Ruby - Bug #14002

libruby soname is changing in Ruby 2.4.x while it should always be libruby.so.2.4.0

10/11/2017 11:11 AM - vo.x (Vit Ondruch)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.4.2p198 (2017-09-14 revision 59899) [x86_64-linux]	**Backport:**	2.3: UNKNOWN, 2.4: REQUIRED

Description

Since the Ruby 2.4.x keeps ABI compatibility with Ruby 2.4.0 (I hope at least), the so name should keep the libruby.so.2.4.0 during the whole Ruby 2.4 lifetime. This always used to be true, while in Ruby 2.4, the soname is surprisingly changing (and it affects Fedora users 1).

Actually this is not just about the soname, the RbConfig::CONFIG["TEENY"] value is impacted by this as well. This was always "0" since the patch releases were abandoned. But now it changes.

I tried to revert r53566, r53500 and r5347 but apparently, this was not enough :/

Related issues:

Has duplicate Ruby - Bug #14001: libruby soname is changing in Ruby 2.4.x whi...	**Rejected**

Associated revisions

Revision c739150c1e7ed9012e07a973c9bd280bc3cb376a - 10/14/2017 03:35 PM - nobu (Nobuyoshi Nakada)

configure.ac: LIBRUBY_SONAME

- configure.ac (LIBRUBY_SONAME): add new variable for the name of
  the library name with compatibility version.
  [ruby-core:83208] [Bug #14002]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@60180 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision c739150c - 10/14/2017 03:35 PM - nobu (Nobuyoshi Nakada)

configure.ac: LIBRUBY_SONAME

- configure.ac (LIBRUBY_SONAME): add new variable for the name of
  the library name with compatibility version.
  [ruby-core:83208] [Bug #14002]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@60180 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision b625191054a66877ea57954db0a81fb34281cbfc - 10/15/2017 02:31 AM - nobu (Nobuyoshi Nakada)

configure.ac: fix SONAME

- configure.ac (RUBY_SO_NAME): revert $(RUBY_API_VERSION:.=) to
  $(MAJOR)$(MINOR), as a string in middle is not replaced.
  [ruby-core:83208] [Bug #14002]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@60184 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision b6251910 - 10/15/2017 02:31 AM - nobu (Nobuyoshi Nakada)

configure.ac: fix SONAME

- configure.ac (RUBY_SO_NAME): revert $(RUBY_API_VERSION:.=) to
  $(MAJOR)$(MINOR), as a string in middle is not replaced.
  [ruby-core:83208] [Bug #14002]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@60184 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision ae0eea21bf52745c6fbb812f9dd4897ae27e36ed - 10/16/2017 04:15 AM - nobu (Nobuyoshi Nakada)

fix missing variables in ruby.pc

- configure.ac (LIBRUBY_SO): get rid of referrence to LIBRUBY_SONAME which is not present in ruby.pc.
- template/ruby.pc.in (RUBY_API_VERSION, SOEXT): add new variables. [ruby-core:83208] [Bug #14002]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@60187 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision ae0eea21 - 10/16/2017 04:15 AM - nobu (Nobuyoshi Nakada)

fix missing variables in ruby.pc

- configure.ac (LIBRUBY_SO): get rid of referrence to LIBRUBY_SONAME which is not present in ruby.pc.
- template/ruby.pc.in (RUBY_API_VERSION, SOEXT): add new variables. [ruby-core:83208] [Bug #14002]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@60187 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 90ab739ae488eeb1aee00d1b0609f605c26041ad - 10/16/2017 05:00 AM - nobu (Nobuyoshi Nakada)

configure.ac: fix SOEXT on Windows

- configure.ac (SOEXT): shoud be "dll" on Windows. [ruby-core:83208] [Bug #14002]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@60190 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 90ab739a - 10/16/2017 05:00 AM - nobu (Nobuyoshi Nakada)

configure.ac: fix SOEXT on Windows

- configure.ac (SOEXT): shoud be "dll" on Windows. [ruby-core:83208] [Bug #14002]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@60190 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 10/11/2017 11:45 AM - hsbt (Hiroshi SHIBATA)

- Has duplicate Bug #14001: libruby soname is changing in Ruby 2.4.x while it should be always libruby.so.2.4.0 added

