Ruby master - Bug #14009

macOS High Sierra and “fork” compatibility

10/12/2017 06:18 PM - ticky (Jessica Stokes)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.4.2p198 (2017-09-14 revision 59899) [x86_64-darwin17]	**Backport:**	2.3: DONE, 2.4: DONE

Description

This was originally discussed on the issue tracker for Puma (https://github.com/puma/puma/issues/1421), however, it is possible that it would make more sense for inclusion in the Ruby implementation itself.

macOS High Sierra has changed the behaviour of the fork syscall such that initialising Objective-C APIs in forked processes are treated as errors. (see http://sealiesoftware.com/blog/archive/2017/6/5/Objective-C_and_fork_in_macOS_1013.html for more details)

This means that many applications which use forking to process concurrently will forcibly crash if the forked process calls out to any Objective-C library when Objective-C was not already initialised in the host process. This includes Puma, Unicorn, iodine and Passenger.

A workaround I proposed for Puma was to implicitly load the Objective-C runtime before performing any forks (https://github.com/puma/puma/issues/1421#issuecomment-332650703). This causes forked processes using other Objective-C APIs to not crash.

The workaround (specific to Puma’s DSL) was:

```
# Work around macOS 10.13 and later being very picky about
# `fork` usage and interactions with Objective-C code
# see: <https://github.com/puma/puma/issues/1421>
if /darwin/ =~ RUBY_PLATFORM
  before_fork do
    require 'fiddle'
    # Dynamically load Foundation.framework, ~implicitly~ initialising
    # the Objective-C runtime before any forking happens in Puma
    Fiddle.dlopen '/System/Library/Frameworks/Foundation.framework/Foundation'
  end
end
```

A similar fix has now been included in Passenger (https://github.com/phusion/passenger/blob/2a55a84e5de721d8bd806a8fea0bcedf27583c29/src/ruby_supportlib/phusion_passenger/loader_shared_helpers.rb#L84-L105).

It was, however, proposed that it might make more sense for Ruby on macOS High Sierra and onward to implicitly initialise the Objective-C framework itself, so that forked processes work roughly as expected even if they intend to use Objective-C APIs.

I understand that this is a heavy-handed move, but it seems to me that this relatively common technique will remain broken in Ruby unless everyone deploys a workaround (iodine has already expressed disinterest in doing so) or Ruby adopts one at the higher level.

This issue is also applicable to all Ruby versions which support fork and run on macOS High Sierra.

Thank you for your time. :)

Related issues:

Related to Ruby master - Feature #5446: at_fork callback API	**Assigned**

Associated revisions

Revision 8b182a7f - 10/14/2017 03:55 PM - nobu (Nobuyoshi Nakada)

configure.ac: link Foundation framework

- configure.ac (XLDFLAGS): link against Foundation framework and let __NSPlaceholderDictionary initialize, to get rid of crash after fork on macOS High Sierra. [ruby-core:83239] [Bug #14009] git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@60182 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 60182 - 10/14/2017 03:55 PM - nobu (Nobuyoshi Nakada)

configure.ac: link Foundation framework

- configure.ac (XLDFLAGS): link against Foundation framework and let __NSPlaceholderDictionary initialize, to get rid of crash after fork on macOS High Sierra. [ruby-core:83239] [Bug #14009]

Revision 60182 - 10/14/2017 03:55 PM - nobu (Nobuyoshi Nakada)

configure.ac: link Foundation framework

- configure.ac (XLDFLAGS): link against Foundation framework and let __NSPlaceholderDictionary initialize, to get rid of crash after fork on macOS High Sierra. [ruby-core:83239] [Bug #14009]

Revision 60182 - 10/14/2017 03:55 PM - nobu (Nobuyoshi Nakada)

configure.ac: link Foundation framework

- configure.ac (XLDFLAGS): link against Foundation framework and let __NSPlaceholderDictionary initialize, to get rid of crash after fork on macOS High Sierra. [ruby-core:83239] [Bug #14009]

Revision 1a028670 - 12/20/2017 11:54 AM - nagachika (Tomoyuki Chikanaga)

merge revision(s) 60182: [Backport #14009]

```
    configure.ac: link Foundation framework

    * configure.ac (XLDFLAGS): link against Foundation framework and
      let __NSPlaceholderDictionary initialize, to get rid of crash
      after fork on macOS High Sierra.  [ruby-core:83239] [Bug #14009]
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_4@61358 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 61358 - 12/20/2017 11:54 AM - nagachika (Tomoyuki Chikanaga)

merge revision(s) 60182: [Backport #14009]

