Ruby master - Bug #13167
Dir.glob is 25x slower since Ruby 2.2
01/30/2017 10:11 AM - ahorek (Pavel Rosický)

Status: Closed
Priority: Normal
Assignee: h.shirosaki (Hiroshi Shirosaki)
Target version: 2.4.0
Backport: 2.2: UNKNOWN, 2.3: UNKNOWN, 2.4: UNKNOWN

Description
Hello,
we've found a huge speed regression in our Rails app. After some digging the reason is in Dir.glob method which is much slower since Ruby 2.2.6. This is probably Windows only!

This code is used heavily in Rails for partial lookups:

Dir.glob('c:/test/myapp/app/views/common/_menu_stats{.en,}{.html,}{}{.erb,.builder,.raw,.ruby,.jbuilder,.coffee,}')

Comparison (x64):
  jruby 9.1.7.0 2540 i/s
  ruby 2.1.5 2568 i/s
  ruby 2.1.9 2569 i/s
  ruby 2.2.6 99 i/s 25 times slower!
  ruby 2.3.3 102 i/s
  ruby 2.4.0 103 i/s

I would like to help, but I don't know much about Ruby C internals. Please let me know if you need any additional info. Now we're stuck at 2.1.9 because this issue makes the development on more recent versions unusable.

Related issues:
Related to Ruby master - Bug #10015: Performance regression in Dir#
Closed
Related to Ruby master - Feature #13873: Optimize Dir.glob with FNM_EXTGLOB
Closed

Associated revisions
Revision 2a119042 - 09/22/2018 01:11 AM - shirosaki
dir.c: performance fix with braces
Braces were expended before ruby_glob0(). This caused to call replace_real_basename() for same plain patterns repeatedly.
Move blace expansion into glob_helper() in ruby_glob0() to reduce replace_real_basename() call.
This fix changes the order of glob results.
[Feature #13167] [Fix GH-1864]
git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@64810 b2dd03c8-39d4-4d8f-98f8-823fe69b0800
e

Revision 64810 - 09/22/2018 01:11 AM - shirosaki
dir.c: performance fix with braces
Braces were expended before ruby_glob0(). This caused to call replace_real_basename() for same plain patterns repeatedly.
Move blace expansion into glob_helper() in ruby_glob0() to reduce replace_real_basename() call.
This fix changes the order of glob results.
[Feature #13167] [Fix GH-1864]

e

Revision 64810 - 09/22/2018 01:11 AM - shirosaki
dir.c: performance fix with braces
Braces were expended before `ruby_glob0()`. This caused to call `replace_real_basename()` for same plain patterns repeatedly. Move brace expansion into `glob_helper()` in `ruby_glob0()` to reduce `replace_real_basename()` call. This fix changes the order of glob results.

[Feature #13167] [Fix GH-1864]

Revision b1432544 - 09/25/2018 03:31 PM - shirosaki
dir.c: fix memory leak of glob with braces
join_path uses malloc. So free is required.
[Feature #13167]
git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@64835 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 64835 - 09/25/2018 03:31 PM - shirosaki
dir.c: fix memory leak of glob with braces
join_path uses malloc. So free is required.
[Feature #13167]

Revision 64835 - 09/25/2018 03:31 PM - shirosaki
dir.c: fix memory leak of glob with braces
join_path uses malloc. So free is required.
[Feature #13167]

Revision f73d504c - 09/25/2018 03:31 PM - shirosaki
dir.c: fix glob with recursive and brace
Fixed bug that glob with recursive and braces (**/{a,b}) pattern fails.
[Feature #13167]
git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@64836 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 64836 - 09/25/2018 03:31 PM - shirosaki
dir.c: fix glob with recursive and brace
Fixed bug that glob with recursive and braces (**/{a,b}) pattern fails.
[Feature #13167]

Revision 64836 - 09/25/2018 03:31 PM - shirosaki
dir.c: fix glob with recursive and brace
Fixed bug that glob with recursive and braces (**/{a,b}) pattern fails.
[Feature #13167]