#2 - 10/11/2017 12:05 PM - vo.x (Vit Ondruch)

- Subject changed from libruby soname is changing in Ruby 2.4.x while it should be always libruby.so.2.4.0 to libruby soname is changing in Ruby 2.4.x while it should always be libruby.so.2.4.0

#3 - 10/11/2017 02:04 PM - jeremyevans0 (Jeremy Evans)

vo.x (Vit Ondruch) wrote:

> Since the Ruby 2.4.x keeps ABI compatibility with Ruby 2.4.0 (I hope at least)

Strict ABI compatibility is not kept. Nonstatic functions are added and even sometimes removed in backported patches, try diffing the tags for 2.4.2 and 2.4.1. In most cases, the functions are not something likely to callable by the user, but if you are strict about bumping whenever symbols changes in the .so, you'll need both minor and major bumps.

> Actually this is not just about the soname, the RbConfig::CONFIG["TEENY"] value is impacted by this as well. This was always "0" since the patch releases were abandoned. But now it changes.

I would expect the TEENY value to change for all new releases.

#4 - 10/11/2017 02:12 PM - vo.x (Vit Ondruch)

jeremyevans0 (Jeremy Evans) wrote:

> vo.x (Vit Ondruch) wrote:
>
> > Since the Ruby 2.4.x keeps ABI compatibility with Ruby 2.4.0 (I hope at least)

> Strict ABI compatibility is not kept. Nonstatic functions are added and even sometimes removed in backported patches, try diffing the tags for 2.4.2 and 2.4.1. In most cases, the functions are not something likely to callable by the user, but if you are strict about bumping whenever symbols changes in the .so, you'll need both minor and major bumps.
>
> > Actually this is not just about the soname, the RbConfig::CONFIG["TEENY"] value is impacted by this as well. This was always "0" since the patch releases were abandoned. But now it changes.
>
> I would expect the TEENY value to change for all new releases.

If nothing else, this was never the case. Reading this 1, I am inclined to interpret it as that the TEENY should not change ....

#5 - 10/11/2017 02:57 PM - jeremyevans0 (Jeremy Evans)

vo.x (Vit Ondruch) wrote:

> If nothing else, this was never the case.

It is true that this changed in 2.4, I consider it a bugfix. RUBY_PROGRAM_VERSION should be "#{MAJOR}.#{MINOR}.#{TEENY}".

> Reading this 1, I am inclined to interpret it as that the TEENY should not change ....

Here's what it says:

```
Format: X.Y.Z
X: major version; increased when incompatible change which can’t be released in MINOR
Y: minor version; increased every christmas, may be API incompatible (always ABI incompatible)
Z: teeny; security or bug fix which maintains API compatibility
```

I'm not sure how that could be more clear that ruby 2.4.2 has major 2, minor 4, teeny 2 and ruby 2.4.1 has major 2, minor 4, teeny 1.

Note that it says API compatibility is maintained in teeny releases, but there is no mention of maintaining ABI compatibility. It guarantees ABI breakage between minor versions, but it doesn't specify ABI compatibility for teeny versions, so teeny versions are allowed to break ABI (and frequently do, at least until the branch enters security maintenance phase).

#6 - 10/11/2017 03:07 PM - nobu (Nobuyoshi Nakada)

LIBRUBY_SO is not for "SONAME" in your terminology, it's just the real file name of the shared library, which is probably "SOVERSION path" in your words.
We haven't provided a variable for the "SONAME", which may be needed.

#7 - 10/11/2017 05:54 PM - shevegen (Robert A. Heiler)

I had a look at the link provided above at the bug-entry of red hat, mentioning how vim links in to ruby and does not detect the correct ruby version. From the behaviour, I think that the vim code to detect which ruby version the operating system has, is wrong, just as Ben Boeckel wrote in the bug report - but I figure that some other programs (or the developers) may also not be entirely sure what the "proper way" to detect the ruby version in use is. I mean the .so numbering alone, by the way.

