Ruby master - Bug #5193

ruby_thread_data_type linker errors fixed with RUBY_EXTERN

08/16/2011 03:51 PM - cfis (Charlie Savage)

Status:	Third Party's Issue		
Priority:	Normal		
Assignee:	nobu (Nobuyoshi Nakada)		
Target version:			
ruby -v:		**Backport:**	

Description

ruby-debug-base19x fails to compile with VC++ 2010 with this error:

ruby_debug.obj : error LNK2001: unresolved external symbol _ruby_thread_data_type

ruby_thread_data_type is declared in vm_core.h as:

extern const rb_data_type_t ruby_thread_data_type;

ruby-debug calls GetThreadPtr which is defined as:

#define GetThreadPtr(obj, ptr) \
TypedData_Get_Struct((obj), rb_thread_t, &ruby_thread_data_type, (ptr))

What makes this interesting is that ruby_thread_data_type is properly included in the msvcr100-ruby191.def file and is exported from msvcr100-ruby191.dll. Somehow the &ruby_thread_data_type causes problems for VC++ (mingw works correctly).

Some Ubuntu users seem to be having a similar issue. See:

http://rubyforge.org/tracker/index.php?func=detail&aid=29222&group_id=8883&atid=34290

The problem is resolved for VC++ by changing the definition to:

RUBY_EXTERN const rb_data_type_t ruby_thread_data_type;

Any chance this change could be made?

Thanks - Charlie

Related issues:

Related to Backport193 - Backport #5844: Can't install ruby-debug-base19	**Rejected**	**01/05/2012**

Associated revisions

Revision 3be8a6a9 - 09/17/2011 01:21 PM - nobu (Nobuyoshi Nakada)

- vm.c (rb_vm_make_env_object, rb_vm_get_sourceline): export as a workaround for ruby-debug19 for the time being. [ruby-core:38972] [Bug #5193]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_1_9_3@33289 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 08/16/2011 04:53 PM - usa (Usaku NAKAMURA)

- Status changed from Open to Assigned

- Assignee set to nobu (Nobuyoshi Nakada)

nobu changed the name of this variable at r30783, and I guess that
he has intended to make this variable public.

nobu, is this correct?
and do you have any opinion about this request?

#2 - 08/16/2011 05:53 PM - nobu (Nobuyoshi Nakada)

- ruby -v changed from ruby 1.9.3dev (2011-08-10 revision 32904) [i386-mswin32_100] to -

Hi,

At Tue, 16 Aug 2011 16:53:07 +0900,
Usaku NAKAMURA wrote in [ruby-core:38973]:

> nobu changed the name of this variable at r30783, and I guess that
> he has intended to make this variable public.

No, it was a follow-up to r30781.

> nobu, is this correct?
> and do you have any opinion about this request?

Even if the variable were public, rb_thread_t is internal use
only, and nothing is guaranteed about it. I think debugger-API
proposal has been awaited for a while, but nothing seems to
progress.

--
Nobu Nakada

#3 - 08/17/2011 02:48 PM - cfis (Charlie Savage)

rb_thread_t is *not* public. ruby-debug-base includes vm_core.h directly to get access to rb_thread_t. So I would guess the developers are well aware that it could change at any time. But given the alternative of no debugger at all, that seems like a good risk (note that ruby-debug is the ruby debugger for RubyMine, Netbeans, Eclipse/Aptana and probably any other IDE that is used for Ruby).

So back to the original request. ruby_thread_data_type is already marked as extern and is already publicly exported via the *.def file generated when compiling ruby from source. But even with this, it doesn't work on VC++ unless the extern is changed to RUBY_EXTERN (because RUBY_EXTERN becomes __declpec(dllimport) when used by a client).

Thanks,

Charlie

#4 - 08/18/2011 03:53 PM - nobu (Nobuyoshi Nakada)

Hi,

At Wed, 17 Aug 2011 14:48:59 +0900,
Charlie Savage wrote in [ruby-core:39003]:

> rb_thread_t is *not* public. ruby-debug-base includes
> vm_core.h directly to get access to rb_thread_t. So I would
> guess the developers are well aware that it could change at
> any time. But given the alternative of no debugger at all,
> that seems like a good risk (note that ruby-debug is the ruby
> debugger for RubyMine, Netbeans, Eclipse/Aptana and probably
> any other IDE that is used for Ruby).

Then if ruby-debug can work, do those work all?

But ruby-debug code seems quite outdated; the check for
rb_method_entry_t in extconf.rb doesn't work since 2 years ago.

> So back to the original request. ruby_thread_data_type is
> already marked as extern and is already publicly exported via
> the *.def file generated when compiling ruby from source.
> But even with this, it doesn't work on VC++ unless the extern
> is changed to RUBY_EXTERN (because RUBY_EXTERN becomes
> __declpec(dllimport) when used by a client).

