Ruby trunk - Bug #14807

2.6.0-preview2 segfaults on OpenBSD due to missing pthread_condattr_init call

06/01/2018 08:51 PM - jeremyevans0 (Jeremy Evans)

Status:	Closed		
Priority:	Normal		
Assignee:	normalperson (Eric Wong)		
Target version:			
ruby -v:	ruby 2.6.0dev (2018-06-01 trunk 63545) [x86_64-openbsd]	**Backport:**	2.3: DONTNEED, 2.4: DONTNEED, 2.5: DONTNEED

Description

r63238 refactored thread_pthread.c, and where there was previously a pthread_condattr_init call to initialize the pthread_condattr_t value, it removed the call and passed the pthread_condattr_t* directly to pthread_condattr_setclock without initializing the value by calling pthread_condattr_init first. On some operating systems that works, but it's not required to work, and it segfaults on OpenBSD because the pthread_condattr_t is not initialized.

The attached patch should fix the problem.

History

#1 - 06/01/2018 10:12 PM - normalperson (Eric Wong)

Thanks, r63548

Btw, is PTHREAD_COND_INITIALIZER supported on OpenBSD?

Something like this:

```
--- a/thread_pthread.c
+++ b/thread_pthread.c
@@ -55,7 +55,7 @@ static struct {
 #if defined(HAVE_PTHREAD_CONDATTR_SETCLOCK) && \
defined(CLOCK_REALTIME) && defined(CLOCK_MONOTONIC) && \
defined(HAVE_CLOCK_GETTIME)
-static pthread_condattr_t condattr_mono;
+static pthread_condattr_t condattr_mono = PTHREAD_COND_INITIALIZER;
static pthread_condattr_t *condattr_monotonic = &condattr_mono;
 #else
static const void *const condattr_monotonic = NULL;
```

...And reverting your patch. Should save a few instructions at
startup since it avoids linkage and function call at startup.

#2 - 06/01/2018 10:23 PM - jeremyevans0 (Jeremy Evans)

normalperson (Eric Wong) wrote:

> Btw, is PTHREAD_COND_INITIALIZER supported on OpenBSD?

It's defined but I don't think it would be usable:

```
/usr/include/pthread.h:#define PTHREAD_COND_INITIALIZER NULL
```

#3 - 09/22/2018 06:08 PM - taca (Takahiro Kambe)

Hi,

The similar problem occurs on NetBSD 8.0_STABLE. (And I belive it would be occur on 7.2.)

PTHREAD_COND_INITIALIZER is for pthread_cond_t not for pthread_condattr_t.
So, initializing condattr_mono (via condattr_monotonic) with pthread_condattr_init() is correct way to fix the problem as the attached patch.

Best regards.

P.S.
Oh, trunk was already applied the patch!

#4 - 09/23/2018 08:04 AM - normalperson (Eric Wong)

> https://bugs.ruby-lang.org/issues/14807#change-74146

Right, already in trunk at r63548

And back to Jeremy's earlier comment:

> It's defined but I don't think it would be usable:
>
> /usr/include/pthread.h:#define PTHREAD_COND_INITIALIZER NULL

Without reading the OpenBSD pthreads library source code;
I suspect that it would be usable.

The condvar implementation can safely perform lazy-initialization
because it can rely on the underlying mutex for protection.

#5 - 09/29/2018 03:54 PM - taca (Takahiro Kambe)

Hi,

Wheather PTHREAD_COND_INITIALIZER is work on OpenBSD or not, it should be used for initialize pthread_cond_t variable and it should not be used for initialize pthread_condattr_t variable since they are diffrernet type and it cause compile error no NetBSD.

#6 - 05/25/2019 12:59 AM - jeremyevans0 (Jeremy Evans)

- Status changed from Open to Closed

Fixed by 832b601e49fd402ec7f30b36a95473131e93ae94. As taca explained, PTHREAD_COND_INITIALIZER should not be used for pthread_condattr_t.

Files

0001-Initialize-condattr_monotonic-via-pthread_condattr_i.patch	1.08 KB	06/01/2018	jeremyevans0 (Jeremy Evans)