The awkward thing is that in that bug report, they already use RbConfig - so I honestly have no idea how they fail to detect the right ruby version, if they have RbConfig? I understand it to some extent if it is only on .so files, but ... RbConfig ... how can you make wrong guesses with it. :\ :/

Link to the bug discussion is at: https://bugzilla.redhat.com/show_bug.cgi?id=1499928

#8 - 10/12/2017 08:05 AM - zdohnal (Zdenek Dohnal)

From Vim configure.ac:

libruby=$vi_cv_path_ruby -r rbconfig -e "puts $ruby_rbconfig::CONFIG[['LIBRUBY_SO']]"

I thought LIBRUBY_SO provides SONAME of libruby (sorry, that fact it is actually provides name of .so file, IMHO it's not very intuitive). This way of getting libruby is in vim upstream since 7.3.678, they used RUBY_SO_NAME before (which returns "ruby", which is again not very intuitive).
If LIBRUBY_SO really doesn't provide SONAME, the issue is there isn't any other ways how to get SONAME ("libruby.so.2.4") by rbconfig (I humbly apologize, but using "hacks" like splitting output of rbconfig[LIBRUBY_ALIAS] doesn't look like proper solution).

So I see two ways of solving it:

1. stop bumping teeny number of soname like it was done in the past (I don't see reason why bumping soname number - changing soname should indicate change API/ABI, which makes applications, which uses Ruby, non-functional, so these application needs to be rebuild with new Ruby). Or there weren't such changes between teeny releases in the past like they are now between 2.4.0 and 2.4.1, and 2.4.1 and 2.4.2?
2. provide config key for rbconfig method, which returns exactly SONAME (for current release libruby.so.2.4) - something like LIBRUBY_SO_NAME

#9 - 10/12/2017 12:06 PM - nobu (Nobuyoshi Nakada)

zdohnal (Zdenek Dohnal) wrote:

> If LIBRUBY_SO really doesn't provide SONAME, the issue is there isn't any other ways how to get SONAME ("libruby.so.2.4") by rbconfig (I humbly apologize, but using "hacks" like splitting output of rbconfig[LIBRUBY_ALIAS] doesn't look like proper solution).

We have provided just LIBRUBYARG, and expected SONAME to be resolved by the linker.
That is, we haven't considered the case loading ruby by dlopen with the specific version.
So there is no dedicated variables, LIBRUBY_ALIAS is the only way right now.

> 1. stop bumping teeny number of soname like it was done in the past (I don't see reason why bumping soname number - changing soname should indicate change API/ABI, which makes applications, which uses Ruby, non-functional, so these application needs to be rebuild with new Ruby). Or there weren't such changes between teeny releases in the past like they are now between 2.4.0 and 2.4.1, and 2.4.1 and 2.4.2?

I think there were no such changes.

> 1. provide config key for rbconfig method, which returns exactly SONAME (for current release libruby.so.2.4) - something like LIBRUBY_SO_NAME

I prefer this, and have made a patch for LIBRUBY_SONAME today.
Though I took the name from -soname option, is an underscore between SO and NAME preferable?

#10 - 10/12/2017 01:10 PM - zdohnal (Zdenek Dohnal)

nobu (Nobuyoshi Nakada) wrote:

> > 1. provide config key for rbconfig method, which returns exactly SONAME (for current release libruby.so.2.4) - something like LIBRUBY_SO_NAME
>
> I prefer this, and have made a patch for LIBRUBY_SONAME today.
> Though I took the name from -soname option, is an underscore between SO and NAME preferable?

I'm fine with LIBRUBY_SONAME. Thank you, good job Nobu!

#11 - 10/12/2017 08:04 PM - jeremyevans0 (Jeremy Evans)

nobu (Nobuyoshi Nakada) wrote:

> zdohnal (Zdenek Dohnal) wrote:
>
> > 1. stop bumping teeny number of soname like it was done in the past (I don't see reason why bumping soname number - changing soname should indicate change API/ABI, which makes applications, which uses Ruby, non-functional, so these application needs to be rebuild with new Ruby). Or there weren't such changes between teeny releases in the past like they are now between 2.4.0 and 2.4.1, and 2.4.1 and 2.4.2?
>
> I think there were no such changes.

There may not have been API changes, but there were definitely ABI changes between 2.4.1 and 2.4.2. If you add a non-static function to the shared object, that's a minor bump. Any removal of a non-static function from the shared object, any change in the number, order, type, or size of arguments or change to the return type to a non-static function in the shared object, or any modification to the number, order, or size of members in a struct that is passed to or returned from a non-static function in the shared object, that is a major bump.

