

11) Failure:

test_inspect(TestBigDecimal) [C:/msys/1.0/home/beta/ruby-1.9.3-p194/test/bigdecimal/test_bigdecimal.rb:811]:
Expected /#\$/ to match "#".

12) Error:

test_marshall(TestBigDecimal):

TypeError: load failed: invalid character in the marshaled string

C:/msys/1.0/home/beta/ruby-1.9.3-p194/test/bigdecimal/test_bigdecimal.rb:468:in _load'

C:/msys/1.0/home/beta/ruby-1.9.3-p194/test/bigdecimal/test_bigdecimal.rb:468:inload'

C:/msys/1.0/home/beta/ruby-1.9.3-p194/test/bigdecimal/test_bigdecimal.rb:468:in `test_marshall'

13) Failure:

test_power_of_three(TestBigDecimal) [C:/msys/1.0/home/beta/ruby-1.9.3-p194/test/bigdecimal/test_bigdecimal.rb:898]:

Expected 0.012345679012345678 - 0.12345679012345679012345679012345679012345679012345679012345679Ezd
(0.1111111111111111) to be < 0.001.

14) Failure:

test_to_f(TestBigDecimal) [C:/msys/1.0/home/beta/ruby-1.9.3-p194/test/bigdecimal/test_bigdecimal.rb:527]:

expected but was

.

15) Failure:

test_to_s(TestBigDecimal) [C:/msys/1.0/home/beta/ruby-1.9.3-p194/test/bigdecimal/test_bigdecimal.rb:786]:

<"0.1234567890123456789E3"> expected but was

<"0.1234567890123456789Ezd">.

16) Failure:

test_atan(TestBigMath) [C:/msys/1.0/home/beta/ruby-1.9.3-p194/test/bigdecimal/test_bigmath.rb:59]:

Expected 1.5707963267948966 - 0.15707963267948966192Ezd (1.413716694115407) to be < 0.001.

17) Failure:

test_const(TestBigMath) [C:/msys/1.0/home/beta/ruby-1.9.3-p194/test/bigdecimal/test_bigmath.rb:13]:

Expected 3.141592653589793 - 0.3141592653589793238462643383279502883919859293521427Ezd (2.827433388230814) to be
< 0.001.

18) Failure:

test_cos(TestBigMath) [C:/msys/1.0/home/beta/ruby-1.9.3-p194/test/bigdecimal/test_bigmath.rb:42]:

Expected 1.0 - 0.1Ezd (0.9) to be < 0.001.

History

#1 - 05/07/2012 06:51 AM - sorah (Sorah Fukumori)

- Assignee set to mrkn (Kenta Murata)

#2 - 05/08/2012 11:18 PM - phasis68 (Heesob Park)

This issue is related with sprintf "%zd" format which is a C99 feature.

To enable sprintf "%zd" format, Ruby 1.9.3 defines macro `_GNU_SOURCE`.

If `_GNU_SOURCE` is defined, MinGW32 defines `__USE_MINGW_ANSI_STDIO` macro.

But MinGW64 ignores `_GNU_SOURCE` macro.

The workaround is define macro `USE_MINGW_ANSI_STDIO` like this:

```
./configure --build=x86_64-w64-mingw32 CFLAGS="-O2 -finline-functions -DUSE_MINGW_ANSI_STDIO"
```

I am not sure this is a bug, but I found MinGW64 cannot handle `-std=c99` flag in this case.

