Ruby - Bug #17656

Improper functions shown in C level backtrace information

02/24/2021 11:23 PM - xtkoba (Tee KOBAYASHI)

Status:	Feedback		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:		**Backport:**	2.5: UNKNOWN, 2.6: UNKNOWN, 2.7: UNKNOWN, 3.0: UNKNOWN

Description

The following is an example of C backtrace output on aarch64-linux, where sig_do_nothing is shown in spite of segfault:

```
-- C level backtrace information -------------------------------------------
/var/tmp/build.debug/aarch64.clang.O0/lib/libruby.so.3.0(rb_print_backtrace+0x24) [0x5500b86c74] vm_dump.c:758
/var/tmp/build.debug/aarch64.clang.O0/lib/libruby.so.3.0(rb_vm_bugreport+0xa8) [0x5500b86d38] vm_dump.c:999
/var/tmp/build.debug/aarch64.clang.O0/lib/libruby.so.3.0(rb_bug_for_fatal_signal+0x108) [0x550092fb7c] error.c:786
/var/tmp/build.debug/aarch64.clang.O0/lib/libruby.so.3.0(sig_do_nothing+0x0) [0x5500abfa90] signal.c:960
/var/tmp/build.debug/aarch64.clang.O0/lib/libruby.so.3.0(sigsegv) (null):0
[0x50c810]
(...)
```

This happens because backtrace(3) fills the buffer with the return addresses retrieved from stack frames. A workaround is to subtract 1 from each return address [1], as in the attached patch.

[1] http://lists.dwarfstd.org/pipermail/dwarf-discuss-dwarfstd.org/2020-July/004694.html

History

#1 - 03/05/2021 02:53 PM - mame (Yusuke Endoh)

Thank you @xtkoba .

Could you add a comment to the #ifdef hack?

And if possible, can you send a PR to github.com/ruby/ruby? It would be useful to check if it works on the CI before I merge it.

#2 - 06/09/2022 07:48 AM - mame (Yusuke Endoh)

- Status changed from Open to Feedback

I couldn't reproduce the issue on AWS Graviton2 instance. Could you elaborate how to reproduce the issue?

```
$ uname -a
Linux rubyci-ubuntu2004-arm 5.13.0-1022-aws #24~20.04.1-Ubuntu SMP Thu Apr 7 22:14:11 UTC 2022 aarch64 aarch64
 aarch64 GNU/Linux
```

Here is ruby complied by gcc version 9.4.0:

```
$ ./miniruby -e 'Process.kill :SEGV, $$'
-e:1: [BUG] Segmentation fault at 0x000003f0001c6a1f
ruby 3.2.0dev (2022-06-09T05:39:18Z master 90b240d127) [aarch64-linux]

-- Control frame information -----------------------------------------------
c:0003 p:---- s:0012 e:000011 CFUNC  :kill
c:0002 p:0015 s:0006 e:000005 EVAL   -e:1 [FINISH]
c:0001 p:0000 s:0003 E:001de0 (none) [FINISH]

-- Ruby level backtrace information ----------------------------------------
-e:1:in `<main>'
-e:1:in `kill'
```

```
-- Machine register context ------------------------------------------------
  x0: 0x0000000000000000  x1: 0x000000000000000b  x2: 0x0000aaaab69a9ed0
  x3: 0x5345475600000000  x4: 0x0000000000004553  x5: 0x5345475600000000
  x6: 0x0000000000000040  x7: 0x0000000000000007 x18: 0x0000000000000000
 x19: 0x0000000000000001 x20: 0x000000000000000b x21: 0x0000000000000002
 x22: 0x0000ffff81c62048 x23: 0x0000000000000800 x24: 0x00000000001c6a1f
 x25: 0x0000000000000001 x26: 0x0000aaaab699b000 x27: 0x0000ffffc01b7c20
 x28: 0x00000000001c6a1f x29: 0x0000ffffc01b7bb0  sp: 0x0000ffffc01b7bb0
 fau: 0x0000000000000000

-- C level backtrace information -------------------------------------------
/home/mame/ruby/miniruby(rb_vm_bugreport+0x660) [0xaaaab6868408] vm_dump.c:762
/home/mame/ruby/miniruby(rb_bug_for_fatal_signal+0xd0) [0xaaaab66728c8] error.c:822
/home/mame/ruby/miniruby(sigsegv+0x58) [0xaaaab67c08d0] signal.c:964
/home/mame/ruby/miniruby(sigill) (null):0
linux-vdso.so.1(__kernel_rt_sigreturn+0x0) [0xffff8239b78c]
[0xffff820b10b8]
/home/mame/ruby/miniruby(rb_f_kill+0x2c8) [0xaaaab67c1d68] signal.c:481
...
```

Here is ruby complied by clang version 10.0.0-4ubuntu1:

```
$ ./miniruby -e 'Process.kill :SEGV, $$'
-e:1: [BUG] Segmentation fault at 0x000003f0001cd023
ruby 3.2.0dev (2022-06-09T05:39:18Z master 90b240d127) [aarch64-linux]

-- Control frame information -----------------------------------------------
c:0003 p:---- s:0012 e:000011 CFUNC  :kill
c:0002 p:0015 s:0006 e:000005 EVAL   -e:1 [FINISH]
c:0001 p:0000 s:0003 E:002460 (none) [FINISH]

-- Ruby level backtrace information ----------------------------------------
-e:1:in `<main>'
-e:1:in `kill'

-- Machine register context ------------------------------------------------
  x0: 0x0000000000000000  x1: 0x000000000000000b  x2: 0x0000000000000000
  x3: 0x5345475600000000  x4: 0x0000000000000028  x5: 0x5345475600000000
  x6: 0x0000000000000040  x7: 0x0000000000000007 x18: 0x0000000000000000
 x19: 0x000000000000000b x20: 0x0000ffff9256b048 x21: 0x00000000001cd023
 x22: 0x0000000000000002 x23: 0x0000000000000001 x24: 0x0000000000000001
 x25: 0x0000000000000001 x26: 0x0000aaaade9f08d4 x27: 0x0000aaaade9f07fc
 x28: 0x0000000000080b90 x29: 0x0000ffffe4017210  sp: 0x0000ffffe4017150
 fau: 0x0000000000000000

-- C level backtrace information -------------------------------------------
/home/mame/ruby/miniruby(rb_print_backtrace+0x14) [0xaaaade918f70] vm_dump.c:762
/home/mame/ruby/miniruby(rb_vm_bugreport) vm_dump.c:1057

-- Other runtime information -----------------------------------------------
...
```

Looks like vm_dump does not work at all on aarch64 with clang. The proposed patch didn't change it. This might be another issue.

#3 - 06/09/2022 08:40 AM - mame (Yusuke Endoh)

I could reproduce it with clang 12.

Could you explaiun the rationale of traces[i] = (void *)(((uintptr_t)traces[i] & (~1)) - 1);? The message you referred says "subtracting 1 from the return address, although not guaranteed to provide the exact calling address, generally will produce an address within the same context as the calling address, and that usually is sufficient." but I have no idea why & (~1) is needed. Is this a common knowledge about 32-bit arm?

Files

ruby-backtrace-address-off-by-1.patch	470 Bytes	02/24/2021	xtkoba (Tee KOBAYASHI)