Ruby - Bug #19331

--enable-rpath results in incorrect RPATH in Ruby 3.1.3+

01/11/2023 02:29 PM - Kulikjak (Jakub Kulik)

Status:	Feedback		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	3.1.3	**Backport:**	2.7: UNKNOWN, 3.0: UNKNOWN, 3.1: UNKNOWN, 3.2: UNKNOWN

Description

I just updated Ruby from 3.1.2 to 3.1.3 and found out that all .so libraries from the enc directory have wrong RPATH/RUNPATH (when building with --enable-rpath). I first hit this on Solaris, but my testing on Linux resulted in the same thing. It can be easily reproduced with the following:

```
wget http://cache.ruby-lang.org/pub/ruby/3.1/ruby-3.1.3.tar.gz
tar xzf ruby-3.1.3.tar.gz
cd ruby-3.1.3
./configure --prefix=/usr/ruby/3.1 --mandir=/usr/ruby/3.1/share/man --bindir=/usr/ruby/3.1/bin --s
bindir=/usr/ruby/3.1/sbin --libdir=/usr/ruby/3.1/lib/amd64 --with-rubylibprefix=/usr/ruby/3.1/lib/
ruby --enable-shared --enable-rpath
/usr/bin/make -j 16 -l 32
objdump -x ./.ext/x86_64-linux/enc/windows_31j.so | grep RPATH
```

While previously this resulted in the following output:
RPATH   /usr/ruby/3.1/lib/amd64

in 3.1.3 it is wrongly set to the build directory:
RPATH   /home/jkulik/rubytest/ruby-3.1.3

I tried looking for some obvious change but didn't find the core of this issue yet. All I found out is that generated enc.mk differs in prefix and libdir

```
-prefix = /usr/ruby/3.1
+prefix = /home/jkulik/rubytest/ruby-3.1.3
 exec_prefix = $(prefix)
-libdir = $(exec_prefix)/lib/amd64
+libdir = /home/jkulik/rubytest/ruby-3.1.3
```

rbconfig.rb is similar in both versions, but when I print out CONFIG in make_encmake.rb right after load File.expand_path("lib/mkmf.rb", dir), I can see the differences from above, so it's probably something in the lib/mkmf.rb file that changes that.

I am seeing the same issue in the current latest ruby 3.1 branch cloned from github.

History

#1 - 01/12/2023 03:25 PM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Feedback

I tried the exactly same commands on Ubuntu 22.04, but couldn't reproduce the result of grep.

And the command to build windows_31j.so seems using the expected options.

```
gcc -shared -o .ext/x86_64-linux/enc/windows_31j.so enc/windows_31j.o -L. -L. -L. -fstack-protector-strong -rd
ynamic -Wl,-export-dynamic -Wl,--no-as-needed -Wl,--compress-debug-sections=zlib    -Wl,-rpath,/usr/ruby/3.1/l
ib/amd64 -L/usr/ruby/3.1/lib/amd64 -lruby -lm
```

#2 - 01/13/2023 08:43 AM - Kulikjak (Jakub Kulik)

nobu (Nobuyoshi Nakada) wrote in #note-1:

> I tried the exactly same commands on Ubuntu 22.04, but couldn't reproduce the result of grep.

That's interesting (and unexpected) - let me investigate it more.

#3 - 01/13/2023 10:19 AM - Kulikjak (Jakub Kulik)

I think I found it. The difference is in whether ruby already available on a given machine is detected/how it's used.

I found out that while 3.1.2 generated the enc.mk was generated like this:
./miniruby -I./lib -I. -I.ext/common ./enc/make_encmake.rb --builtin-encs="enc/ascii.o enc/us_ascii.o enc/unicode.o enc/utf_8.o" --builtin-transes="enc/trans/newline.o" --module enc.mk

However, in 3.1.3 (if base ruby is detected during configure), it is generated with system ruby like this:
/usr/bin/ruby --disable=gems -r./x86_64-linux-fake ./enc/make_encmake.rb --builtin-encs="enc/ascii.o enc/us_ascii.o enc/unicode.o enc/utf_8.o" --builtin-transes="enc/trans/newline.o" --module enc.mk

In this case, tool/fake.rb (-r./x86_64-linux-fake) is used and that is what changes prefix and libdir somewhere during the load File.expand_path("lib/mkmf.rb", dir) (in make_encmake.rb) execution.

When I run the 3.1.3 configure with --without-baseruby, it works again and the enc.mk generation looks the same as before.

So, I have a workaround, but my guess is that this behavior is still not desirable?

#4 - 04/12/2023 09:43 AM - Kulikjak (Jakub Kulik)

- Subject changed from --enable-rpath results in incorrect RPATH in Ruby 3.1.3 to --enable-rpath results in incorrect RPATH in Ruby 3.1.3+

I see the same issue in 3.1.4 as well.

As mentioned in the previous comment, I can workaround it with --without-baseruby option, although I doubt this is a desirable behavior.