Ruby - Bug #14816

Extension build failure on a system with musl libc

06/03/2018 02:06 PM - akamch (Anatoly Kamchatnov)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.6.0preview2 (2018-05-31 trunk 63539) [x86_64-linux]	**Backport:**	2.3: REQUIRED, 2.4: DONE, 2.5: DONE

Description

Some extensions fail to build on a Linux with musl (Void Linux). Build of unf_ext is an example.
isinf() and isnan() are defined as macros in musl: https://git.musl-libc.org/cgit/musl/tree/include/math.h
https://github.com/gliderlabs/docker-alpine/issues/261 "Cannot build native extensions for unf_ext gem" seems to be a related issue.

```
$ gem install unf_ext

Building native extensions. This could take a while...
ERROR:  Error installing unf_ext:
        ERROR: Failed to build gem native extension.

    current directory: /home/rev/.gem/ruby/2.6.0/gems/unf_ext-0.0.7.5/ext/unf_ext
/home/rev/.rbenv/versions/2.6.0-preview2/bin/ruby -r ./siteconf20180603-29655-wgzu56.rb extconf.rb
checking for -lstdc++... yes
creating Makefile

current directory: /home/rev/.gem/ruby/2.6.0/gems/unf_ext-0.0.7.5/ext/unf_ext
make "DESTDIR=" clean

current directory: /home/rev/.gem/ruby/2.6.0/gems/unf_ext-0.0.7.5/ext/unf_ext
make "DESTDIR="
compiling unf.cc
cc1plus: warning: command line option '-Wimplicit-int' is valid for C/ObjC but not for C++
cc1plus: warning: command line option '-Wdeclaration-after-statement' is valid for C/ObjC but not
for C++
cc1plus: warning: command line option '-Wimplicit-function-declaration' is valid for C/ObjC but no
t for C++
In file included from /home/rev/.rbenv/versions/2.6.0-preview2/include/ruby-2.6.0/ruby/defines.h:1
53:0,
                 from /home/rev/.rbenv/versions/2.6.0-preview2/include/ruby-2.6.0/ruby/ruby.h:29,
                 from /home/rev/.rbenv/versions/2.6.0-preview2/include/ruby-2.6.0/ruby.h:33,
                 from unf.cc:3:
/home/rev/.rbenv/versions/2.6.0-preview2/include/ruby-2.6.0/ruby/missing.h:172:29: error: 'int isi
nf(double)' conflicts with a previous declaration
 RUBY_EXTERN int isinf(double);
                             ^
In file included from /usr/include/c++/7.3/math.h:36:0,
                 from /home/rev/.rbenv/versions/2.6.0-preview2/include/ruby-2.6.0/ruby/missing.h:2
3,
                 from /home/rev/.rbenv/versions/2.6.0-preview2/include/ruby-2.6.0/ruby/defines.h:1
53,
                 from /home/rev/.rbenv/versions/2.6.0-preview2/include/ruby-2.6.0/ruby/ruby.h:29,
                 from /home/rev/.rbenv/versions/2.6.0-preview2/include/ruby-2.6.0/ruby.h:33,
                 from unf.cc:3:
/usr/include/c++/7.3/cmath:599:3: note: previous declaration 'constexpr bool std::isinf(double)'
   isinf(double __x)
   ^~~~~
In file included from /home/rev/.rbenv/versions/2.6.0-preview2/include/ruby-2.6.0/ruby/defines.h:1
53:0,
                 from /home/rev/.rbenv/versions/2.6.0-preview2/include/ruby-2.6.0/ruby/ruby.h:29,
                 from /home/rev/.rbenv/versions/2.6.0-preview2/include/ruby-2.6.0/ruby.h:33,
                 from unf.cc:3:
```

```
/home/rev/.rbenv/versions/2.6.0-preview2/include/ruby-2.6.0/ruby/missing.h:179:29: error: 'int isn
an(double)' conflicts with a previous declaration
 RUBY_EXTERN int isnan(double);
                             ^
In file included from /usr/include/c++/7.3/math.h:36:0,
                 from /home/rev/.rbenv/versions/2.6.0-preview2/include/ruby-2.6.0/ruby/missing.h:2
3,
                 from /home/rev/.rbenv/versions/2.6.0-preview2/include/ruby-2.6.0/ruby/defines.h:1
53,
                 from /home/rev/.rbenv/versions/2.6.0-preview2/include/ruby-2.6.0/ruby/ruby.h:29,
                 from /home/rev/.rbenv/versions/2.6.0-preview2/include/ruby-2.6.0/ruby.h:33,
                 from unf.cc:3:
/usr/include/c++/7.3/cmath:626:3: note: previous declaration 'constexpr bool std::isnan(double)'
   isnan(double __x)
   ^~~~~
cc1plus: warning: unrecognized command line option '-Wno-cast-function-type'
cc1plus: warning: unrecognized command line option '-Wno-self-assign'
cc1plus: warning: unrecognized command line option '-Wno-constant-logical-operand'
cc1plus: warning: unrecognized command line option '-Wno-parentheses-equality'
make: *** [Makefile:211: unf.o] Error 1

make failed, exit code 2
```