History
#1 - 01/30/2017 11:25 AM - nobu (Nobuyoshi Nakada)
- Related to Bug #10015: Performance regression in Dir#[] added

#2 - 01/30/2017 11:33 AM - nobu (Nobuyoshi Nakada)
- Description updated

#3 - 07/24/2017 07:42 PM - aherek (Pavel Rosicky)
- File logruby21.txt added
- File logruby24.txt added

I used Procmon.exe [https://live.sysinternals.com](https://live.sysinternals.com) to monitor system calls and it looks like ruby 2.4.1 is traversing the whole directory tree over and over again for each {} matcher. This should be definitely avoided!

take a look, the same single call for a Dir.glob takes
30 sys-calls on Ruby 2.1.9 but 2086 sys-calls on Ruby 2.4.1!

Ruby 2.1.9 just tries to open all combinations without checking the directory structure

c:/test/myapp/app/views/common/_menu_stats.en.html.erb open
...but Ruby 2.4.1 behaves like this

c:/ open
c:/ stats
c:/ close
c:/test open
c:/test stats
c:/test close
c:/test/myapp open
c:/test/myapp stats
c:/test/myapp close
...c:/test/myapp/app/views/common/_menu_stats.en.html.erb open
c:/test/myapp/app/views/common/_menu_stats.en.html.erb stats
c:/test/myapp/app/views/common/_menu_stats.en.html.erb close
*** AND AGAIN ***
c:/ open
c:/ stats
c:/ close
c:/test open
c:/test stats
c:/test close
c:/test/myapp open
c:/test/myapp stats
c:/test/myapp close
...c:/test/myapp/app/views/common/_menu_stats.en.html.builder open
c:/test/myapp/app/views/common/_menu_stats.en.html.builder stats
c:/test/myapp/app/views/common/_menu_stats.en.html.builder close
*** AND AGAIN ***
c:/ open
c:/ stats
c:/ close
c:/test open
c:/test stats
c:/test close
c:/ etc ...
ksh93, zsh, bash all exhibit the same behavior, even. And it appears a major refactoring of dir.c is necessary to support optimizing away redundant readdir (getdents on Linux) calls.

### #6 - 07/24/2017 10:40 PM - ahorek (Pavel Rosicky)

There isn't noticeable difference on Linux, it's even slightly faster.

Linux

```
2.1.9 77991 i/s
2.4.1 78497 i/s
```

Windows

```
2.1.9 1143000 i/s
2.4.1 39829 i/s
```

https://github.com/ruby/ruby/blob/trunk/dir.c

### #7 - 07/25/2017 02:51 AM - normalperson (Eric Wong)

pdahorek@seznam.cz wrote:

There isn't noticeable difference on Linux, it's even slightly faster.

The problem isn't the noticeability in Linux. I suspect the problem here is Linux hides performance problems with fast syscalls:

Linux

```
2.1.9 77991 i/s
2.4.1 78497 i/s
```

Windows

```
2.1.9 1143000 i/s
2.4.1 39829 i/s
```

Are those numbers on the same hardware? If so, it's because our glob performance on Linux always sucked :)

So, I suspect the performance on 2.1.9 was good because Ruby used Win32-specific APIs; but when the code path changed to use work the same on both systems, it got silly slow.

I've been having a tough time figuring out what changes in the 2.1..2.2 era did what over time, especially on a platform I don't run...

Can you run "git bisect" to narrow down the performance problem to a particular commit?

Thanks.

### #8 - 07/25/2017 11:22 AM - ahorek (Pavel Rosicky)

yes, it's on the same hardware and also with the same file path. I used Bash on Windows which could be slower then the native Windows app. So I also compared it on a native Ubuntu and 2.4.1 is faster on it

```
2.1.9 695000 i/s
2.4.1 766827 i/s
```

after some digging I found out that this change introduced the problem

https://bugs.ruby-lang.org/issues/5994
around this commit
https://github.com/ruby/ruby/commit/5b92c0bea3dc23b0c2be356bedafdd4e7f9110d7

#9 - 07/25/2017 11:37 AM - ahorek (Pavel Rosický)
https://github.com/ruby/ruby/pull/1669

2.1.9        1143000 i/s
2.4.1        39829 i/s
2.5.0        40730 i/s
2.5.0 + patch 936338 i/s

this patch is probably wrong, but it's a good place to start

normalperson (Eric Wong) - could you take a look?