```
configure.ac: link Foundation framework

* configure.ac (XLDFLAGS): link against Foundation framework and
  let __NSPlaceholderDictionary initialize, to get rid of crash
  after fork on macOS High Sierra.  [ruby-core:83239] [Bug #14009]
```

Revision 766c3744 - 01/31/2018 01:58 PM - usa (Usaku NAKAMURA)

merge revision(s) 60182: [Backport #14009]

```
    configure.ac: link Foundation framework

    * configure.ac (XLDFLAGS): link against Foundation framework and
      let __NSPlaceholderDictionary initialize, to get rid of crash
      after fork on macOS High Sierra.  [ruby-core:83239] [Bug #14009]
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_3@62144 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 62144 - 01/31/2018 01:58 PM - usa (Usaku NAKAMURA)

merge revision(s) 60182: [Backport #14009]

```
configure.ac: link Foundation framework

* configure.ac (XLDFLAGS): link against Foundation framework and
  let __NSPlaceholderDictionary initialize, to get rid of crash
  after fork on macOS High Sierra.  [ruby-core:83239] [Bug #14009]
```

History

#1 - 10/12/2017 08:18 PM - hongli (Hongli Lai)

Perhaps it helps if I further clarify this issue for those who are unfamiliar with it.

As you probably already know, forking (but without exec'ing) in a multithreaded environment is inherently dangerous and the environment must be carefully written to support such a thing. Apple's Objective-C libraries have traditionally not supported being called in a forked (but not exec'd) child process at all, but since High Sierra 10.13 they've tried to add limited support for this. However in doing so they've also defined rules on what is *not* allowed after forking. One of the rules state that it is not allowed to call the initialize function of certain Objective-C classes after forking; that may only happen before forking.

Makes sense so far. The problem occurs because of a combination of three things:

1. Ruby itself is not linked to any Objective-C libraries, and so does not initialize Objective-C classes by itself.
2. The user may use gems that do link to Objective-C libraries. Due to how these gems are used, it can occur that these gems end up calling Objective-C initializers after the app server has forked.
3. The new Apple-enforced rule checks then abort the process with a warning like this:

```
objc[81924]: +[__NSPlaceholderDictionary initialize] may have been in progress in another thread when fork() w
as called.
objc[81924]: +[__NSPlaceholderDictionary initialize] may have been in progress in another thread when fork() w
as called. We cannot safely call it or ignore it in the fork() child process. Crashing instead. Set a breakpoi
nt on objc_initializeAfterForkError to debug.
```

By itself, Apple's error check makes sense. Forking is dangerous. But all these factors combined make less sense. Adding a workaround in Ruby (in the form of ensuring that Objective-C initializers are called before forking) will at least ensure that we return to pre-High Sierra behavior.

Adding a workaround inside Ruby would only be a part of the whole solution. In order to fully fix the problem, cooperation from the wider Ruby community is required: all the gem authors will also have to ensure that native libraries don't spawn any threads until the app server has forked. But I do believe that having a workaround inside Ruby is an essential part of the entire fix, because updating all gems takes a lot of time and effort.

#2 - 10/13/2017 12:52 AM - shyouhei (Shyouhei Urabe)

- Related to Feature #5446: at_fork callback API added

#3 - 10/13/2017 12:58 AM - myst (Boaz Segev)

I should point out that "Foundation.framework/Foundation" seems to be enough.

This is the C code I'm currently considering for the iodine Ruby server.

```
#ifdef __APPLE__
  void *obj_c_runtime = dlopen("Foundation.framework/Foundation", RTLD_LAZY);
  /* iodine runs here and `fork` might be called */
  dlclose(obj_c_runtime);
#else
  /* iodine runs here and `fork` might be called */
#endif
```

Comments:

- The __APPLE__ directive will only work on gcc (gnu compiler), clang and intel compilers (other compilers might be supported, but I don't know).
- The code doesn't check for errors, because a quite failure doesn't effect behavior. If Objective-C isn't available, than forking will not be effected.
- The code doesn't check for High Sierra specifically and will load Objective-C on all macOS versions (anything over 10.9).

The memory increase was from 4388Kb (Ruby VM) to ~4560Kb (compared using Fiddle, not a direct C call), which seems insignificant on the larger scale of things.