Associated revisions

Revision b5d6db65b49528c45cdfa75200dc8013d332b0db - 06/05/2018 06:50 AM - shyouhei (Shyouhei Urabe)

int isnan(double) is a POSIXism

- isnan is something relatively new. We need to provide one for those systems without it. However:
- X/Open defines int isnan(double). Note the int.
- C99 defines isnan(x) to be a macro.
- C++11 nukes them all, undefines all the "masking macro"s, and defines its own bool isnan(double). Note the bool.
- In C++, int isnan(double) and bool isnan(double) are incompatible.
- So the mess.

[Bug #14816][ruby-core:87364]
further reading: https://developers.redhat.com/blog/2016/02/29/why-cstdlib-is-more-complicated-than-you-might-think/

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@63571 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision b5d6db65b49528c45cdfa75200dc8013d332b0db - 06/05/2018 06:50 AM - shyouhei (Shyouhei Urabe)

int isnan(double) is a POSIXism

- isnan is something relatively new. We need to provide one for those systems without it. However:
- X/Open defines int isnan(double). Note the int.
- C99 defines isnan(x) to be a macro.
- C++11 nukes them all, undefines all the "masking macro"s, and defines its own bool isnan(double). Note the bool.
- In C++, int isnan(double) and bool isnan(double) are incompatible.
- So the mess.

[Bug #14816][ruby-core:87364]
further reading: https://developers.redhat.com/blog/2016/02/29/why-cstdlib-is-more-complicated-than-you-might-think/

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@63571 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision b5d6db65 - 06/05/2018 06:50 AM - shyouhei (Shyouhei Urabe)

int isnan(double) is a POSIXism

- isnan is something relatively new. We need to provide one for those systems without it. However:
- X/Open defines int isnan(double). Note the int.
- C99 defines isnan(x) to be a macro.
- C++11 nukes them all, undefines all the "masking macro"s, and defines its own bool isnan(double). Note the bool.

- In C++, int isnan(double) and bool isnan(double) are incompatible.
- So the mess.

[Bug #14816][ruby-core:87364]
further reading: https://developers.redhat.com/blog/2016/02/29/why-cstdlib-is-more-complicated-than-you-might-think/

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@63571 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision b7595f2c2ec14389170808dfb36b1d99c9d0e899 - 06/05/2018 02:16 PM - ngoto (Naohisa Goto)

include/ruby/missing.h: defined(__cplusplus) before using __cplusplus

- include/ruby/missing.h (isinf, isnan): For non-C++ programs, defined(__cplusplus) may be needed before using __cplusplus. [Bug #14816]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@63572 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision b7595f2c2ec14389170808dfb36b1d99c9d0e899 - 06/05/2018 02:16 PM - ngoto (Naohisa Goto)

