Ruby trunk - Feature #2759

Regexp /g and /G options

02/19/2010 04:01 AM - manveru (Michael Fellinger)

Status:	Closed
Priority:	Normal
Assignee:	
Target version:	1.9.2

Description

=begin
Oniguruma has flags called ONIG_OPTION_CAPTURE_GROUP, ONIG_OPTION_DONT_CAPTURE_GROUP.
There has been a discussion about it in the past, which can be found at
http://blade.nagaokaut.ac.jp/cgi-bin/vframe.rb/ruby/ruby-dev/28446?28233-28923 (Japanese).
As far as I can tell, the proposed patch hasn't been applied yet due to lack of demand.
With this Issue, I'd like to vote for application of the patch, as I need the functionality in my code.
=end

History

#1 - 02/19/2010 05:28 AM - naruse (Yui NARUSE)

=begin
I can't imagine a use case even if we have non capturing group (?: ..).

Could you explain why you want /g and /G ?
=end

#2 - 02/19/2010 06:00 AM - murphy (Kornelius Kalnbach)

=begin
The /g modifier is widely used to mark a regexp as "global" (Perl, JavaScript, vim). It might confuse people. But it's also the obvious name for such an option –

How is the use case? That sounds interesting.
=end

#3 - 02/20/2010 03:08 AM - manveru (Michael Fellinger)

=begin
Given the following code in 1.8 with oniguruma gem, first without the flag:

Oniguruma::ORegexp.new('(?=([ ]{4}|\t)(?-\k))')
ArgumentError: Oniguruma Error: invalid backref number/name

Now with the flag:

Oniguruma::ORegexp.new('(?=([ ]{4}|\t)(?-\k))', :options => Oniguruma::OPTION_CAPTURE_GROUP)
/(?=([ ]{4}|\t)(?-\k))/

It fails like this in 1.9, clearly the flag is needed:

/(?=([ ]{4}|\t)(?-\k))/
SyntaxError: (irb):3: invalid backref number/name: /(?=([ ]{4}|\t)(?-\k))/

This is an excerpt from a regular expression found in the Markdown syntax used by Textmate.
Textmate uses the Oniguruma C API directly, and uses this flag for the regular expressions it uses for syntax highlighting.
There are a couple of other regular expressions that I cannot process due to the lack of this flag, and the oniguruma gem is not easy to port to Ruby 1.9.
I've considered writing a FFI wrapper for the C API, but that would just duplicate all of the work already in Ruby core just for the addition of this flag, that is why I propose adding it to Ruby instead.
=end

#4 - 02/20/2010 06:30 AM - murphy (Kornelius Kalnbach)

=begin
I don't understand why Ruby 1.9 raises that exception. The Oniguruma documentation (http://www.geocities.jp/kosako3/oniguruma/doc/RE.txt) states that the use of named group forbids the use of numbered *back references*, not un-named groups entirely:

- Back reference by group number is forbidden if named group is defined in the pattern and ONIG_OPTION_CAPTURE_GROUP is not setted.

So, you can fix it for Ruby 1.9 by giving all groups a name: /(?=(?<_>[ ]{4}|\t)(?-\k))/, but it shouldn't be necessary. Ruby's fault, or Oniguruma's?
=end

#5 - 02/20/2010 07:36 AM - naruse (Yui NARUSE)

=begin
If you don't use $1 after matching, you can use /(?=(?:[ ]{4}|\t)(?-\k))/.

Otherwise you use a string matched by ([ ]{4}|\t), you should give name to it, like Kornelius says,
/(?=(?[ ]{4}|\t)(?-\k))/
=end

#6 - 02/20/2010 08:41 AM - murphy (Kornelius Kalnbach)

=begin
Michael: If I'm not mistaken, you want to use the TextMate syntax files without change. The lack of a OPTION_CAPTURE_GROUP modifier currently forces you to transform Oniguruma regexps to Ruby 1.9 regexps before using them.

So, in essence you want more seamless support for Oniguruma.
=end

#7 - 02/20/2010 02:49 PM - manveru (Michael Fellinger)

=begin
Well, yes, of course I want seamless support for Oniguruma, wasn't that one of the big new features of the 1.9 series?
Some of these regular expressions fill a page, and there is no way I know of to translate them automatically and keep the capture groups identical.