For comparison:

- Baseline: loading the Objective-C using Fiddle caused a 3.9% increase in memory consumption relative to the basic the VM core.
- Loading the Socket Standard library (require 'socket') caused a 16% increase in memory consumption relative to the basic the VM core.
- Loading Sinatra caused more than a 196% increase in memory consumption relative to the basic the VM core.
- Loading Rails caused more than a 264% increase in memory consumption relative to the basic the VM core.

It's my understanding that a simple patch will load the Objective-C library within the ruby_init() function and ignore any errors in loading the library.

A more complex approach will lazy load the library only when Process#fork is called... but this might effect C extensions that might call fork directly and it might also effect the use of Ruby as a library (when ruby_init() is called within a C application that uses Ruby internally).

Personally, I'm both lazy and paranoid and I would assume that the simple approach would both work better and fail better (expose issues earlier rather than later).

#4 - 10/13/2017 02:23 AM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Feedback

Does linking Foundation framework instead of CoreFoundation framework work?

For 2.4:

```
diff --git a/configure.in b/configure.in
index bb7cc4daa5..4a2243afa1 100644
--- a/configure.in
+++ b/configure.in
@@ -4144,8 +4144,8 @@ AS_CASE(["$target_os"],
    ],
     [darwin*], [
    RUBY_APPEND_OPTION(CFLAGS, -pipe)
-   RUBY_APPEND_OPTION(XLDFLAGS, [-framework CoreFoundation])
-   RUBY_APPEND_OPTION(LIBRUBYARG_STATIC, [-framework CoreFoundation])
+   RUBY_APPEND_OPTION(XLDFLAGS, [-framework Foundation])
+   RUBY_APPEND_OPTION(LIBRUBYARG_STATIC, [-framework Foundation])
    ],
     [osf*], [
    if test "$GCC" != "yes" ; then
```

For trunk:

```
diff --git a/configure.ac b/configure.ac
index e6cd95cc9d..6cd99e6ef2 100644
--- a/configure.ac
+++ b/configure.ac
@@ -4116,8 +4116,8 @@ AS_CASE(["$target_os"],
    ],
     [darwin*], [
    RUBY_APPEND_OPTION(CFLAGS, -pipe)
-   RUBY_APPEND_OPTION(XLDFLAGS, [-framework CoreFoundation])
-   RUBY_APPEND_OPTION(LIBRUBYARG_STATIC, [-framework CoreFoundation])
+   RUBY_APPEND_OPTION(XLDFLAGS, [-framework Foundation])
+   RUBY_APPEND_OPTION(LIBRUBYARG_STATIC, [-framework Foundation])
    ],
     [osf*], [
    AS_IF([test "$GCC" != "yes" ], [
```

#5 - 10/13/2017 02:35 AM - myst (Boaz Segev)

I can't test because I can't install High Sierra on my machine (I'm also a musician and my professional audio applications don't support High Sierra).... ticky (Jessica Stokes)?

#6 - 10/13/2017 06:02 AM - hongli (Hongli Lai)

nobu (Nobuyoshi Nakada) wrote:

> Does linking Foundation framework instead of CoreFoundation framework work?

According to my tests on High Sierra, linking to Foundation helps, but linking to CoreFoundation does not. CoreFoundation does not (at load time) initialize any Objective-C classes. Here are test scripts which simulate the issue (and a working fix and a non-working fix) on High Sierra:

broken.rb

```
# This script simulates a child process that initializes ObjC classes,
# and should crash.
require 'fiddle'

def init_objc_classes
  Fiddle.dlopen('/System/Library/Frameworks/Foundation.framework/Foundation')
end

pid = fork do
  init_objc_classes
end
Process.waitpid(pid)
```

Test run:

```
$ ruby broken.rb
objc[768]: +[__NSPlaceholderDictionary initialize] may have been in progress in another thread when fork() was
 called.
objc[768]: +[__NSPlaceholderDictionary initialize] may have been in progress in another thread when fork() was
 called. We cannot safely call it or ignore it in the fork() child process. Crashing instead. Set a breakpoint
 on objc_initializeAfterForkError to debug.
```

fixed.rb

```
# This script simulates a process that initializes ObjC classes before,
# forking, and should NOT crash.
require 'fiddle'

def init_objc_classes
  Fiddle.dlopen('/System/Library/Frameworks/Foundation.framework/Foundation')
end

init_objc_classes
pid = fork do
  # The following does nothing because ObjC classes
  # are already initialized before forking.
  init_objc_classes
end
Process.waitpid(pid)
```

```
$ ruby fixed.rb
(no crash)
```