#10 - 07/25/2017 04:33 PM - normalperson (Eric Wong)
pdahorek@seznam.cz wrote:

Issue #13167 has been updated by ahorek (Pavel Rosický).

https://github.com/ruby/ruby/pull/1669

2.1.9        1143000 i/s
2.4.1        39829 i/s
2.5.0        40730 i/s
2.5.0 + patch 936338 i/s

Thanks.

this patch is probably wrong, but it's a good place to start

normalperson (Eric Wong) - could you take a look?

This is nobu (Nobuyoshi Nakada)'s job, since he made the original change
and knows far more about case-insensitive FSes than I do.

I think the performance on Linux is a separate problem.
The 766827 i/s you got on Ubuntu is still worse than Win32;
so I think that could be improved, possibly on all platforms.

#11 - 07/25/2017 07:15 PM - ahorek (Pavel Rosický)

Sure, faster glob could make a big difference in overall performance. It's a very good candidate for optimalization.

for Windows and maybe other case-insensitive FS that shares the same codepath we should
avoid (or cache) recurring tree-stats for each magic {.txt} which are very expensive ( explained here https://bugs.ruby-lang.org/issues/13167#note-3 )

#12 - 07/31/2017 02:22 PM - ahorek (Pavel Rosický)

there's a good article about this
https://research.swtch.com/glob
https://perl6.perl.org/perl.git/commitdiff/33252c318625f3c6c89b816ee88481940e3e8f95?hp=57ab6c610267dba697199c82564258af7d391c1

take a look at the python's implementation
https://github.com/python/cpython/commits/3.6/Lib/glob.py

Ruby has tons of ifs, gotos and recursions for many special cases, it's not very readable and I have a tough time to understand what's happening
For instance this Windows problem is solved, Python has different approach, because results of the glob will be the same even with the previous
Ruby 2.1 implementation, you just need to normalize the output according to realpaths (I expect that I can't create two files or directories with a same
name like "test.txt" and "Test.txt", am I right?)

simpler example

Dir.glob("c:/test/myapp")

Python
CreateFile
QueryInformationVolume
QueryAllInformationFile
CloseFile
I think that Python has fully compatible syntax, even with \{\} expansion and also works fast on Windows (case sensitive)

#13 - 08/27/2017 04:22 PM - ahorek (Pavel Rosicky)
https://github.com/ruby/ruby/pull/1685

I reverted nobu's change and instead of recursion for simple patterns I want to call "replace_real_basename" only for results. There's no need to call it for each directory because the result will always be same. It's not final and I'll be really glad if someone more experienced can help me with it. Also other parts like path normalization could be called only once.

What do you think about optimizing most common use cases like

```
Dir.glob('/test/file.{html,erb}')
Dir.glob('/test/*')
```

Ruby 2.4.1

<table>
<thead>
<tr>
<th></th>
<th>i/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>plain</td>
<td>1089.3</td>
</tr>
<tr>
<td>*</td>
<td>324.9</td>
</tr>
<tr>
<td>braces</td>
<td>37.7</td>
</tr>
<tr>
<td>* 2</td>
<td>8.6</td>
</tr>
<tr>
<td>**</td>
<td>3.1</td>
</tr>
</tbody>
</table>

Trunk (2.5)

<table>
<thead>
<tr>
<th></th>
<th>i/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>plain</td>
<td>1013.7</td>
</tr>
<tr>
<td>*</td>
<td>569.6</td>
</tr>
<tr>
<td>braces</td>
<td>34.7</td>
</tr>
<tr>
<td>* 2</td>
<td>23.3</td>
</tr>
<tr>
<td>**</td>
<td>2.8</td>
</tr>
</tbody>
</table>