Here is a longer example: http://pastie.org/833906
It took me almost a week of studying and modifying that Regexp to conclude that Ruby in fact is not providing all features of Oniguruma, at first I even tried adjusting them manually and automatically.
The thing is, I cannot transform this without changing the semantics, numbering of capture groups and the like, which are important for the handling of results.
Here is a little collection of Regular expressions I fix currently to mitigate warnings and syntax errors that can be avoided:
http://github.com/manveru/ver/blob/master/lib/ver/plist.rb#L167-240
I have no control over the Textmate bundles that may be thrown at this, and right now I'd have to spend days fixing regular expressions manually and praying they still work as intended, and repeating the whole process once the bundle has changed, given that I even notice it is incompatible with Rubys fork of Oniguruma in the first place, which I doubt many people will report.

So yes, please, add some way to use this flag, even if it's as a secret argument to Regexp.new, which seems to ignore flags now for some reason.
I don't even need it to be in syntax, as there is only one place where it is used (when converting from the original XML Plists to Ruby Regexp).
=end

#8 - 02/21/2010 06:47 PM - naruse (Yui NARUSE)

=begin
Ruby 1.9's regexp is from Oniguruma, but is not Oniguruma.
It is fork one and is not intended to support compatibility to Oniguruma.
=end

#9 - 02/21/2010 08:10 PM - manveru (Michael Fellinger)

=begin
So the solution is to create a whole new wrapper for Oniguruma?
=end

#10 - 02/21/2010 08:51 PM - naruse (Yui NARUSE)

=begin
If you want full compatibility to Oniguruma, yes.
Or original Oniguruma seems not maintained now, so you can use common functions of Ruby 1.9 and Oniguruma 5.9.2.

But I think, what you want is only resolve mixing captured group and named capture.
Until the problem is only this one, you can simply replace captured group to not captured group.

(?:subexp) not captured group
(subexp) captured group
http://www.geocities.jp/kosako3/oniguruma/doc/RE.txt

This solution doesn't work when the regexp uses numbered-backref/call, but your example seems not hit it.

Anyway you should report this to original author (Textmate people? manveru?).
Of course those regexp doesn't work in Ruby 1.9, so they also need to change there regexps.

FYI, document of Ruby's regexp is (thanks Roger!):
http://svn.ruby-lang.org/cgi-bin/viewvc.cgi/trunk/doc/re.rdoc?revision=24992&view=markup

=end

#11 - 02/22/2010 02:36 AM - manveru (Michael Fellinger)

=begin
The Textmate maintainer couldn't care less, they use the Oniguruma C API directly, and the editor is not written in Ruby, so it's unlikely that they would want to fix something that's not broken.
I'm the user manveru on github, so you just reported this to me, but as I said before, I cannot modify the regular expression without changing its semantics and match offsets.
I'm simply trying to combine the comfort of writing the editor in Ruby and using the existing functionality of Oniguruma to take advantage of hundreds of already existing syntax highlighting definitions that exist for Textmate.

Someone just informed me that, after the patch from Kosako was rejected, Kosako stopped his work on Ruby altogether.
His last comment was http://d.hatena.ne.jp/kkos/20070525#1180100250 and matz answered with http://www.rubyist.net/~matz/20070525.html

I still don't understand the reasons to fork Oniguruma, why would a fork provide less features than the original?

I'll get started on writing FFI::Oniguruma now, you may flag this issue as rejected.
=end

#12 - 02/22/2010 03:15 AM - murphy (Kornelius Kalnbach)

=begin
Actually, a converter doesn't seem too hard: http://pastie.textmate.org/835608
=end

#13 - 02/22/2010 03:27 AM - manveru (Michael Fellinger)

=begin
That's if you don't accept numeric backrefs.
Here is an example of a regular expression used in the PHP bundle, including the modified version and original: http://pastie.org/835626
=end

#14 - 02/22/2010 03:43 AM - murphy (Kornelius Kalnbach)

=begin
Right...my post was actually just a scaffold for a converter. This one handles backrefs: http://pastie.textmate.org/835652
=end

#15 - 02/22/2010 04:10 AM - manveru (Michael Fellinger)

=begin
Thank you very much, that seems handle most of what I need, have to make sure it works correctly tomorrow.
=end

#16 - 02/22/2010 04:23 AM - murphy (Kornelius Kalnbach)

=begin
If it does work, we could forge a Gem out of this, and continue the discussion outside of ruby-core ;)
=end

#17 - 02/23/2010 12:50 PM - mame (Yusuke Endoh)

=begin
Hi,

2010/2/22 Michael Fellinger redmine@ruby-lang.org:

> I still don't understand the reasons to fork Oniguruma, why would a fork provide less features than the original?

I think that the "fork" is not a right word in this case.