nobu-fix.rb

```
# This script simulates a process that tries to initialize ObjC classes before,
# forking, and still crashes because the method doesn't actually work. This
# simulates nobu's proposed fix in https://bugs.ruby-lang.org/issues/14009#note-4
require 'fiddle'

def invoke_nobu_fix
  Fiddle.dlopen('/System/Library/Frameworks/CoreFoundation.framework/CoreFoundation')
end

def init_objc_classes
  Fiddle.dlopen('/System/Library/Frameworks/Foundation.framework/Foundation')
end

invoke_nobu_fix
pid = fork do
  init_objc_classes
end
Process.waitpid(pid)
```

Test run:

```
$ ruby nobu-fix.rb
objc[1066]: +[__NSPlaceholderDictionary initialize] may have been in progress in another thread when fork() wa
s called.
objc[1066]: +[__NSPlaceholderDictionary initialize] may have been in progress in another thread when fork() wa
s called. We cannot safely call it or ignore it in the fork() child process. Crashing instead. Set a breakpoin
t on objc_initializeAfterForkError to debug.
```

And here is a sanity check to verify that loading CoreFoundation in fact does not initialize prohibited ObjC classes:

```
# This script should not crash in the child. It should verify
# that CoreFoundation does not initialize any Objective-C
# classes.
require 'fiddle'

def load_core_foundation
  Fiddle.dlopen('/System/Library/Frameworks/CoreFoundation.framework/CoreFoundation')
end

load_core_foundation
pid = fork do
  load_core_foundation
end
Process.waitpid(pid)
```

Test run:

```
$ ruby corefoundation-sanity-check.rb
(no crash)
```

#7 - 10/13/2017 07:04 AM - nobu (Nobuyoshi Nakada)

"linking Foundation framework" means linking ruby and libruby.dylib with -framework Foundation compiler option.
You need to apply my patch to the source then rebuild it.

#8 - 10/13/2017 02:55 PM - myst (Boaz Segev)

EDIT:

Please ignore the original comment.

I tested linkage in Sierra (not High Sierra), the -framework instruction is honored even when no symbols are used.

The linker approach seems safe to use in macOS. I wish I could test it on High Sierra.

Original:

nobu (Nobuyoshi Nakada) , I'm wondering if some linkers might optimize this instruction away...?

My question is whether or not an optimized linker might ignore the instruction after noticing that no symbols from the shared library are used.

If this is the case, than the linker approach will work on some linkers and not on others.

I think the documentation I am referencing (the documentation for ld) is the same for both static and dynamic libraries (but maybe it's only how static libraries are resolved):

> -larchive
> --library=archive
> Add archive file archive to the list of files to link. This option may be used any number of times. ld will search its path-list for occurrences of libarchive.a for every archive specified. On systems which support shared libraries, ld may also search for libraries with extensions other than .a. Specifically, on ELF and SunOS systems, ld will search a directory for a library with an extension of .so before searching for one with an extension of .a. By convention, a .so extension indicates a shared library. The linker will search an archive only once, at the location where it is specified on the command line. **If the archive defines a symbol which was undefined in some object which appeared before the archive on the command line, the linker will include the appropriate file(s) from the archive**. However, an undefined symbol in an object appearing later on the command line will not cause the linker to search the archive again. See the -(option for a way to force the linker to search archives multiple times. You may list the same archive multiple times on the command line. This type of archive searching is standard for Unix linkers. However, if you are using ld on AIX, note that it is different from the behaviour of the AIX linker.

Because Ruby doesn't have any undefined symbols that are defined in the Foundation library, this library might be ignored and the patch will fail.

#9 - 10/13/2017 05:18 PM - ticky (Jessica Stokes)

Thank you all for your input! :)

I'll look at getting 2.4.2 built with nobu's patch and check how it looks from High Sierra.

#10 - 10/13/2017 05:30 PM - ticky (Jessica Stokes)

Alright, I've checked it out, and can confirm that the patch nobu (Nobuyoshi Nakada) suggested does indeed work around the original issue I encountered in Puma, with my mitigations removed from our application code.

I can also confirm that with that patch applied, none of the code snippets supplied by hongli (Hongli Lai) fail on High Sierra, which means this definitely works around the problem. :)

#11 - 10/13/2017 07:24 PM - mperham (Mike Perham)

The original cause was lazy loading the pg gem in the forked child process. Does this fix allow the same pg gem loading to work as before?

