Ruby master - Bug #17189

test_thread.rb in testsuite deadlocks with musl libc 1.2.1

09/25/2020 05:56 PM - ncopa (Natanael Copa)

Status:	Open		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.7.1p83 (2020-03-31 revision a0c7c23c9c) [x86_64-linux-musl]	**Backport:**	2.5: UNKNOWN, 2.6: UNKNOWN, 2.7: UNKNOWN

Description

When we build ruby-2.7.1 in alpine linux the test suite deadlocks in test_thread.rb.

There are multiple processes:

```
$ ps xa | grep ruby
12401 ncopa     0:01 /home/ncopa/aports/main/ruby/src/ruby-2.7.1/ruby --disable-gems ./bootstrapte
st/runner.rb --ruby=ruby -I./lib --disable-gems -q
24150 ncopa    11:00 /home/ncopa/aports/main/ruby/src/ruby-2.7.1/ruby -I/home/ncopa/aports/main/ru
by/src/ruby-2.7.1/lib --disable-gems -W0 bootstraptest.tmp.rb
24152 ncopa     0:00 /home/ncopa/aports/main/ruby/src/ruby-2.7.1/ruby -I/home/ncopa/aports/main/ru
by/src/ruby-2.7.1/lib --disable-gems -W0 bootstraptest.tmp.rb
```

I'm gonna focus on the two last ones, 24150 and 24152.

pid 24150

strace shows that it waits on ppoll:

```
$ doas strace -p 24150
doas (ncopa@ncopa-edge-x86_64) password:
strace: Process 24150 attached
ppoll([{fd=3, events=POLLIN}], 1, NULL, NULL, 8
```

