ ./a.out '3 -0100' '%s %z'
1970-01-01 09:00:03 -0100
```

In this case, the difference was to be read indeed .

```
$ date -d '1970-01-01 09:00:03 +0100' +%s
```

```
28803
$ date -d '1970-01-01 09:00:03 -0100' +%s
36003
```

But , just I 'm just reading . Or something if symmetry 'm not a story .

Instance of Time and DateTime is an object, what mktime (3 ) basis  
You can not eliminate . I should have to as it is fraught elements inconsistent or DateTime  
It is not impossible in Time. Object is not a struct tm. You  
C still does not work , in fact that you are saying , and so any DateTime  
There is not even going to prepare Na . The fact that you said is nonsense at all .

Again, Time is irrelevant, we are talking about DateTime, so mktime() or whatever backend is used is irrelevant, the documentation mentions  
strptime(3), and strptime(3) accepts '%s %z', therefore DateTime.strptime() should accept it too, and it can if the attached patch is applied.

```
> Time.strptime('1 +1200', '%s %z')
#=> 1970-01-01 09:00:01 +0900
> Time.strptime('1 +2400', '%s %z')
#=> 1970-01-01 09:00:01 +0900
```

Again, there is no strptime in Time.

```
Time.strptime('1 +1200', '%s %z')
NoMethodError: undefined method `strptime' for Time:Class
  from (irb):1
  from /usr/bin/irb:12:in `'
```

In fact , there are other combinations do not make sense . On the other hand , one by one  
I do not think it is trying to provide a correct interpretation . And be modified as you say in this matter  
Lever , so although I have not yet obtained the perfect symmetry .

```
> Time.strptime('1 0', '%s %w').wday
#=> 4

> Time.strptime('1 PM', '%s %p').hour
#=> 9

> DateTime.new(2001,2,3,16,5,6).strftime('%F %T %p')
#=> "2001-02-03 16:05:06 PM"
> DateTime.strptime('%F %T %p', "2001-02-03 16:05:06 PM")
ArgumentError: invalid date
```

That's because you are passing the arguments the other way around:

```
DateTime.strptime("2001-02-03 16:05:06 PM", '%F %T %p')
=> #<DateTime: 2001-02-03T16:05:06+00:00 ((2451944j,57906s,0n),+0s,2299161j)>
```

Geezus.

I have never seen that the date format `%%s %%z` to me. Time difference and time zone  
What is needed , it will be because it is local time in nature. % s 's shown ,  
In time\_t, there is no information of the time difference and time zone to time\_t. Its on Unix  
As given , self-explanatory , as there is no need to explain , just fire Re  
Only some convex , this is the basic .

Again, time\_t is totally and completely irrelevant. strptime(3) and strftime(3) do NOT use it.

For example, let's look at the ISO 8601. There a difference in local time , but the Universal time  
Would not . I do not know why actively accept this . Bug at least  
Not a . I do not know reason to say as a matter of course , you'd been accepted .

I have thought and time zone difference as something like modifier but , '% s% z'  
In There is nothing of interest. something '%s %p%' is the same . Such things  
I do not think you must give a meaning to .

It doesn't matter what you think. strptime(3) does give a meaning to '%s %z'. Period.

I think it strange that as a date you '%s %z%' is attempting to represent . Its  
If interpret force the level , it is not a date , an incomplete copy of the date object  
I think that it is not only a kind of copy .

It is not incomplete, every date can be represented with '%s %z'.

That's why Git, the most successful DVCS by far, which is headed to replace Subversion as the most popular VCS in general uses precisely this:

```
% git cat-file -p trunk
tree 075a08c3e84f214ab9426957b3682b94c56b3046
parent bbf366bad8b51c491820b67338125d36b88c8f8e
author zzak <zzak@b2dd03c8-39d4-4d8f-98ff-823fe69b080e> 1380550541 +0000
committer zzak <zzak@b2dd03c8-39d4-4d8f-98ff-823fe69b080e> 1380550541 +0000
```

```
* ext/objspace/objspace.c: [DOC] Cleaned up many rdoc formatting
issues and several duplicate grammar bugs.
```