#12 - 10/13/2017 08:49 PM - hongli (Hongli Lai)

Regarding nobu's comment on whether linkers can ignore "useless" library links (comment #8): yes they can, but both on GNU/Linux systems as well as on macOS, this behavior is at present not the default.

On GNU systems, --as-needed lets the GNU linker skip linking unused libraries. There is one blog article out there that claims that --as-needed has been made the default on newer GNU systems, but I've not personally verified this, nor have I found any other sources so far.

macOS's linker has -dead_strip_dylibs which does the same.

So we don't have to worry about this issue.

#13 - 10/13/2017 09:34 PM - ticky (Jessica Stokes)

mperham (Mike Perham) loading pg within a fork works just fine with this patch :)

#14 - 10/14/2017 01:54 AM - nobu (Nobuyoshi Nakada)

Thank you for the confirmation.
As for "useless" library link, -u linker option could force to link it.
You can see which libraries are linked dynamically, by otool -L command.

#15 - 10/14/2017 03:55 PM - nobu (Nobuyoshi Nakada)

- Status changed from Feedback to Closed

Applied in changeset trunk|r60182.

---

configure.ac: link Foundation framework

- configure.ac (XLDFLAGS): link against Foundation framework and let __NSPlaceholderDictionary initialize, to get rid of crash after fork on macOS High Sierra. [ruby-core:83239] [Bug #14009]

#16 - 10/15/2017 08:50 AM - nagachika (Tomoyuki Chikanaga)

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

#17 - 10/15/2017 09:50 AM - nobu (Nobuyoshi Nakada)

Just an idea, environment variable DYLD_INSERT_LIBRARIES=/System/Library/Frameworks/Foundation.framework/Versions/Current/Foundation doesn't work?

#18 - 10/16/2017 01:01 AM - myst (Boaz Segev)

I'm happy this was merged to the Ruby trunk.

> Just an idea, environment variable
> DYLD_INSERT_LIBRARIES=/System/Library/Frameworks/Foundation.framework/Versions/Current/Foundation doesn't work?

Interesting idea :)

It would probably work exactly like the patch, except it will be user controlled instead of persistent.

#19 - 10/16/2017 06:44 AM - nobu (Nobuyoshi Nakada)

myst (Boaz Segev) wrote:

> It would probably work exactly like the patch, except it will be user controlled instead of persistent.

It's a temporary repair until the next release, of course.

#20 - 12/20/2017 01:45 AM - ticky (Jessica Stokes)

Just for reference, people seem confused about the fact this patch hasn't made it into Ruby 2.4.3; as far as I can tell it will only appear in version 2.5.
Is a backport forthcoming or is the idea for people who need High Sierra compatibility to simply move to 2.5?

https://github.com/puma/puma/issues/1421#issuecomment-352083160

#21 - 12/20/2017 03:24 AM - nagachika (Tomoyuki Chikanaga)

Thank you for your notice.
It's just due to my laziness that the backport has been delayed.
I will backport this soon.

#22 - 12/20/2017 03:50 AM - ticky (Jessica Stokes)

nagachika (Tomoyuki Chikanaga) thank you for the update! I am glad it hasn't been forgotten. :)

#23 - 12/20/2017 11:55 AM - nagachika (Tomoyuki Chikanaga)

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

ruby_2_4 r61358 merged revision(s) 60182.

#24 - 01/31/2018 01:58 PM - usa (Usaku NAKAMURA)

- Backport changed from 2.3: UNKNOWN, 2.4: DONE to 2.3: DONE, 2.4: DONE

ruby_2_3 r62144 merged revision(s) 60182.

#25 - 07/30/2018 04:35 PM - quaeler (loki der quaeler)

nagachika (Tomoyuki Chikanaga) wrote:

> ruby_2_4 r61358 merged revision(s) 60182.

With 2.4.4 (ruby 2.4.4p296 (2018-03-28 revision 63013) [x86_64-darwin17]), I am still seeing
09:27:49 processor.1 | objc[44872]: +[**NSCFConstantString initialize] may have been in progress in another thread when fork() was called.**
09:27:49 processor.1 | objc[44872]: +[NSCFConstantString initialize] may have been in progress in another thread when fork() was called. We cannot safely call it or ignore it in the fork() child process. Crashing instead. Set a breakpoint on objc_initializeAfterForkError to debug.

Shouldn't this be fixed since the backport happened?