include/ruby/missing.h: defined(__cplusplus) before using __cplusplus

- include/ruby/missing.h (isinf, isnan): For non-C++ programs, defined(__cplusplus) may be needed before using __cplusplus. [Bug #14816]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@63572 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision b7595f2c - 06/05/2018 02:16 PM - ngoto (Naohisa Goto)

include/ruby/missing.h: defined(__cplusplus) before using __cplusplus

- include/ruby/missing.h (isinf, isnan): For non-C++ programs, defined(__cplusplus) may be needed before using __cplusplus. [Bug #14816]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@63572 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision b6ab48d8b1e90a095388598de95d9116b7f7c3a2 - 07/30/2018 01:57 PM - U.Nakamura

merge revision(s) 63571,63572: [Backport #14816]

```
    `int isnan(double)` is a POSIXism

    - `isnan` is something relatively new.  We need to provide one for
      those systems without it.  However:
    - X/Open defines `int isnan(double)`. Note the `int`.
    - C99 defines `isnan(x)` to be a macro.
    - C++11 nukes them all, undefines all the "masking macro"s, and
      defines its own `bool isnan(double)`.  Note the `bool`.
    - In C++, `int isnan(double)` and `bool isnan(double)` are
      incompatible.
    - So the mess.

    [Bug #14816][ruby-core:87364]
    further reading: https://developers.redhat.com/blog/2016/02/29/why-cstdlib-is-more-complicated-than-you-mi
ght-think/


    include/ruby/missing.h: defined(__cplusplus) before using __cplusplus

    * include/ruby/missing.h (isinf, isnan): For non-C++ programs,
      defined(__cplusplus) may be needed before using __cplusplus.
      [Bug #14816]
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_4@64126 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision b6ab48d8b1e90a095388598de95d9116b7f7c3a2 - 07/30/2018 01:57 PM - U.Nakamura

merge revision(s) 63571,63572: [Backport #14816]

```
    `int isnan(double)` is a POSIXism

    - `isnan` is something relatively new.  We need to provide one for
```

```
      those systems without it.  However:
    - X/Open defines `int isnan(double)`. Note the `int`.
    - C99 defines `isnan(x)` to be a macro.
    - C++11 nukes them all, undefines all the "masking macro"s, and
      defines its own `bool isnan(double)`.  Note the `bool`.
    - In C++, `int isnan(double)` and `bool isnan(double)` are
      incompatible.
    - So the mess.

    [Bug #14816][ruby-core:87364]
    further reading: https://developers.redhat.com/blog/2016/02/29/why-cstdlib-is-more-complicated-than-you-mi
ght-think/


    include/ruby/missing.h: defined(__cplusplus) before using __cplusplus

    * include/ruby/missing.h (isinf, isnan): For non-C++ programs,
      defined(__cplusplus) may be needed before using __cplusplus.
      [Bug #14816]
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_4@64126 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision b6ab48d8 - 07/30/2018 01:57 PM - U.Nakamura

merge revision(s) 63571,63572: [Backport #14816]

```
    `int isnan(double)` is a POSIXism

    - `isnan` is something relatively new.  We need to provide one for
      those systems without it.  However:
    - X/Open defines `int isnan(double)`. Note the `int`.
    - C99 defines `isnan(x)` to be a macro.
    - C++11 nukes them all, undefines all the "masking macro"s, and
      defines its own `bool isnan(double)`.  Note the `bool`.
    - In C++, `int isnan(double)` and `bool isnan(double)` are
      incompatible.
    - So the mess.

    [Bug #14816][ruby-core:87364]
    further reading: https://developers.redhat.com/blog/2016/02/29/why-cstdlib-is-more-complicated-than-you-mi
ght-think/


    include/ruby/missing.h: defined(__cplusplus) before using __cplusplus

    * include/ruby/missing.h (isinf, isnan): For non-C++ programs,
      defined(__cplusplus) may be needed before using __cplusplus.
      [Bug #14816]
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_4@64126 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 99d1f2c5dc02dbeac42b4d27e4798774133c618a - 08/18/2018 04:18 AM - nagachika (Tomoyuki Chikanaga)

merge revision(s) 63571,63572: [Backport #14816]

```
    `int isnan(double)` is a POSIXism

    - `isnan` is something relatively new.  We need to provide one for
      those systems without it.  However:
    - X/Open defines `int isnan(double)`. Note the `int`.
    - C99 defines `isnan(x)` to be a macro.
    - C++11 nukes them all, undefines all the "masking macro"s, and
      defines its own `bool isnan(double)`.  Note the `bool`.
    - In C++, `int isnan(double)` and `bool isnan(double)` are
      incompatible.
    - So the mess.