Oniguruma has not only one regexp. It has language flavors (configura-
tions) to allow each language to select an enabled set of features.
And, Oniguruma's Ruby flavor just disabled ONIG_OPTION_CAPTURE_GROUP.
For another example, Oniguruma itself supports the \Q\E feature for Perl
flavor. Ruby flavor disables \Q\E too.

The reason why Ruby flavor disabled ONIG_OPTION_CAPTURE_GROUP seems:

- ONIG_OPTION_CAPTURE_GROUP is not easy to understand (even Kosako said)
- using both named and unnamed capture at the time is considered as bad and confusing style in Ruby
- when it was discussed, there was no compatibility issue because there was no oniguruma gem

It was better for oniguruma gem to disable ONIG_OPTION_CAPTURE_GROUP as
long as it quotes 1.9. Otherwise, it had to explicitly state "the gem
should not be used as 1.9-compat layer, but a wrapper for Oniguruma."

--
Yusuke ENDOH mame@tsg.ne.jp

=end

#18 - 03/01/2010 10:23 PM - naruse (Yui NARUSE)

=begin
Is this example really correct?

```
Oniguruma::ORegexp.new('(?=([ ]{4}|\t)(?-\k))')
```

This can't match.
=end

#19 - 03/02/2010 03:11 AM - manveru (Michael Fellinger)

=begin
It's not correct, I simply shortened a longer expression.
The original is:

```
(?x)\G
(?=    ([ ]{4}|\t)
|  [#]{1,6}\s*+
|  [ ]{,3}(?[-_])([ ]{,2}\k){2,}[ \t]+$
)
```
=end

#20 - 03/02/2010 06:47 PM - naruse (Yui NARUSE)

- Status changed from Open to Closed

- % Done changed from 0 to 100

=begin
This issue was solved with changeset r26796.
Michael, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

=end

#21 - 03/02/2010 08:44 PM - naruse (Yui NARUSE)

=begin
r26796 allows your example to pass.

It fixes a bug of Oniguruma: captured groups in anchors (look-ahead, look-behind, and so on) are not removed
and unintended invalid backref error occur.
You hit this.

Thank you for reporting.
=end

#22 - 03/03/2010 12:02 PM - manveru (Michael Fellinger)

=begin
That's one out of many, and the shortest I could find.
My other samples still fail... here's a quick paste from when I try to convert the Markdown bundle: http://pastie.org/851062
=end

#23 - 03/04/2010 02:53 PM - coatl (caleb clausen)

=begin
It seems to me that Michael has a burning need for this feature and it can't be (easily) provided any other way. TextMate is one of the foremost development environments for ruby; it should be possible to rewrite it entirely in ruby but the lack of this feature makes that it hard to run textmate bundles. It appears there is a patch already. The objections to this feature amounted to "it's complicated for users to understand and no-one needs it". Which would be a perfectly valid justification if true, but clearly Michael does need it, and has been willing to argue extensively for it. In short, there are compelling reasons to accept this feature, and very little reason to reject it.

This issue has been closed, but it ought to be reopened. As Michael's last comment indicates, the fix supplied (r26796) does address the smallest of the problems he's seeing, but the broader issue is not yet resolved. It's a step in the right direction, but it looks like his original suggested new feature needs to be implemented in its full glory.
=end

#24 - 03/04/2010 03:00 PM - scritch (Vincent Isambart)

=begin

> The objections to this feature amounted to "it's complicated for users to understand and no-one needs it". Which would be a perfectly valid justification if true, but clearly Michael does need it, and has been willing to argue extensively for it. In short, there are compelling reasons to accept this feature, and very little reason to reject it.

A language cannot and should not implement all features request asked
by users. It would only end up in feature bloat.

=end

#25 - 03/04/2010 07:53 PM - hgs (Hugh Sasse)

=begin
On Thu, 4 Mar 2010, Vincent Isambart wrote:

> > The objections to this feature amounted to "it's complicated for users to understand and no-one needs it". Which would be a perfectly valid justification if true, but clearly Michael does need it, and has been willing to argue extensively for it. In short, there are compelling reasons to accept this feature, and very little reason to reject it.
>
> A language cannot and should not implement all features request asked
> by users. It would only end up in feature bloat.
>
> As a general principle, that is correct (see for example, Lua).
> My understanding is that this is a request for the complete implementation
> of an existing feature, i.e Onigoruma. As far as I can see pcre
> does implement this
> http://manpages.courier-mta.org/htmlman3/pcresyntax.3.html

Apparently PHP 5 has Oniguruma, but I can't see if it implements this
or not, but my blunt use of Google suggests it uses pcre to do this.

```
    Hugh
```

=end

#26 - 03/04/2010 08:41 PM - naruse (Yui NARUSE)

=begin
Hi,

(2010/03/03 12:02), Michael Fellinger wrote:

> Issue #2759 has been updated by Michael Fellinger.
>
> That's one out of many, and the shortest I could find.
> My other samples still fail... here's a quick paste from when I try to convert the Markdown bundle:
> http://pastie.org/851062

Hmm, I haven't heard using nonnamed captured group...