gdb shows two threads:

```
Attaching to process 24150
[New LWP 24151]
__cp_end () at src/thread/x86_64/syscall_cp.s:29
29      src/thread/x86_64/syscall_cp.s: No such file or directory.
(gdb) info threads
  Id   Target Id                   Frame
* 1    LWP 24150 "ruby"            __cp_end () at src/thread/x86_64/syscall_cp.s:29
  2    LWP 24151 "bootstraptest.*" VM_EP_LEP (ep=ep@entry=0x7ff1e63e7988) at vm.c:51
(gdb) bt
#0  __cp_end () at src/thread/x86_64/syscall_cp.s:29
#1  0x00007ff1e93fbfeb in __syscall_cp_c (nr=271, u=<optimized out>, v=<optimized out>, w=<optimiz
ed out>, x=<optimized out>,
    y=<optimized out>, z=0) at src/thread/pthread_cancel.c:33
#2  0x00007ff1e93ca14b in ppoll (fds=fds@entry=0x7ffcdb9d0520, n=n@entry=1, to=to@entry=0x0, mask=
mask@entry=0x0)
    at src/linux/ppoll.c:24
#3  0x00007ff1e9263160 in rb_sigwait_sleep (th=th@entry=0x7ff1e9440ae0, sigwait_fd=sigwait_fd@entr
y=3, rel=rel@entry=0x0)
    at hrtime.h:148
#4  0x00007ff1e92638cb in native_sleep (th=th@entry=0x7ff1e9440ae0, rel=rel@entry=0x0) at thread_p
thread.c:2145
#5  0x00007ff1e92658c4 in rb_thread_sleep_interruptible () at thread.c:1334
#6  0x00007ff1e920a27f in waitpid_sleep (x=x@entry=140723992987392) at process.c:1123
#7  0x00007ff1e91836d1 in rb_ensure (b_proc=b_proc@entry=0x7ff1e920a26b <waitpid_sleep>, data1=dat
a1@entry=140723992987392,
    e_proc=e_proc@entry=0x7ff1e920d463 <waitpid_cleanup>, data2=data2@entry=140723992987392) at ev
```

```
al.c:1129
#8  0x00007ff1e920dde1 in waitpid_wait (w=0x7ffcdb9d0700) at process.c:1182
#9  rb_waitpid (pid=pid@entry=24152, st=st@entry=0x7ffcdb9d0774, flags=<optimized out>) at process
.c:1221
#10 0x00007ff1e920dfa3 in proc_wait (argc=1, argv=0x7ff1e8f79090) at process.c:1260
#11 0x00007ff1e9285503 in vm_call_cfunc_with_frame (empty_kw_splat=<optimized out>, cd=0x7ff1e64e8
2f0, calling=<optimized out>,
    reg_cfp=0x7ff1e9078fd0, ec=0x7ff1e9440140) at vm_insnhelper.c:2514
#12 vm_call_cfunc (ec=0x7ff1e9440140, reg_cfp=0x7ff1e9078fd0, calling=<optimized out>, cd=0x7ff1e6
4e82f0) at vm_insnhelper.c:2539
#13 0x00007ff1e9281d23 in vm_sendish (ec=ec@entry=0x7ff1e9440140, reg_cfp=reg_cfp@entry=0x7ff1e907
8fd0, cd=0x7ff1e64e82f0,
    block_handler=block_handler@entry=0, method_explorer=method_explorer@entry=0x7ff1e9287476 <vm_
search_method_wrap>)
    at vm_insnhelper.c:4023
#14 0x00007ff1e928dce7 in vm_exec_core (ec=ec@entry=0x7ff1e9440140, initial=initial@entry=0) at in
sns.def:801
#15 0x00007ff1e929065d in rb_vm_exec (ec=<optimized out>, mjit_enable_p=mjit_enable_p@entry=1) at
vm.c:1929
#16 0x00007ff1e9291114 in rb_iseq_eval_main (iseq=iseq@entry=0x7ff1e8f56fe0) at vm.c:2179
#17 0x00007ff1e9181424 in rb_ec_exec_node (ec=ec@entry=0x7ff1e9440140, n=n@entry=0x7ff1e8f56fe0) a
t eval.c:278
#18 0x00007ff1e9184da2 in ruby_run_node (n=0x7ff1e8f56fe0) at eval.c:336
#19 0x00005597798fe11b in main (argc=<optimized out>, argv=<optimized out>) at ./main.c:50
(gdb) thread 2
[Switching to thread 2 (LWP 24151)]
#0  VM_EP_LEP (ep=ep@entry=0x7ff1e63e7988) at vm.c:51
51          return ep;
(gdb) bt
#0  VM_EP_LEP (ep=ep@entry=0x7ff1e63e7988) at vm.c:51
#1  0x00007ff1e9283799 in VM_CF_LEP (cfp=<optimized out>, cfp=<optimized out>) at vm.c:97
#2  VM_CF_BLOCK_HANDLER (cfp=<optimized out>) at vm.c:97
#3  0x00007ff1e928af18 in check_block_handler (ec=<optimized out>, ec=<optimized out>) at vm.c:115
9
#4  0x00007ff1e929237e in vm_yield (kw_splat=0, argv=0x0, argc=0, ec=0x7ff1e6522a60) at vm.c:1179
#5  rb_yield_0 (argc=argc@entry=0, argv=argv@entry=0x0) at vm_eval.c:1227
#6  0x00007ff1e9292521 in loop_i (_=_@entry=0) at vm_eval.c:1330
#7  0x00007ff1e9183143 in rb_vrescue2 (b_proc=0x7ff1e9292514 <loop_i>, data1=0, r_proc=0x7ff1e9281
7d5 <loop_stop>, data2=0,
    args=args@entry=0x7ff1e63e74e0) at eval.c:990
#8  0x00007ff1e918334e in rb_rescue2 (b_proc=<optimized out>, data1=<optimized out>, r_proc=<optim
ized out>, data2=<optimized out>)
    at eval.c:967
#9  0x00007ff1e9285503 in vm_call_cfunc_with_frame (empty_kw_splat=<optimized out>, cd=0x7ff1e6518
840, calling=<optimized out>,
    reg_cfp=0x7ff1e64e78d0, ec=0x7ff1e6522a60) at vm_insnhelper.c:2514
#10 vm_call_cfunc (ec=0x7ff1e6522a60, reg_cfp=0x7ff1e64e78d0, calling=<optimized out>, cd=0x7ff1e6
518840) at vm_insnhelper.c:2539
#11 0x00007ff1e9281d23 in vm_sendish (ec=ec@entry=0x7ff1e6522a60, reg_cfp=reg_cfp@entry=0x7ff1e64e
78d0, cd=cd@entry=0x7ff1e6518840,
    block_handler=<optimized out>, method_explorer=method_explorer@entry=0x7ff1e9287476 <vm_search
_method_wrap>)
    at vm_insnhelper.c:4023
#12 0x00007ff1e928dc52 in vm_exec_core (ec=ec@entry=0x7ff1e6522a60, initial=initial@entry=0) at in
sns.def:782
#13 0x00007ff1e929065d in rb_vm_exec (ec=ec@entry=0x7ff1e6522a60, mjit_enable_p=mjit_enable_p@entr
y=1) at vm.c:1929
#14 0x00007ff1e9291189 in invoke_block (ec=ec@entry=0x7ff1e6522a60, iseq=iseq@entry=0x7ff1e8f56bf8
, self=self@entry=140676971683160,
    cref=cref@entry=0x0, type=type@entry=572653569, opt_pc=0, captured=<optimized out>, captured=<
optimized out>) at vm.c:1044
#15 0x00007ff1e9292e5b in invoke_iseq_block_from_c (me=0x0, is_lambda=0, cref=0x0, passed_block_ha
ndler=0, kw_splat=0,
    argv=0x7ff1e63e7890, argc=0, self=140676971683160, captured=0x7ff1e64ff030, ec=0x7ff1e6522a60)
 at vm.c:1116
#16 invoke_block_from_c_proc (me=0x0, is_lambda=0, passed_block_handler=0, kw_splat=0, argv=0x7ff1
e63e7890, argc=0,
```

```
    self=140676971683160, proc=0x7ff1e6522a60, ec=0x7ff1e6522a60) at vm.c:1216
#17 vm_invoke_proc (passed_block_handler=0, kw_splat=0, argv=0x7ff1e63e7890, argc=0, self=14067697
1683160, proc=0x7ff1e6522a60,
    ec=0x7ff1e6522a60) at vm.c:1238
#18 rb_vm_invoke_proc (ec=0x7ff1e6522a60, proc=proc@entry=0x7ff1e64ff030, argc=0, argv=0x7ff1e63e7
890, kw_splat=0,
    passed_block_handler=passed_block_handler@entry=0) at vm.c:1259
#19 0x00007ff1e92657c5 in thread_do_start (th=th@entry=0x7ff1e64e89c0) at thread.c:697
#20 0x00007ff1e92663fa in thread_start_func_2 (th=<optimized out>, stack_start=<optimized out>) at
 thread.c:745
#21 0x00007ff1e9266781 in thread_start_func_1 (th_ptr=<optimized out>) at thread_pthread.c:1015
#22 0x00007ff1e93fcaa9 in start (p=0x7ff1e64e7ae8) at src/thread/pthread_create.c:196
#23 0x00007ff1e93fec5a in __clone () at src/thread/x86_64/clone.s:22
Backtrace stopped: frame did not save the PC
```