Trunk (2.5) + patch

<table>
<thead>
<tr>
<th></th>
<th>i/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>plain</td>
<td>18020.3</td>
</tr>
<tr>
<td>*</td>
<td>1432.5</td>
</tr>
<tr>
<td>braces</td>
<td>917.7</td>
</tr>
<tr>
<td>* 2</td>
<td>25.4</td>
</tr>
<tr>
<td>**</td>
<td>3.1</td>
</tr>
</tbody>
</table>

Ruby 2.1.9

<table>
<thead>
<tr>
<th></th>
<th>i/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>plain</td>
<td>20519.1</td>
</tr>
<tr>
<td>*</td>
<td>1905.2</td>
</tr>
<tr>
<td>braces</td>
<td>1094.0</td>
</tr>
<tr>
<td>* 2</td>
<td>46.4</td>
</tr>
<tr>
<td>**</td>
<td>6.7</td>
</tr>
</tbody>
</table>

btw Python's performace is even faster then Ruby 2.1.9 (20x), this is a huge difference.
ahorek (Pavel Rosický) wrote:

There's no need to call it for each directory because the result will always be same.

It is not same.
Path components in middle also should be replaced.

replace_real_basename() is called for same head plain paths because braces are expanded early before ruby_glob0().

Moving braces expansion to later phase in glob_helper() is a way to reduce replace_real_basename(). The idea is same as #13873.

Another idea is caching real name of each directory and use the cache.

I attached a patch and benchmark script.

Here is my benchmark result.

+ patch: 0001-dir.c-performance-fix-with-braces.patch
+ cache: 0001-dir.c-performance-fix-with-braces-using-cache.patch

braces:
Dir["v:/test/myapp/app/views/common/_menu_stats{.en,}{.html,}{}{.erb,.builder,.raw,.ruby,.jbuilder,.coffee,}"

recursive:
Dir["v:/test/myapp/app/views/**/_menu_stats{.en,}{.html,}{}{.erb,.builder,.raw,.ruby,.jbuilder,.coffee,}"

On Windows 10
ruby 2.5.0dev (2017-09-11 trunk 59831) [x64-mingw32]

<table>
<thead>
<tr>
<th></th>
<th>braces</th>
<th>+ patch</th>
<th>recursive</th>
</tr>
</thead>
<tbody>
<tr>
<td>i/s</td>
<td>148.111</td>
<td>1.809k</td>
<td>71.280</td>
</tr>
<tr>
<td>in</td>
<td>742.000</td>
<td>9.078k</td>
<td>357.000</td>
</tr>
<tr>
<td>5.015963s</td>
<td></td>
<td>5.027256s</td>
<td>5.014841s</td>
</tr>
</tbody>
</table>

On Linux(Ubuntu 16.04)
ruby 2.5.0dev (2017-09-11 trunk 59831) [x86_64-linux]

<table>
<thead>
<tr>
<th></th>
<th>braces</th>
<th>+ patch</th>
<th>recursive</th>
</tr>
</thead>
<tbody>
<tr>
<td>i/s</td>
<td>6.171k</td>
<td>11.241k</td>
<td>720.448</td>
</tr>
<tr>
<td>in</td>
<td>31.408k</td>
<td>57.252k</td>
<td>3.640k</td>
</tr>
<tr>
<td>5.090401s</td>
<td></td>
<td>5.093467s</td>
<td></td>
</tr>
</tbody>
</table>

#16 - 09/26/2017 07:24 PM - naruse (Yui NARUSE)
- Related to Feature #13873: Optimize Dir.glob with FNM_EXTGLOB added

#17 - 02/05/2018 04:00 PM - sfceorge (Simon George)
Is there any progress on this, I see feature #13873 is related, but it looks like that got reverted again? https://bugs.ruby-lang.org/issues/13873

I ran into this issue with Rails; when a request doesn't specify the format Rails uses this Glob and it takes 10x longer to respond. In our real-world app that means collection partials that normally take 40ms each now take 300ms, thus the whole page takes 5 or more seconds! There's an issue I opened with Rails but it seems this is the root cause https://github.com/rails/rails/issues/30502