Between 2.4.1 and 2.4.2, there were at least the following ABI changes:

Added functions (minor bump):

```
+int rb_block_min_max_arity(int *max);
+VALUE rb_lambda_call(VALUE obj, ID mid, int argc, const VALUE *argv,
+        rb_block_call_func_t bl_proc, int min_argc, int max_argc,
+        VALUE data2);
```

Modified function to add arguments (major bump):

```
-VALUE rb_func_lambda_new(rb_block_call_func_t func, VALUE val);
+VALUE rb_func_lambda_new(rb_block_call_func_t func, VALUE val, int min_argc, int max_argc);
```

Now, you may say that external users should not be calling these functions. And that's probably true, since they aren't in a public header file (no API

change). But that doesn't change the fact that the ABI did change between teeny versions.

If you want to keep a strict stable ABI (no change to the major or minor in the shared object filename), you cannot add, remove, or modify non-static functions. This can make it infeasible or much more difficult to backport fixes, so can I understand why it would not be a high priority or even a worthwhile tradeoff. All a strict stable ABI gives you is:

1. any code linked against an earlier version will work with a later version (no major bumps, minor bumps are OK)
2. any code linked against a later version will work with an earlier version (no major or minor bumps)

Even if you are just concerned about functions that appear in public header files, there have been functions added to them in teeny versions, such as "VALUE rb_big_hash(VALUE)" added to ruby/intern.h in 2.3.2 (it switched from being static to being non-static in bignum.c at the same time).

#12 - 10/13/2017 01:49 AM - nobu (Nobuyoshi Nakada)

jeremyevans0 (Jeremy Evans) wrote:

> Added functions (minor bump):
>
> ```
> +int rb_block_min_max_arity(int *max);
> +VALUE rb_lambda_call(VALUE obj, ID mid, int argc, const VALUE *argv,
> +        rb_block_call_func_t bl_proc, int min_argc, int max_argc,
> +        VALUE data2);
> ```
>
> Modified function to add arguments (major bump):
>
> ```
> -VALUE rb_func_lambda_new(rb_block_call_func_t func, VALUE val);
> +VALUE rb_func_lambda_new(rb_block_call_func_t func, VALUE val, int min_argc, int max_argc);
> ```
>
> Now, you may say that external users should not be calling these functions. And that's probably true, since they aren't in a public header file (no API change). But that doesn't change the fact that the ABI did change between teeny versions.

And they are stripped from libruby.so, on most platforms (gcc 4+, and Windows).

> Even if you are just concerned about functions that appear in public header files, there have been functions added to them in teeny versions, such as "VALUE rb_big_hash(VALUE)" added to ruby/intern.h in 2.3.2 (it switched from being static to being non-static in bignum.c at the same time).

I haven't noticed it, seems r54178 has not been backported.

#13 - 10/13/2017 02:11 AM - nobu (Nobuyoshi Nakada)

nobu (Nobuyoshi Nakada) wrote:

> And they are stripped from libruby.so, on most platforms (gcc 4+, and Windows).

Correction: they are not stripped on Windows.

#14 - 10/13/2017 02:52 AM - jeremyevans0 (Jeremy Evans)

nobu (Nobuyoshi Nakada) wrote:

> jeremyevans0 (Jeremy Evans) wrote:
>
> > Added functions (minor bump):
> >
> > ```
> > +int rb_block_min_max_arity(int *max);
> > +VALUE rb_lambda_call(VALUE obj, ID mid, int argc, const VALUE *argv,
> > +        rb_block_call_func_t bl_proc, int min_argc, int max_argc,
> > +        VALUE data2);
> > ```
> >
> > Modified function to add arguments (major bump):
> >
> > ```
> > -VALUE rb_func_lambda_new(rb_block_call_func_t func, VALUE val);
> > +VALUE rb_func_lambda_new(rb_block_call_func_t func, VALUE val, int min_argc, int max_argc);
> > ```
> >
> > Now, you may say that external users should not be calling these functions. And that's probably true, since they aren't in a public header file (no API change). But that doesn't change the fact that the ABI did change between teeny versions.
>
> And they are stripped from libruby.so, on most platforms (gcc 4+, and Windows).