Try Regexp.new(pattern, 256) with following patch.
But without /G, it still seems to need some fixes.

--- a/re.c
+++ b/re.c
@@ -243,7 +243,7 @@ rb_memsearch(const void *x0, long m, const void *y0, long n, rb_encoding *enc)
#define KCODE_FIXED FL_USER4

#define ARG_REG_OPTION_MASK \

- (ONIG_OPTION_IGNORECASE|ONIG_OPTION_MULTILINE|ONIG_OPTION_EXTEND)
- (ONIG_OPTION_IGNORECASE|ONIG_OPTION_MULTILINE|ONIG_OPTION_EXTEND|ONIG_OPTION_CAPTURE_GROUP) #define ARG_ENCODING_FIXED 16 #define ARG_ENCODING_NONE 32

--
NARUSE, Yui naruse@airemix.jp

=end

#27 - 03/04/2010 08:50 PM - naruse (Yui NARUSE)

=begin
(2010/03/04 19:52), Hugh Sasse wrote:

> On Thu, 4 Mar 2010, Vincent Isambart wrote:
>
>> The objections to this feature amounted to "it's complicated for users to understand and no-one needs it". Which would be a perfectly valid justification if true, but clearly Michael does need it, and has been willing to argue extensively for it. In short, there are compelling reasons to accept this feature, and very little reason to reject it.
>
> A language cannot and should not implement all features request asked
> by users. It would only end up in feature bloat.
>
> As a general principle, that is correct (see for example, Lua).
> My understanding is that this is a request for the complete implementation
> of an existing feature, i.e Onigoruma. As far as I can see pcre
> does implement this
> http://manpages.courier-mta.org/htmlman3/pcresyntax.3.html
>
> Apparently PHP 5 has Oniguruma, but I can't see if it implements this
> or not, but my blunt use of Google suggests it uses pcre to do this.

As I wrote above, Ruby 1.9's regexp engine is a fork of Oniguruma.
It isn't a Oniguruma itself.
Our one has some feature changes and extension and limitation.

--
NARUSE, Yui naruse@airemix.jp

=end

#28 - 03/04/2010 09:54 PM - naruse (Yui NARUSE)

=begin
(2010/03/04 14:53), caleb clausen wrote:

> It seems to me that Michael has a burning need for this feature and
> it can't be (easily) provided any other way.

We can know how easy to provide only after implement and test it.
In this case, Michael want ONIG_OPTION_CAPTURE_GROUP and
ONIG_OPTION_DONT_CAPTURE_GROUP, but the flag he really needed
was ONIG_OPTION_CAPTURE_GROUP.

> TextMate is one of the
> foremost development environments for ruby; it should be possible to
> rewrite it entirely in ruby but the lack of this feature makes that
> it hard to run textmate bundles.
>
> It appears there is a patch already.

Do you really know how the patch is?

> The objections to this feature amounted to "it's complicated for
> users to understand and no-one needs it". Which would be a perfectly
> valid justification if true, but clearly Michael does need it, and
> has been willing to argue extensively for it. In short, there are
> compelling reasons to accept this feature, and very little reason to
> reject it.

I don't know what is really need by those regexps.
Oniguruma has many compile options and
our fork one has some difference from Oniguruma.

Possible traps are \d, \s, \w difference, duplicate charclass warning,
meaning of /m, Unicode properties, Capture history and so on.

If I fix this issue but some of regexp still doesn't work,
the goal of this ticket won't achieved.
So we can't start to run before the goal is concrete.

> This issue has been closed, but it ought to be reopened. As Michael's
> last comment indicates, the fix supplied (r26796) does address the
> smallest of the problems he's seeing, but the broader issue is not
> yet resolved. It's a step in the right direction, but it looks like
> his original suggested new feature needs to be implemented in its
> full glory.

I set the goal to pass following regexp:
/(?x)\G
(?= ([ ]{4}|\t)
| [#]{1,6}\s*+
| [ ]{,3}(?[-_])([ ]{,2}\k){2,}[\t]+$
)/
And it can pass after r 26796.