#18 - 02/06/2018 02:08 AM - h.shirosaki (Hiroshi Shirosaki)
- File deleted (0001-dir.c-performance-fix-with-braces.patch)

#19 - 02/06/2018 02:22 AM - h.shirosaki (Hiroshi Shirosaki)
#13873 seems reverted in order to avoid test changes (incompatibility of the order).
My patch (0001-dir.c-performance-fix-with-braces.patch) passes test-all and test-rubyspec without test changes.
It would be more similar to trunk behavior than #13873 implementation although not 100% compatible.
I rebased a patch for latest trunk and did some format fix.

#20 - 08/05/2018 04:46 PM - ahorek (Pavel Rosický)
- File bench_dir_glob2.rb added
- File windows_recursive.png added
- File windows_list.png added
- File windows_braces.png added
- File linux_recursive.png added
- File linux_list.png added
- File linux_braces.png added

#21 - 08/05/2018 05:40 PM - ahorek (Pavel Rosický)

h.shirosaki (Hiroshi Shirosaki), thanks for your work on this. I tested your patch 0001-dir.c-performance-fix-with-braces.patch (ruby head + braces) based on the current trunk [https://github.com/ruby/ruby/pull/1864](https://github.com/ruby/ruby/pull/1864)

environment:

Samsung 850 Pro 250GB
AMD 8350FX 8C
Windows 10 and Ubuntu
16GB DDR3

eruby 2.6.0dev (2018-08-05 trunk 64192) [x86_64-linux]
eruby 9.2.1.0-SNAPSHOT (2.5.0) 2018-08-02 5aa064b Java HotSpot(TM) 64-Bit Server VM 10.0.1+10 on 10.0.1+10 +jit [linux-x86_64]

ratio (faster than trunk)

linux braces 1.26x
linux recursive 0.99x
windows braces 10.75x
windows recursive 1.66x

I think the patch fixes the main problem I originally reported. Especially "windows braces" is almost 11-times faster, almost as fast as ruby 2.1.9 was.

I also tested it with my rspec suite and it runs 2.14x faster, this is a huge perf difference. It passes all tests.

ruby trunk
22 minutes 46 seconds
ruby trunk + patch
10 minutes 5 seconds

cc nobu (Nobuyoshi Nakada) if you have time, could you please review it?