    [Bug #14816][ruby-core:87364]
    further reading: https://developers.redhat.com/blog/2016/02/29/why-cstdlib-is-more-complicated-than-you-mi
ght-think/


    include/ruby/missing.h: defined(__cplusplus) before using __cplusplus

    * include/ruby/missing.h (isinf, isnan): For non-C++ programs,
```

```
      defined(__cplusplus) may be needed before using __cplusplus.
      [Bug #14816]
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_5@64434 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 99d1f2c5dc02dbeac42b4d27e4798774133c618a - 08/18/2018 04:18 AM - nagachika (Tomoyuki Chikanaga)

merge revision(s) 63571,63572: [Backport #14816]

```
    `int isnan(double)` is a POSIXism

    - `isnan` is something relatively new.  We need to provide one for
      those systems without it.  However:
    - X/Open defines `int isnan(double)`. Note the `int`.
    - C99 defines `isnan(x)` to be a macro.
    - C++11 nukes them all, undefines all the "masking macro"s, and
      defines its own `bool isnan(double)`.  Note the `bool`.
    - In C++, `int isnan(double)` and `bool isnan(double)` are
      incompatible.
    - So the mess.

    [Bug #14816][ruby-core:87364]
    further reading: https://developers.redhat.com/blog/2016/02/29/why-cstdlib-is-more-complicated-than-you-mi
ght-think/


    include/ruby/missing.h: defined(__cplusplus) before using __cplusplus

    * include/ruby/missing.h (isinf, isnan): For non-C++ programs,
      defined(__cplusplus) may be needed before using __cplusplus.
      [Bug #14816]
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_5@64434 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 99d1f2c5 - 08/18/2018 04:18 AM - nagachika (Tomoyuki Chikanaga)

merge revision(s) 63571,63572: [Backport #14816]

```
    `int isnan(double)` is a POSIXism

    - `isnan` is something relatively new.  We need to provide one for
      those systems without it.  However:
    - X/Open defines `int isnan(double)`. Note the `int`.
    - C99 defines `isnan(x)` to be a macro.
    - C++11 nukes them all, undefines all the "masking macro"s, and
      defines its own `bool isnan(double)`.  Note the `bool`.
    - In C++, `int isnan(double)` and `bool isnan(double)` are
      incompatible.
    - So the mess.

    [Bug #14816][ruby-core:87364]
    further reading: https://developers.redhat.com/blog/2016/02/29/why-cstdlib-is-more-complicated-than-you-mi
ght-think/


    include/ruby/missing.h: defined(__cplusplus) before using __cplusplus

    * include/ruby/missing.h (isinf, isnan): For non-C++ programs,
      defined(__cplusplus) may be needed before using __cplusplus.
      [Bug #14816]
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_5@64434 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 06/04/2018 03:17 AM - shyouhei (Shyouhei Urabe)

akamch (Anatoly Kamchatnov) wrote:

> isinf() and isnan() are defined as macros in musl: https://git.musl-libc.org/cgit/musl/tree/include/math.h

Yes, and the corresponding missing.h lines are this:

```
#ifndef isnan
# ifndef HAVE_ISNAN
```

```
RUBY_EXTERN int isnan(double);
# endif
#endif
```

So the conflicting function *is* actually guarded by #ifdef. I guess this is not our fault?

#2 - 06/04/2018 11:37 AM - akamch (Anatoly Kamchatnov)

> I guess this is not our fault?