Anyway, We won't simply implement what reporter says.
Better design of language needs to know the use case of each function and,
think and think the API is really suitable to the use case.
So we always want to know what is the use case,
what is the true goal the reporter want to go.
This is because some feature request is rejected
even if its implementation is easy.

See also akr's good document:
http://www.a-k-r.org/pub/howto-persuade-matz-rubykaigi2008.pdf

--
NARUSE, Yui naruse@airemix.jp

=end

#29 - 03/05/2010 09:26 AM - manveru (Michael Fellinger)

=begin
Hi Naruse, Allowing for the flag to be passed is fine, I'm not that eager on the /g or /G syntax itself as long as I can do it another way.
I'll try your patch and get back to you, FWIW, this is almost identical to what I did in Rubinius, and it worked.
See my ticket at http://github.com/evanphx/rubinius/issues/issue/198 for the spec and my attempt to implement it (even if i know no YACC or C).
=end

#30 - 03/05/2010 02:51 PM - naruse (Yui NARUSE)

=begin
2010/3/5 Caleb Clausen vikkous@gmail.com:

> On 3/4/10, NARUSE, Yui naruse@airemix.jp wrote:
>
> > (2010/03/04 14:53), caleb clausen wrote:
> >
> > > It appears there is a patch already.
> >
> > Do you really know how the patch is?
>
> No. Michael referred to a discussion on the japanese ML and said there
> was a patch in it. As I can't understand japanese, I didn't try to
> look into it further. Yusuke Endoh said:
>
> > The reason why Ruby flavor disabled ONIG_OPTION_CAPTURE_GROUP seems:
> >
> > - ONIG_OPTION_CAPTURE_GROUP is not easy to understand (even Kosako said)
> > - using both named and unnamed capture at the time is considered as bad
> >   and confusing style in Ruby
> > - when it was discussed, there was no compatibility issue because there
> >   was no oniguruma gem

> I don't know if he was referring to that specific patch or not. I summarized this as:
>
> > The objections to this feature amounted to "it's complicated for users to understand and no-one needs it".
>
> Is my summary a mischaracterization? Are there objections to the patch beyond those above?

The patch they tought should be the same of my previous patch.
If it is all ok, it should be easy.

But it is not perfetct; those regexp can't be marshalized.
% ./ruby -e'Marshal.load(Marshal.dump(Regexp.new(%q/(a)(?n)\1\k/,256)))'
-e:1:in load': numbered backref/call is not allowed. (use name):
/(a)(?<n>n)\1\k<n>/ (RegexpError)
from -e:1:in'

You still think fixing Marshal should be easy.
But in marshal.c:
case T_REGEXP:
w_uclass(obj, rb_cRegexp, arg);
w_byte(TYPE_REGEXP, arg);
{
int opts = rb_reg_options(obj);
w_bytes(RREGEXP_SRC_PTR(obj), RREGEXP_SRC_LEN(obj), arg);
w_byte((char)opts, arg);
}
You know the value of this option is 256,
but current code marshalize the option as char...
It is difficult to extend this with compatibility.

This is only an example but it shall show the difficulty of
managing complex system.
This is because we are carefull.

--
NARUSE, Yui
naruse@airemix.jp

=end

#31 - 03/05/2010 04:31 PM - naruse (Yui NARUSE)

=begin
2010/3/5 Caleb Clausen vikkous@gmail.com:

> Can't you rewrite that as something like this:
>
> ```
> /untested, intended more as an example/
> case T_REGEXP:
>   w_uclass(obj, rb_cRegexp, arg);
>   {
>     int opts = rb_reg_options(obj);
>     if (opts<=0xFF)
>       w_byte(TYPE_REGEXP, arg);
>     else
>       w_byte(TYPE_WIDEREGEXP, arg);
>     w_bytes(RREGEXP_SRC_PTR(obj), RREGEXP_SRC_LEN(obj), arg);
>     if (opts<=0xFF)
>       w_byte((char)opts, arg);
>     else
>       w_int(opts, arg);
>   }
> ```
>
> That is, use the old format if the regexp options will fit in a byte, else use a new format with a wider field for options. There'd be similar logic on the unmarshal side. This new regexp format would be unreadable on older ruby versions, but then they won't know what to do with 256 in the options field anyway, so there's no loss there.

This code allows 1.9.2 to dump and load older regexps and new extended regexps.
But when older than Ruby 1.9.2 can't load all of dumped data
which include such dumped wideregexp,

I'm thinking the way to dump the regexp as normal object and invisible instance
encoding.

--
NARUSE, Yui
naruse@airemix.jp

=end