pid 24152

The second process waits on a futex says strace:

```
$ doas strace -p 24152
strace: Process 24152 attached
futex(0x7ff1e943ffd8, FUTEX_WAIT_PRIVATE, 2147483650, NULL
```

Backtrace:

```
Attaching to process 24152
Reading symbols from /home/ncopa/aports/main/ruby/src/ruby-2.7.1/ruby...
Reading symbols from /home/ncopa/aports/main/ruby/src/ruby-2.7.1/libruby.so.2.7.1...
Reading symbols from /lib/ld-musl-x86_64.so.1...
Reading symbols from /usr/lib/debug//lib/ld-musl-x86_64.so.1.debug...
Reading symbols from /lib/libz.so.1...
Reading symbols from /usr/lib/debug//lib/libz.so.1.2.11.debug...
Reading symbols from /usr/lib/libgmp.so.10...
Reading symbols from /usr/lib/debug//usr/lib/libgmp.so.10.4.0.debug...
Reading symbols from /home/ncopa/aports/main/ruby/src/ruby-2.7.1/.ext/x86_64-linux-musl/enc/encdb.
so...
Reading symbols from /home/ncopa/aports/main/ruby/src/ruby-2.7.1/.ext/x86_64-linux-musl/enc/trans/
transdb.so...
__futexwait (priv=128, val=-2147483646, addr=0x7ff1e943ffd8 <__malloc_lock>) at ./arch/x86_64/sysc
all_arch.h:40
40      ./arch/x86_64/syscall_arch.h: No such file or directory.
(gdb) info threads
  Id   Target Id            Frame
* 1    process 24152 "ruby" __futexwait (priv=128, val=-2147483646, addr=0x7ff1e943ffd8 <__malloc_
lock>)
    at ./arch/x86_64/syscall_arch.h:40
(gdb) bt
#0  __futexwait (priv=128, val=-2147483646, addr=0x7ff1e943ffd8 <__malloc_lock>) at ./arch/x86_64/
syscall_arch.h:40
#1  __lock (l=l@entry=0x7ff1e943ffd8 <__malloc_lock>) at src/thread/__lock.c:44
#2  0x00007ff1e93ce6ee in rdlock () at src/malloc/mallocng/glue.h:63
#3  malloc (n=n@entry=56) at src/malloc/mallocng/malloc.c:337
#4  0x00007ff1e9197a22 in objspace_xmalloc0 (objspace=0x7ff1e907a570, size=56) at gc.c:9860
#5  0x00007ff1e9197ab6 in ruby_xmalloc0 (size=<optimized out>) at vm_core.h:1806
#6  ruby_xmalloc_body (size=<optimized out>) at gc.c:10089
#7  0x00007ff1e6dfc2c0 in ?? ()
#8  0x00007ff1e8f564f0 in ?? ()
#9  0x00007ff1e8f56518 in ?? ()
#10 0x0000000000000d41 in ?? ()
#11 0x00007ffcdb9cfe40 in ?? ()
#12 0x00007ff1e927c297 in iv_index_tbl_make (obj=<optimized out>) at variable.c:1138
Backtrace stopped: frame did not save the PC
```

Let me know if I can provide any more info. Thanks!

History

#1 - 09/25/2020 07:13 PM - dalias (Rich Felker)

From my reading of the test source, it seems to be testing the ability to do various async-signal-unsafe things after fork (but without execve) from a multithreaded parent, which has undefined behavior. Does Ruby (and Ruby programs) actually depend on the ability to do this, or is it just a gratuitous test for something that's not supposed to be supported usage. It looks dubious to me that Ruby even exposes a fork operation to the program it's executing; it's really hard to get forking of an interpreter right, and basically impossible if the interpreter process is allowed to be multi-threaded.