```
git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@43099 b2dd03c8-39d4-4d8f-98ff-823fe69b080e
```

Oh, look at that '1380550541 +0000', so '%s %z', that's all Git uses to represent dates, and it's not missing any information. Probably hundreds of thousands, maybe even millions of people don't have a problem with these dates.

Oh, and since I've worked on Mercurial and Bazaar, I can tell you that these two fields is all they need as well.

From it , because it may contain a leap second time\_t, portability is poor . So , This is even able to read and write in various places , is recommended the use of less active The fact that it will not also like I want to say . the built-in classes of ruby I do not feel the significance that I'll overdo it that you do not Spoken and even for . A new feature If " Use steadily . Bug was a bug because I grew out of it by this " Madashimo I do not want to say absolutely is .

Or future , it might be The possible as undefined behaviors . However , now I do not think at all of place , if you want to accept . It must be accepted right now to reverse Will there why not . Never complains about anyone until now , Oh Other than thou is .

You can see if you look at the code , but ignore consciously . Has been doing intended . Thought does not Hen-tsu also now, I feel to be persuaded you will not . Consent this Consult your great men of the other if you can not .

I don't understand what you mean, but it is a bug. The documentation says this.

"See also strptime(3) and strftime."

1) strptime(3) does work correctly with '%s %z'

2) strftime()

"This is not a bug " is the conclusion of me.

Well you are wrong, and your are completely unreasonable and uncooperative attitude is not helping, so I'm going to escalate to other channels.

**#14 - 10/06/2013 10:17 AM - felipec (Felipe Contreras)**

- File 0001-time-fix-strptime.patch added

Also, if '%s' is used for UTC, then why do we get this?

```
require 'time'
Time.strptime('0', '%s')
=> 1969-12-31 18:00:00 -0600
```

I'm attaching a patch that does exactly what we want:

```
Time.strptime('0', '%s')
=> 1970-01-01 00:00:00 +0000
Time.strptime('0 +0100', '%s %z')
=> 1970-01-01 01:00:00 +0100
```



#### #15 - 10/07/2013 10:28 AM - charliesome (Charlie Somerville)

+1 Patch looks good to me

#### #16 - 10/09/2013 07:46 AM - felipec (Felipe Contreras)

So let's recap the current situation.

Time.strptime() has been fixed in Ruby Core, and Rubinius has fixed.strptime() in both Time, and DateTime in rubysl.

So we have this:

### Ruby MRI

```
puts Time.strptime('1 +0100', '%s %z').strftime('%s %z')
=> 1 +0100
```

### Rubinius

```
puts Time.strptime('1 +0100', '%s %z').strftime('%s %z')
=> 1 +0100
```

```
puts DateTime.strptime('1 +0100', '%s %z').strftime('%s %z')
=> 1 +0100
```

### C

```
#define _XOPEN_SOURCE
#include <stdio.h>
#include <time.h>

int main(int argc, char *argv[])
{
    struct tm tm;
    char buf[0x100];
   .strptime("1 +0100", "%s %z", &tm);
    strftime(buf, sizeof(buf), "%s %z", &tm);
    printf("%s\n", buf);
    return 0;
}

=> 1 +0100
```

### Perl

```
use DateTime::Format::Strptime;

my $format = DateTime::Format::Strptime->new(
    pattern => '%s %z',
);

my $dt = $format->parse_datetime('1 +0100');
print $dt->strftime('%s %z'), "\n";

=> 1 +0100
```

And you still maintain that this is desirable?

```
puts DateTime.strptime('1 +0100', '%s %z').strftime('%s %z')
=> 1 +0000
```

Come on, just apply the patch.

#### #17 - 10/15/2013 02:57 PM - felipec (Felipe Contreras)

felipec (Felipe Contreras) wrote:

Time.strptime() has been fixed in Ruby Core, and Rubinius has fixed.strptime() in both Time, and DateTime in rubysl.

I guess the lack of reply means that you won't do the sensible thing here, even though it makes sense to pretty much everyone else, you just don't want to accept you were wrong.

#### #18 - 05/02/2014 10:57 AM - tadf (tadayoshi funaba)

- Related to Feature #9794: `DateTime.strptime()` doesn't work correctly for '%s %z' added

## Files

---

bug-7445.patch	1.34 KB	11/27/2012	charliesome (Charlie Somerville)
0001-time-fix-strptime.patch	1.44 KB	10/06/2013	felipec (Felipe Contreras)