<table>
<thead>
<tr>
<th></th>
<th>list</th>
<th>braces</th>
<th>recursive</th>
</tr>
</thead>
<tbody>
<tr>
<td>linux</td>
<td>12.627k</td>
<td>4.332k</td>
<td>81.603k</td>
</tr>
<tr>
<td>ruby 2.1.9</td>
<td>(± 1.6%) i/s</td>
<td>(± 1.9%) i/s</td>
<td>(± 1.2%) i/s</td>
</tr>
<tr>
<td></td>
<td>63.232k in</td>
<td>21.889k in</td>
<td>413.000k in</td>
</tr>
<tr>
<td>ruby 2.5.0</td>
<td>11.752k</td>
<td>4.305k</td>
<td>248.731k</td>
</tr>
<tr>
<td></td>
<td>(± 1.3%) i/s</td>
<td>(± 2.0%) i/s</td>
<td>(± 1.6%) i/s</td>
</tr>
<tr>
<td></td>
<td>59.176k in</td>
<td>21.600k in</td>
<td>1.248k in</td>
</tr>
<tr>
<td>ruby head</td>
<td>12.128k</td>
<td>4.667k</td>
<td>254.704k</td>
</tr>
<tr>
<td></td>
<td>(± 2.4%) i/s</td>
<td>(± 3.1%) i/s</td>
<td>(± 2.0%) i/s</td>
</tr>
<tr>
<td></td>
<td>60.840k in</td>
<td>23.613k in</td>
<td>1.275k in</td>
</tr>
<tr>
<td>ruby head + braces</td>
<td>12.123k</td>
<td>5.885k</td>
<td>251.895k</td>
</tr>
<tr>
<td></td>
<td>(± 3.3%) i/s</td>
<td>(± 2.2%) i/s</td>
<td>(± 2.0%) i/s</td>
</tr>
<tr>
<td></td>
<td>61.048k in</td>
<td>29.784k in</td>
<td>1.275k in</td>
</tr>
<tr>
<td></td>
<td>list</td>
<td>iops</td>
<td>total</td>
</tr>
<tr>
<td>---------------</td>
<td>------------</td>
<td>-------------</td>
<td>-----------</td>
</tr>
<tr>
<td>jruby-head</td>
<td>9.931k (±2.4%)</td>
<td>49.764k</td>
<td>5.014070s</td>
</tr>
<tr>
<td></td>
<td>4.758k (±1.7%)</td>
<td>23.940k</td>
<td>5.032956s</td>
</tr>
<tr>
<td></td>
<td>35.933 (±5.6%)</td>
<td>180.000</td>
<td>5.022796s</td>
</tr>
<tr>
<td>Windows</td>
<td>2.683k (±5.9%)</td>
<td>13.566k</td>
<td>5.077196s</td>
</tr>
<tr>
<td>ruby 2.1.9</td>
<td>1.200k (±3.2%)</td>
<td>6.000k</td>
<td>5.005971s</td>
</tr>
<tr>
<td></td>
<td>111.844 (±0.9%)</td>
<td>561.000</td>
<td>5.016557s</td>
</tr>
<tr>
<td>ruby 2.5.0</td>
<td>945.309 (±3.0%)</td>
<td>4.794k</td>
<td>5.076069s</td>
</tr>
<tr>
<td></td>
<td>67.879 (±2.9%)</td>
<td>342.000</td>
<td>5.041694s</td>
</tr>
<tr>
<td></td>
<td>33.314 (±3.0%)</td>
<td>168.000</td>
<td>5.046526s</td>
</tr>
<tr>
<td>ruby head</td>
<td>1.001k (±1.8%)</td>
<td>5.049k</td>
<td>5.047494s</td>
</tr>
<tr>
<td></td>
<td>72.145 (±2.9%)</td>
<td>364.000</td>
<td>5.046341s</td>
</tr>
<tr>
<td></td>
<td>34.943 (±2.9%)</td>
<td>177.000</td>
<td>5.046275s</td>
</tr>
<tr>
<td>ruby head + braces</td>
<td>1.001k (±1.3%)</td>
<td>5.049k</td>
<td>5.044865s</td>
</tr>
<tr>
<td></td>
<td>773.822 (±0.9%)</td>
<td>3.927k</td>
<td>5.075205s</td>
</tr>
<tr>
<td></td>
<td>58.596 (±1.7%)</td>
<td>295.000</td>
<td>5.034900s</td>
</tr>
<tr>
<td>jruby-head</td>
<td>5.121k (?1.3%)</td>
<td>25.935k</td>
<td>5.064926s</td>
</tr>
<tr>
<td></td>
<td>1.308k (?2.1%)</td>
<td>6.625k</td>
<td>5.066130s</td>
</tr>
<tr>
<td></td>
<td>9.987 (?0.0%)</td>
<td>50.000</td>
<td>5.008338s</td>
</tr>
</tbody>
</table>

#22 - 08/09/2018 07:55 AM - h.shirosaki (Hiroshi Shirosaki)
- Assignee set to nobu (Nobuyoshi Nakada)
- Status changed from Open to Assigned

#23 - 09/13/2018 07:53 AM - nobu (Nobuyoshi Nakada)
- Assignee changed from nobu (Nobuyoshi Nakada) to h.shirosaki (Hiroshi Shirosaki)

Thank you for the patch, let's try, please commit the patch for braces.

#24 - 09/13/2018 07:53 AM - naruse (Yui NARUSE)

0001-dir.c-performance-fix-with-braces.patch
It would be more similar to trunk behavior than #13873 implementation although not 100% compatible.