Sorry for spreading misinformation. I see now where RUBY_SYMBOL_EXPORT_BEGIN and RUBY_SYMBOL_EXPORT_END are used in internal.h for symbol visibility, and also the use of objdump in Makefile.in. It appears that the GCC visibility pragma is also supported by clang.

> Even if you are just concerned about functions that appear in public header files, there have been functions added to them in teeny versions, such as "VALUE rb_big_hash(VALUE)" added to ruby/intern.h in 2.3.2 (it switched from being static to being non-static in bignum.c at the same time).

> I haven't noticed it, seems r54178 has not been backported.

Backporting it would make things worse, as it would turn a minor bump (adding visible symbol) into a major bump (removing visible symbol).

#15 - 10/13/2017 07:50 AM - zdohnal (Zdenek Dohnal)

jeremyevans0 (Jeremy Evans) wrote:

> nobu (Nobuyoshi Nakada) wrote:
>
> > zdohnal (Zdenek Dohnal) wrote:
> >
> > 1. stop bumping teeny number of soname like it was done in the past (I don't see reason why bumping soname number - changing soname should indicate change API/ABI, which makes applications, which uses Ruby, non-functional, so these application needs to be rebuild with new Ruby). Or there weren't such changes between teeny releases in the past like they are now between 2.4.0 and 2.4.1, and 2.4.1 and 2.4.2?
> >
> > I think there were no such changes.
>
> There may not have been API changes, but there were definitely ABI changes between 2.4.1 and 2.4.2. If you add a non-static function to the shared object, that's a minor bump. Any removal of a non-static function from the shared object, any change in the number, order, type, or size of arguments or change to the return type to a non-static function in the shared object, or any modification to the number, order, or size of members in a struct that is passed to or returned from a non-static function in the shared object, that is a major bump.
>
> Between 2.4.1 and 2.4.2, there were at least the following ABI changes:
>
> Added functions (minor bump):
>
> ```
> +int rb_block_min_max_arity(int *max);
> +VALUE rb_lambda_call(VALUE obj, ID mid, int argc, const VALUE *argv,
> +        rb_block_call_func_t bl_proc, int min_argc, int max_argc,
> +        VALUE data2);
> ```
>
> Modified function to add arguments (major bump):
>
> ```
> -VALUE rb_func_lambda_new(rb_block_call_func_t func, VALUE val);
> +VALUE rb_func_lambda_new(rb_block_call_func_t func, VALUE val, int min_argc, int max_argc);
> ```

So similar changes weren't in previous teeny releases like 2.3.0., 2.3.1 etc., so last number in library name didn't change and stayed 0?

> Now, you may say that external users should not be calling these functions. And that's probably true, since they aren't in a public header file (no API change). But that doesn't change the fact that the ABI did change between teeny versions.
>
> If you want to keep a strict stable ABI (no change to the major or minor in the shared object filename), you cannot add, remove, or modify non-static functions. This can make it infeasible or much more difficult to backport fixes, so can I understand why it would not be a high priority or even a worthwhile tradeoff. All a strict stable ABI gives you is:
>
> 1. any code linked against an earlier version will work with a later version (no major bumps, minor bumps are OK)
> 2. any code linked against a later version will work with an earlier version (no major or minor bumps)
>
> Even if you are just concerned about functions that appear in public header files, there have been functions added to them in teeny versions, such as "VALUE rb_big_hash(VALUE)" added to ruby/intern.h in 2.3.2 (it switched from being static to being non-static in bignum.c at the same time).

What I would like Ruby to achieve is consistency - why have you started bumping last number in library name, when it wasn't done in the past? Was there an issue which raised it? If it wasn't, IMHO it is no need for change without reason, especially when it can break something, and stay with 'not bumping last number of library name between teeny releases'.

#16 - 10/14/2017 03:35 PM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Closed

Applied in changeset trunk|r60180.

---

configure.ac: LIBRUBY_SONAME

- configure.ac (LIBRUBY_SONAME): add new variable for the name of the library name with compatibility version.
  [ruby-core:83208] [Bug #14002]

#17 - 10/14/2017 03:38 PM - nobu (Nobuyoshi Nakada)

And created a gem to update RbConfig.
https://github.com/nobu/rbconfig-update