Not entirely. Most likely it's nobody's fault but you can always blame autoconf :) Alpine's author thinks that "the configure script does not detect isnan/isinf as macros, call ruby devs". Autoconf's doc says

```
isinf
isnan
    The C99 standard says that isinf and isnan are macros. On some systems just macros are available (e.g., HP
-UX and Solaris 10), on some systems both macros and functions (e.g., glibc 2.3.2), and on some systems only f
unctions (e.g., IRIX 6 and Solaris 9). In some cases these functions are declared in nonstandard headers like
<sunmath.h> and defined in non-default libraries like -lm or -lsunmath.

    The C99 isinf and isnan macros work correctly with long double arguments, but pre-C99 systems that use fun
ctions typically assume double arguments. On such a system, isinf incorrectly returns true for a finite long d
ouble argument that is outside the range of double.

    The best workaround for these issues is to use gnulib modules isinf and isnan (see Gnulib). But a lighter
weight solution involves code like the following.

              #include <math.h>

              #ifndef isnan
              # define isnan(x) \
                  (sizeof (x) == sizeof (long double) ? isnan_ld (x) \
                   : sizeof (x) == sizeof (double) ? isnan_d (x) \
                   : isnan_f (x))
              static inline int isnan_f  (float       x) { return x != x; }
              static inline int isnan_d  (double      x) { return x != x; }
              static inline int isnan_ld (long double x) { return x != x; }
              #endif

              #ifndef isinf
              # define isinf(x) \
                  (sizeof (x) == sizeof (long double) ? isinf_ld (x) \
                   : sizeof (x) == sizeof (double) ? isinf_d (x) \
                   : isinf_f (x))
              static inline int isinf_f  (float       x)
              { return !isnan (x) && isnan (x - x); }
              static inline int isinf_d  (double      x)
              { return !isnan (x) && isnan (x - x); }
              static inline int isinf_ld (long double x)
              { return !isnan (x) && isnan (x - x); }
              #endif
```

https://www.gnu.org/software/autoconf/manual/autoconf.html#Function-Portability

Looks like musl is not unlike HP-UX and Solaris 10 in that regard. The question is where one should fix this: some header or ruby routines around extconf.rb, gem in question, anywhere else? That I don't know, but the first option still seems plausible to me.

#3 - 06/04/2018 06:32 PM - shevegen (Robert A. Heiler)

I think ngoto is knowing quite a bit about Solaris so perhaps if he has
time he could comment.

I also understand you wanting to use Ruby even on exotic combinations
like musl + void. The original rack author also uses void. :)

#4 - 06/04/2018 07:06 PM - akamch (Anatoly Kamchatnov)

> I also understand you wanting to use Ruby even on exotic combinations like musl + void.

Indeed, yet one can also easily envision many a failed build of Ruby codebases inside the musl-only Alpine-based Docker containers. Void is almost irrelevant here.

> The original rack author also uses void. :)

And does hell of a job maintaining that fine distribution.

#5 - 06/04/2018 11:58 PM - shyouhei (Shyouhei Urabe)

akamch (Anatoly Kamchatnov) wrote:

> > I guess this is not our fault?
>
> Not entirely. Most likely it's nobody's fault but you can always blame autoconf

Highly skeptical. Can you build ruby from source and show us your config.log then?

Like I showed before the RUBY_EXTERN int isnan(double); line is effective only if (1) isnan is *not* a macro, and (2) isnan is *not* provided as a function. "The best workaround" that the autoconf says does not work on your system because in case it would, ours must also.

#6 - 06/05/2018 12:21 AM - shyouhei (Shyouhei Urabe)

Let me directly point out what is actually to be blamed:

akamch (Anatoly Kamchatnov) wrote:

> In file included from /usr/include/c++/7.3/math.h:36:0,
> from /home/rev/.rbenv/versions/2.6.0-preview2/include/ruby-2.6.0/ruby/missing.h:23,
> from /home/rev/.rbenv/versions/2.6.0-preview2/include/ruby-2.6.0/ruby/defines.h:153,
> from /home/rev/.rbenv/versions/2.6.0-preview2/include/ruby-2.6.0/ruby/ruby.h:29,
> from /home/rev/.rbenv/versions/2.6.0-preview2/include/ruby-2.6.0/ruby.h:33,
> from unf.cc:3:
> /usr/include/c++/7.3/cmath:626:3: note: previous declaration 'constexpr bool std::isnan(double)'
> isnan(double __x)
> ^~~~~

This is wrong. Your C++ header file *does* define a function named isnan. You have to include C's one (the one you referred) instead.