It was accidentally left exported on only mswin version.
Cygwin and mingw versions don't export it, and fixed now.

--
Nobu Nakada

#5 - 08/18/2011 07:17 PM - nobu (Nobuyoshi Nakada)

- Status changed from Assigned to Third Party's Issue

I'd sent workarounds to ruby-debug.

#6 - 08/20/2011 05:59 PM - cfis (Charlie Savage)

Commit dea63e4ba22d354984ce0e9e4395f8aba7152ed3 breaks ruby-debug-base19 (not the old ruby-debug-base). ruby-debug-base19 is here:

https://github.com/JetBrains/ruby-debug-base19

The result is this breaks the debugger in JetBrains, NetBeans, etc when running 1.9.3. This seems like a *huge* step backwards and a showstopper for a 1.9.3 release.

Can this please be reverted or an alternate solution provided that will fix the problem?

Thanks - Charlie

#7 - 08/20/2011 06:07 PM - cfis (Charlie Savage)

- File msvcrt-ruby191-exports.txt added

MinGW built versions of ruby *did* export ruby_thread_data_type before the last commit.

$ ruby -v
ruby 1.9.3dev (2011-08-10 revision 32901) [i386-mingw32]

$ pexports /usr/local/ruby193/bin/msvcrt-ruby191.dll > msvcrt-ruby191-exports.txt

Output is attached - see line 1632.

Thanks,

Charlie

#8 - 08/30/2011 03:24 PM - cfis (Charlie Savage)

Just wanted to follow up on this. Can we do something about this for the 1.9.3 release?

Thanks - Charlie

#9 - 09/03/2011 10:01 PM - nobu (Nobuyoshi Nakada)

Doesn't https://github.com/mark-moseley/ruby-debug/pull/14 work?

#10 - 09/09/2011 11:17 AM - cfis (Charlie Savage)

Hi Nobu,

Thanks for sending the pull request.

It seems Mark is still having trouble though. Do you mind helping him out? See:

https://github.com/mark-moseley/ruby-debug/pull/14#issuecomment-2041066

Thank you.

Charlie

#11 - 09/14/2011 08:25 AM - cfis (Charlie Savage)

Bringing this one up again - can we get resolution for the 1.9.3 release?

Thanks - Charlie

#12 - 09/14/2011 10:36 AM - kosaki (Motohiro KOSAKI)

At first, I have to put a disclaimer. The following comment is only my personal opinion, but not commiters consensus.

Symbol exposing doesn't have any regression risk. So, it can be commited into 193 if nobu agreed. But, I strongly hope to add explicitly disclaimer comments. We may stop export rb_method_entry() when debugger issue has been resoleved more cleaner way. I don't recommend other gems use it.

Nobu, what do you think?

Thank you.

#13 - 09/15/2011 03:29 PM - nobu (Nobuyoshi Nakada)

Hi,

At Wed, 14 Sep 2011 10:36:09 +0900,
Motohiro KOSAKI wrote in [ruby-core:39536]:

> Symbol exposing doesn't have any regression risk. So, it can
> be commited into 193 if nobu agreed. But, I strongly hope to
> add explicitly disclaimer comments. We may stop export
> rb_method_entry() when debugger issue has been resoleved more
> cleaner way. I don't recommend other gems use it.

rb_method_entry() doesn't seem necessary for me. The uses of
it are:

- extracting cfuncs from defined methods, ObjectSpace._id2ref and Thread#alive?, and
- checking the current method is bound at RUBY_EVENT_C_RETURN.

The reason for _id2ref is that threads are stored in debug
contexts as object IDs, and reversed when accessing with
checking if it's alive at same time. I have no idea why such
roundabout way is used, and guess this could be simpler.

For the latter use, I suspect rb_method_boundp() would be
better than rb_method_entry().

> Nobu, what do you think?

Current state is https://github.com/nobu/ruby-debug/commit/4453c34537

There are still rb_vm_make_env_object() and
rb_vm_get_sourceline(). Seems the only workaround is to export
these functions.

```
diff --git a/vm.c b/vm.c
index 6e2e9a4..18cc81d 100644
--- a/vm.c
+++ b/vm.c
@@ -50,6 +50,15 @@ void vm_analysis_operand(int insn, int n, VALUE op);
void vm_analysis_register(int reg, int isset);
void vm_analysis_insn(int insn);

+/*
```

- * TODO: replace with better interface at the next patch release.
- *
- * these functions are exported just as a workaround for ruby-debug
- * for the time being.
- */ +RUBY_FUNC_EXPORTED VALUE rb_vm_make_env_object(rb_thread_t *th, rb_control_frame_t *cfp); +RUBY_FUNC_EXPORTED int rb_vm_get_sourceline(const rb_control_frame_t *cfp); + void rb_vm_change_state(void) {

--
Nobu Nakada

#14 - 09/16/2011 02:03 AM - cfis (Charlie Savage)

Thanks Nobu for looking at this.