Consider this:

```
C:\work>type foo.c
```

```
#include
```

```
int main()
```

```
{
```

```
printf("%zd", (size_t)10);
```

```
return 0;
```

```
}
```

```
C:\work>gcc -v
```

```
Using built-in specs.
```

```
COLLECT_GCC=gcc
```

```
COLLECT_LTO_WRAPPER=c:/mingw47/mingw/bin/./libexec/gcc/i686-pc-mingw32/4.7.0/lt
```

```
o-wrapper.exe
Target: i686-pc-mingw32
Configured with: ../src/configure --prefix=/c/temp/gcc/dest --with-gmp=/c/temp/g
cc/gmp --with-mpfr=/c/temp/gcc/mpfr --with-mpc=/c/temp/gcc/mpc --enable-language
s=c,c++ --with-arch=i686 --with-tune=generic --disable-libstdcxx-pch --disable-n
ls --disable-shared --disable-sjlj-exceptions --disable-win32-registry --enable-
checking=release --enable-lto
Thread model: win32
gcc version 4.7.0 (GCC)
```

```
C:\work>gcc -std=c99 foo.c -ofoo.exe
```

```
C:\work>foo
10
```

```
C:\work>gcc -D_GNU_SOURCE foo.c -ofoo.exe
```

```
C:\work>foo
10
```

```
C:\work>gcc -D__USE_MINGW_ANSI_STDIO foo.c -ofoo.exe
```

```
C:\work>foo
10
```

```
C:\work>gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=c:/mingw64_7/bin/./libexec/gcc/x86_64-w64-mingw32/4.7.0/lto
-wrapper.exe
Target: x86_64-w64-mingw32
Configured with: ../gcc-4.7.0/configure --build=x86_64-w64-mingw32 --enable-targ
ets=all --disable-multilib --enable-64bit --prefix=/mingw64 --with-sysroot=/ming
w64 --disable-shared --enable-static --disable-nls --enable-version-specific-run
time-libs --disable-win32-registry --without-dwarf2 --enable-sjlj-exceptions --e
nable-fully-dynamic-string --enable-languages=c,ada,lto,c++,objc,obj-c++,fortran
--enable-libgomp --enable-lto --enable-libssp --enable-gnattools --disable-boots
trap --with-gcc --with-gnu-as --with-gnu-ld --with-stabs --enable-interwork --wi
th-mpfr-include=/home/beta/gcc-build/./gcc-4.7.0/mpfr/src --with-mpfr-lib=/home
/beta/gcc-build/mpfr/src/.libs
Thread model: win32
gcc version 4.7.0 (GCC)
```

```
C:\work>gcc -std=c99 foo.c -ofoo.exe
```

```
C:\work>foo
zd
```

```
C:\work>gcc -D_GNU_SOURCE foo.c -ofoo.exe
```

```
C:\work>foo
zd
```

```
C:\work>gcc -D__USE_MINGW_ANSI_STDIO foo.c -ofoo.exe
```

```
C:\work>foo
10
```

#3 - 05/09/2012 01:21 AM - raylinn@gmail.com (ray linn)

it is due to the mingw/mingw64 both use the default msvcrt.dll, which is not C99 compatible. but both of them provide a extension to override the msvcrt printf. and MinGW64 prefer use __USE_MINGW_ANSI_STDIO as a switch for this, instead of __GNU_SOURCE as GNU source doesn't mean at all POSIX necessarily.

I recommend update the GNU_SOURCE to __USE_MINGW_ANSI_STDIO in ruby code,this will provide more compatible to both mingw and mingw64.

#4 - 05/09/2012 02:35 AM - luislavena (Luis Lavena)

- Category set to build
- Assignee changed from mrkn (Kenta Murata) to nobu (Nobuyoshi Nakada)
- Target version set to 1.9.3

#5 - 05/09/2012 02:35 AM - luislavena (Luis Lavena)

- *Status changed from Open to Assigned*

#6 - 05/09/2012 03:22 PM - phasis68 (Heesob Park)

Due to the Revision r33978, this issue is not occurred with the trunk version.
I think r33978 should be backported to 1.9.3.

#7 - 02/26/2013 10:19 AM - naruse (Yui NARUSE)

- *Target version changed from 1.9.3 to 2.6*

#8 - 12/25/2017 06:15 PM - naruse (Yui NARUSE)

- *Target version deleted (2.6)*