#7 - 06/05/2018 01:53 AM - shyouhei (Shyouhei Urabe)

Proposed workaround, not tested though.

```
From 01839b509c1bb914337124ac3d1f644b05ef90d8 Mon Sep 17 00:00:00 2001
From: "Urabe, Shyouhei" <shyouhei@ruby-lang.org>
Date: Tue, 5 Jun 2018 10:26:06 +0900
Subject: [PATCH] C++11 is so bad it introduces a nightmare.

TL;DR see https://developers.redhat.com/blog/2016/02/29/why-cstdlib-is-more-complicated-than-you-might-think/

 - `isnan` is something relatively new.  We need to provide one for
   those systems without it.  However:
 - X/Open defines `int isnan(double)`. Note the `int`.
 - C99 defines `isnan(x)` to be a macro.
 - C++11 nukes them all, undefines all the "masking macro"s, and
   define its own `bool isnan(double)`.  Note the `bool`.
 - In C++, `int isnan(double)` and `bool isnan(double)` are
   incompatible.
 - So the mess.

Signed-off-by: Urabe, Shyouhei <shyouhei@ruby-lang.org>
---
 include/ruby/missing.h | 6 ++++++
 1 file changed, 6 insertions(+)

diff --git a/include/ruby/missing.h b/include/ruby/missing.h
index dc3fd502b5..8df917498e 100644
--- a/include/ruby/missing.h
+++ b/include/ruby/missing.h
@@ -168,6 +168,8 @@ RUBY_EXTERN const union bytesequence4_or_float rb_nan;
 #    include <ieeefp.h>
 #    endif
```

```
 #  define isinf(x) (!finite(x) && !isnan(x))
+#  elsif __cplusplus >= 201103L
+#    include <cmath> // it must include constexpr bool isinf(double);
 #  else
 RUBY_EXTERN int isinf(double);
 #  endif
@@ -176,7 +178,11 @@ RUBY_EXTERN int isinf(double);

 #ifndef isnan
 # ifndef HAVE_ISNAN
+#  if __cplusplus >= 201103L
+#    include <cmath> // it must include constexpr bool isnan(double);
+#  else
 RUBY_EXTERN int isnan(double);
+#  endif
 # endif
 #endif

--
2.17.1
```

#8 - 06/05/2018 05:00 AM - akamch (Anatoly Kamchatnov)

shyouhei (Shyouhei Urabe) wrote:

> Let me directly point out what is actually to be blamed

Works best for me! Thank you very much for getting to the root of the problem, there's much to reflect upon.