I'm wondering whether this incompatibility is critical or not.
Anyway the easiest way is just merge it and wait the feedback from Rails.

#25 - 09/22/2018 01:11 AM - Anonymous
- Status changed from Assigned to Closed

Applied in changeset trunk|r64810.

dir.c: performance fix with braces

Braces were expended before ruby_glob0(). This caused to call replace_real_basename() for same plain patterns repeatedly.
Move brace expansion into glob_helper() in ruby_glob0() to reduce replace_real_basename() call.
This fix changes the order of glob results.
[Feature #13167] [Fix GH-1864]
After this commit is merged, some CIs that has -DVM_CHECK_MODE=2 and continue to test latest revision started to randomly crash 
"TestGem#test_load_plugins":
http://ci.rvm.jp/results/trunk-asserts@silicon-docker
http://ci.rvm.jp/results/trunk-vm-asserts@silicon-docker

Their logs will be lost after 3 days, so I attach persisted failed logs too:
https://gist.github.com/ko1/2c905ef9194b72701bea1fa5cb227f0
https://gist.github.com/ko1/4f9af0b4a2e48600467ca0a75dced58
https://gist.github.com/ko1/ba7cc479072764cb46482f112811d4b6
... and more

There may be a possibility that rubygems will become unstable by this (but currently it's reproductive only when -DVM_CHECK_MODE=2 is used),
and I'm writing here since CI notifies the failure too often.

Is that fixed by r64849? Thanks for the patch.

I hope so too (not confident enough since it was a random failure). Thank you for your attention to these CIs.

Files

<table>
<thead>
<tr>
<th>File Name</th>
<th>Size</th>
<th>Date</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>logruby24.txt</td>
<td>484 KB</td>
<td>07/24/2017</td>
<td>ahlerek (Pavel Rosicky)</td>
</tr>
<tr>
<td>logruby21.txt</td>
<td>10.8 KB</td>
<td>07/24/2017</td>
<td>ahlerek (Pavel Rosicky)</td>
</tr>
<tr>
<td>bench_dir_glob.rb</td>
<td>880 Bytes</td>
<td>09/11/2017</td>
<td>h.shirosaki (Hiroshi Shirosaki)</td>
</tr>
<tr>
<td>0001-dir.c-performance-fix-with-braces-using-cache.patch</td>
<td>5.84 KB</td>
<td>09/11/2017</td>
<td>h.shirosaki (Hiroshi Shirosaki)</td>
</tr>
<tr>
<td>0001-dir.c-performance-fix-with-braces.patch</td>
<td>8.64 KB</td>
<td>02/06/2018</td>
<td>h.shirosaki (Hiroshi Shirosaki)</td>
</tr>
<tr>
<td>linux_braces.png</td>
<td>24.1 KB</td>
<td>08/05/2018</td>
<td>ahlerek (Pavel Rosicky)</td>
</tr>
<tr>
<td>linux_list.png</td>
<td>23.5 KB</td>
<td>08/05/2018</td>
<td>ahlerek (Pavel Rosicky)</td>
</tr>
<tr>
<td>linux_recursive.png</td>
<td>26.6 KB</td>
<td>08/05/2018</td>
<td>ahlerek (Pavel Rosicky)</td>
</tr>
<tr>
<td>windows_braces.png</td>
<td>23.6 KB</td>
<td>08/05/2018</td>
<td>ahlerek (Pavel Rosicky)</td>
</tr>
<tr>
<td>windows_list.png</td>
<td>23.1 KB</td>
<td>08/05/2018</td>
<td>ahlerek (Pavel Rosicky)</td>
</tr>
<tr>
<td>windows_recursive.png</td>
<td>28 KB</td>
<td>08/05/2018</td>
<td>ahlerek (Pavel Rosicky)</td>
</tr>
<tr>
<td>bench_dir_glob2.rb</td>
<td>982 Bytes</td>
<td>08/05/2018</td>
<td>ahlerek (Pavel Rosicky)</td>
</tr>
</tbody>
</table>