> The reason for _id2ref is that threads are stored in
> debug contexts as object IDs, and reversed when accessing
> with checking if it's alive at same time. I have no
> idea why such roundabout way is used, and guess this
> could be simpler.

Could this be to avoid holding a reference to a thread object, and thus not allowing it to be GCed?

> Current state is https://github.com/nobu/ruby-debug/commit/4453c34537

Great, thanks. I have also taken your changes, except the id2ref ones, and have applied them to:

https://github.com/JetBrains/ruby-debug-base19/blob/master/ruby-debug-base19x.gemspec

With the changes, that version of ruby-debug now compiles and works on 1.9.3.

If rb_vm_make_env_object() and rb_vm_get_sourceline() are exported, would that solve the need for compiling against vm_core.h?

#15 - 09/16/2011 10:23 AM - nobu (Nobuyoshi Nakada)

Hi,

At Fri, 16 Sep 2011 02:03:39 +0900,
Charlie Savage wrote in [ruby-core:39564]:

> > The reason for _id2ref is that threads are stored in
> > debug contexts as object IDs, and reversed when accessing
> > with checking if it's alive at same time. I have no
> > idea why such roundabout way is used, and guess this
> > could be simpler.
>
> Could this be to avoid holding a reference to a thread
> object, and thus not allowing it to be GCed?

No, it keeps thread objects. But I'm not sure if allowing GC
is necessary. Seems a thread object will be removed from the
table by check_thread_contexts() if the thread has terminated.
Isn't it enough?

If it is really critical, you would be possible to remove dead
threads and mark living ones only.

> If rb_vm_make_env_object() and rb_vm_get_sourceline() are
> exported, would that solve the need for compiling against
> vm_core.h?

I'm uncertain of "the need". You mean that you won't need the
header to compile ruby-debug?

--
Nobu Nakada

#16 - 09/16/2011 11:32 AM - cfis (Charlie Savage)

> No, it keeps thread objects. But I'm not sure if allowing
> GC is necessary. Seems a thread object will be removed
> from the table by check_thread_contexts() if the thread has
> terminated. Isn't it enough?

Since I don't know the code, I don't the answer. Mark would be best to answer that.

> I'm uncertain of "the need". You mean that you won't
> need the header to compile ruby-debug?

Yes. To compile ruby-debug you must have the Ruby source code installed because it refers to the headers iseq.h and vm_core.h neither of which are installed by Ruby.

It would be much better for ruby-debug to not have to use those headers and instead could just use Ruby's public api. Don't know if that is currently possible. If not, seems like a nice addition for 1.9.4 (I know you have mentioned work on an official debug api).

#17 - 09/16/2011 01:53 PM - nobu (Nobuyoshi Nakada)

Hi,

At Fri, 16 Sep 2011 11:32:04 +0900,
Charlie Savage wrote in [ruby-core:39570]:

> No, it keeps thread objects. But I'm not sure if allowing GC is necessary. Seems a thread object will be removed from the table by check_thread_contexts() if the thread has terminated. Isn't it enough?

Since I don't know the code, I don't the answer. Mark would be best to answer that.

You don't need to use object_id in C, just don't mark it instead. I updated my clone.

> I'm uncertain of "the need". You mean that you won't need the header to compile ruby-debug?

Yes. To compile ruby-debug you must have the Ruby source code installed because it refers to the headers iseq.h and vm_core.h neither of which are installed by Ruby.

It wouldn't be possible since ruby-debug depens on the ruby internal structures.

> It would be much better for ruby-debug to not have to use those headers and instead could just use Ruby's public api. Don't know if that is currently possible. If not, seems like a nice addition for 1.9.4 (I know you have mentioned work on an official debug api).

It's much more preferable way, of course. So we had requested the API proposal and its use cases long before, but no response till the last month and just "it can't build".

--
Nobu Nakada

#18 - 09/17/2011 10:44 AM - cfis (Charlie Savage)

> So we had requested the API proposal and its use cases long before, but no response till the last month and just "it can't build".

First, is there anything else we need to do for ruby 1.9.3? The current status is that both ruby-debug-base19 and ruby-debug-base19x (JetBrain's fork) both compile now. ruby-debug-base19x works for me, but ruby-debug-base19 does not due to segmentation faults when evaluating expressions. I assume that is due to an issue in ruby-debug-base19 and not ruby. So my original goal for this ticket has been met, thank you.

Second:

> We had requested the API proposal and its use cases long before, but no response till the last month and just "it can't build".