The patch does away with the isnan error, but the other one still remains:

```
compiling unf.cc
cc1plus: warning: command line option '-Wimplicit-int' is valid for C/ObjC but not for C++
cc1plus: warning: command line option '-Wdeclaration-after-statement' is valid for C/ObjC but not for C++
cc1plus: warning: command line option '-Wimplicit-function-declaration' is valid for C/ObjC but not for C++
In file included from /home/rev/.rbenv/versions/2.6.0-preview2/include/ruby-2.6.0/ruby/defines.h:153:0,
                 from /home/rev/.rbenv/versions/2.6.0-preview2/include/ruby-2.6.0/ruby/ruby.h:29,
                 from /home/rev/.rbenv/versions/2.6.0-preview2/include/ruby-2.6.0/ruby.h:33,
                 from unf.cc:3:
/home/rev/.rbenv/versions/2.6.0-preview2/include/ruby-2.6.0/ruby/missing.h:174:29: error: 'int isinf(double)'
conflicts with a previous declaration
 RUBY_EXTERN int isinf(double);
                             ^
In file included from /usr/include/c++/7.3/math.h:36:0,
                 from /home/rev/.rbenv/versions/2.6.0-preview2/include/ruby-2.6.0/ruby/missing.h:23,
                 from /home/rev/.rbenv/versions/2.6.0-preview2/include/ruby-2.6.0/ruby/defines.h:153,
                 from /home/rev/.rbenv/versions/2.6.0-preview2/include/ruby-2.6.0/ruby/ruby.h:29,
                 from /home/rev/.rbenv/versions/2.6.0-preview2/include/ruby-2.6.0/ruby.h:33,
                 from unf.cc:3:
/usr/include/c++/7.3/cmath:599:3: note: previous declaration 'constexpr bool std::isinf(double)'
   isinf(double __x)
   ^~~~~
cc1plus: warning: unrecognized command line option '-Wno-cast-function-type'
cc1plus: warning: unrecognized command line option '-Wno-self-assign'
cc1plus: warning: unrecognized command line option '-Wno-constant-logical-operand'
cc1plus: warning: unrecognized command line option '-Wno-parentheses-equality'
make: *** [Makefile:211: unf.o] Error 1

make failed, exit code 2
```

#9 - 06/05/2018 06:16 AM - shyouhei (Shyouhei Urabe)

akamch (Anatoly Kamchatnov) wrote:

> The patch does away with the isnan error, but the other one still remains:

Ah sorry. Embarrassing typo :(. Try applying this patch over the previous one.

```
diff --git a/include/ruby/missing.h b/include/ruby/missing.h
index 8df917498e..2d3852c131 100644
--- a/include/ruby/missing.h
+++ b/include/ruby/missing.h
```

```
@@ -168,7 +168,7 @@ RUBY_EXTERN const union bytesequence4_or_float rb_nan;
 #    include <ieeefp.h>
 #    endif
 #  define isinf(x) (!finite(x) && !isnan(x))
-#  elsif __cplusplus >= 201103L
+#  elif __cplusplus >= 201103L
 #    include <cmath> // it must include constexpr bool isinf(double);
 #  else
 RUBY_EXTERN int isinf(double);
```

#10 - 06/05/2018 06:27 AM - akamch (Anatoly Kamchatnov)

Works great, many thanks, Shyouhei!

The wider question on what could/should have been done to avoid these ifdef dances is definitely not Ruby's to answer.

#11 - 06/05/2018 06:51 AM - shyouhei (Shyouhei Urabe)

- Status changed from Open to Closed

Applied in changeset trunk|r63571.

---

int isnan(double) is a POSIXism

- isnan is something relatively new. We need to provide one for those systems without it. However:
- X/Open defines int isnan(double). Note the int.
- C99 defines isnan(x) to be a macro.
- C++11 nukes them all, undefines all the "masking macro"s, and defines its own bool isnan(double). Note the bool.
- In C++, int isnan(double) and bool isnan(double) are incompatible.
- So the mess.

[Bug #14816][ruby-core:87364]
further reading: https://developers.redhat.com/blog/2016/02/29/why-cstdlib-is-more-complicated-than-you-might-think/

#12 - 06/05/2018 06:52 AM - shyouhei (Shyouhei Urabe)

- Backport changed from 2.3: UNKNOWN, 2.4: UNKNOWN, 2.5: UNKNOWN to 2.3: REQUIRED, 2.4: REQUIRED, 2.5: REQUIRED

#13 - 07/30/2018 01:57 PM - usa (Usaku NAKAMURA)

- Backport changed from 2.3: REQUIRED, 2.4: REQUIRED, 2.5: REQUIRED to 2.3: REQUIRED, 2.4: DONE, 2.5: REQUIRED

ruby_2_4 r64126 merged revision(s) 63571,63572.

#14 - 08/18/2018 04:18 AM - nagachika (Tomoyuki Chikanaga)

- Backport changed from 2.3: REQUIRED, 2.4: DONE, 2.5: REQUIRED to 2.3: REQUIRED, 2.4: DONE, 2.5: DONE

ruby_2_5 r64434 merged revision(s) 63571,63572.