Who is in charge of this? Is ruby-core supposed to propose an API? If so, who? Or is it the ruby-debug developers (Mark and JetBrains)?

Remember I'm just a user of ruby-debug who was alarmed when it stopped working last month. Having said that, it is important enough to me that I am happy to help with whatever is needed.

But instead of discussing that here, should a new ticket be opened for a creating an official debugger api for 1.9.4?

#19 - 09/17/2011 12:23 PM - ko1 (Koichi Sasada)

Hi,

(11/09/16 18:44), Charlie Savage wrote:

> But instead of discussing that here, should a new ticket be opened for a creating an official debugger api for 1.9.4?

I (and maybe nobu) will try them. Actually, I've needed to implement
it. However, because of my laziness, I've postponed it.
Sorry about that. Maybe I'll make it at least the end of Apr/2012.

(I'm not sure which version contains this feature)

Thanks,
Koichi

#20 - 10/09/2011 11:49 AM - mark-moseley (Mark Moseley)

I've merged nobu's changes, and they're at rubyforge (http://rubyforge.org/frs/?group_id=8883)

I'm getting a segmentation fault running the "where" command in ruby-debug19. I've traced it to this ruby code:

id = context_frame_method(pos)
call_str << id.id2name

The first line calls into ruby_debug.c, method context_frame_id, which was changed.
The segfault is on the second line; CFUNC :id2name is at the top of the control frame dump.

The purpose of this section (from frame.rb) was to get the method name of the given frame for the stack trace.

This code worked fine with 1.9.2.

#21 - 11/04/2011 12:56 AM - mark-moseley (Mark Moseley)

Any chance this will get looked at? I can't release ruby-debug19 for 1.9.3 until it's fixed.

#22 - 11/04/2011 05:57 AM - jrochkind (jonathan rochkind)

There is lots of interest in having a ruby-debug that works for 1.9.3. Lots of people depend upon it. This solution is floating around the net:

http://blog.wyeworks.com/2011/11/1/ruby-1-9-3-and-ruby-debug

People say that's working, I'm not sure what prevents that from turning into an actual tagged release of ruby-debug19, rather than requiring building of unreleased gem sources. But one way or another, lots of developers are hurting from loss of ruby-debug.

#23 - 02/08/2012 10:14 PM - janvarwig (Jan Varwig)

Are there any news on this issue?

Manually installing the unreleased gems via somthing like

```
curl -LO http://rubyforge.org/frs/download.php/75414/linecache19-0.5.13.gem
curl -LO http://rubyforge.org/frs/download.php/75415/ruby-debug-base19-0.11.26.gem

gem install linecache19-0.5.13.gem
gem install ruby-debug-base19-0.11.26.gem -- --with-ruby-include=$HOME/.rvm/rubies/ruby-1.9.3-p0-falcon/includ
e/ruby-1.9.1/ruby-1.9.3-p0/

rm linecache19-0.5.13.gem
rm ruby-debug-base19-0.11.26.gem
```

seems to work perfectly for me and others. What's holding this back?

The current situation makes it really painful to work with bundler and making the jump from 1.8.7 to 1.9.x
is out of the question within my company (and I'm sure in other places too) as long as the debugger isn't easily available.

#24 - 02/11/2012 01:36 AM - mark-moseley (Mark Moseley)

What's holding this back is the segmentation fault running the "where" command.

#25 - 02/11/2012 01:58 AM - naruse (Yui NARUSE)

- Status changed from Third Party's Issue to Feedback

Could you summarize current situation?

I want to release Ruby 1.9.3 patch release and know which commits should be backported.
I know 1.9.3 hides internal functions and it breaks ruby-debug, so the workaround (r33289) is committed to trunk.
Only I need to do is backport r33289 to 1.9.3?

#26 - 02/11/2012 05:18 AM - naruse (Yui NARUSE)

- Tracker changed from Bug to Backport

- Project changed from Ruby master to Backport193

- Category deleted (core)

- Target version deleted (1.9.3)

#27 - 02/11/2012 05:46 AM - nobu (Nobuyoshi Nakada)

Yui NARUSE wrote:

> I know 1.9.3 hides internal functions and it breaks ruby-debug, so the workaround (r33289) is committed to trunk.
> Only I need to do is backport r33289 to 1.9.3?

No, the workaround is to *1.9.3*, not trunk.

#28 - 02/12/2012 09:45 AM - naruse (Yui NARUSE)

- Tracker changed from Backport to Bug

- Project changed from Backport193 to Ruby master

- Status changed from Feedback to Third Party's Issue

Ah, I see, thanks nobu.

Other issues reported to this tickets are considered Third Party's Issue.

Files

msvcrt-ruby191-exports.txt	30.7 KB	08/20/2011	cfis (Charlie Savage)