Ruby - Bug #14480

miniruby crashing when compiled with -O2 or -O1 on aarch64

02/16/2018 08:54 AM - vo.x (Vit Ondruch)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.5.0p0 (2017-12-25 revision 61468) [aarch64-linux]	**Backport:**	2.3: UNKNOWN, 2.4: UNKNOWN, 2.5: UNKNOWN

Description

Recently, it is not possible to build Ruby 2.5.0 on aarch64 on Fedora Rawhide, because miniruby fails during build:

```
... snip ...

./miniruby -I./lib -I. -I.ext/common  -n \
-e 'BEGIN{version=ARGV.shift;mis=ARGV.dup}' \
-e 'END{abort "UNICODE version mismatch: #{mis}" unless mis.empty?}' \
-e '(mis.delete(ARGF.path); ARGF.close) if /ONIG_UNICODE_VERSION_STRING +"#{Regexp.quote(version)}
"/o' \
10.0.0 ./enc/unicode/10.0.0/casefold.h ./enc/unicode/10.0.0/name2ctype.h
generating encdb.h
./miniruby -I./lib -I. -I.ext/common  ./tool/generic_erb.rb -c -o encdb.h ./template/encdb.h.tmpl
./enc enc
generating prelude.c
./miniruby -I./lib -I. -I.ext/common  ./tool/generic_erb.rb -I. -c -o prelude.c \
 ./template/prelude.c.tmpl ./prelude.rb ./gem_prelude.rb ./abrt_prelude.rb
*** stack smashing detected ***: <unknown> terminated
encdb.h updated

... snip ...
```

This might by Ruby or gcc issue. Not sure yet. However, there is already lengthy analysis available in Fedora's Bugzilla 1. Would be anybody able to help to resolve this issue?

Related issues:

Related to Ruby - Bug #5407: Cannot build ruby-1.9.3-rc1 with TDM-GCC 4.6.1 o...	**Closed**	**10/05/2011**
Related to Ruby - Bug #9710: __builtin_setjmp/longjmp causes SEGV with mingw	**Closed**	**04/07/2014**
Related to Ruby - Bug #17511: Segmentation fault when compiled with -O2 or hi...	**Closed**	

Associated revisions

Revision 029d92b8988d26955d0622f0cbb8ef3213200749 - 06/06/2024 10:46 PM - jeremyevans (Jeremy Evans)

Disable __builtin_setjmp usage on linux-aarch64

It is questionable whether __builtin_setjmp should default to yes
at all, but since it appears to still have problems on this platform,
it seems safest to disable it.

Fixes [Bug #14480]

Revision 029d92b8988d26955d0622f0cbb8ef3213200749 - 06/06/2024 10:46 PM - jeremyevans (Jeremy Evans)

Disable __builtin_setjmp usage on linux-aarch64

It is questionable whether __builtin_setjmp should default to yes
at all, but since it appears to still have problems on this platform,
it seems safest to disable it.

Fixes [Bug #14480]

Revision 029d92b8 - 06/06/2024 10:46 PM - jeremyevans (Jeremy Evans)

Disable __builtin_setjmp usage on linux-aarch64

It is questionable whether __builtin_setjmp should default to yes
at all, but since it appears to still have problems on this platform,
it seems safest to disable it.

Fixes [Bug #14480]

History

#1 - 02/19/2018 12:00 PM - wanabe (_ wanabe)

- Related to Bug #5407: Cannot build ruby-1.9.3-rc1 with TDM-GCC 4.6.1 on Windows XP SP3 added

#2 - 02/21/2018 12:51 AM - wanabe (_ wanabe)

- File Dockerfile added

I confirmed it can be reproduced with docker + qemu-aarch64-static + binfmt.
Dockerfile is attached.
(Note that it requires a copy of your /usr/bin/qemu-aarch64-static in working directory)

I also confirmed the issue is avoidable with -fno-omit-frame-pointer workaround as commented on
https://bugzilla.redhat.com/show_bug.cgi?id=1545239#c19.
But I expect gcc experts will resolve this. See https://bugzilla.redhat.com/show_bug.cgi?id=1545239#c26.

#3 - 02/21/2018 08:10 PM - vo.x (Vit Ondruch)

It seems they are getting further:

```
With -fomit-frame-pointer on *everything*, and hacking out the call to rb_thread_create_timer_thread in Init_T
hread (to keep this single-threaded for simplicity), the bug appears to be a problem with setjmp/longjmp.

x29 (the frame pointer) is corrupted deep within the 15th call to vm_exec_core within the 10th call to vm_call
_opt_call.

The write of the bogus value to x29 occurs here:

0x000000000057d9bc in rb_ec_tag_jump (ec=0x46b050 <all_iter_i>, st=RUBY_TAG_NONE) at vm.i:10459
10459     __builtin_longjmp(((ec->tag->buf)), (1));
2: /x $x29 = 0x7fffff8080

(gdb) disassemble
Dump of assembler code for function rb_ec_tag_jump:
   0x000000000057d988 <+0>:     stp     x29, x30, [sp,#-32]!
   0x000000000057d98c <+4>:     mov     x29, sp
   0x000000000057d990 <+8>:     str     x0, [x29,#24]
   0x000000000057d994 <+12>:    str     w1, [x29,#20]
   0x000000000057d998 <+16>:    ldr     x0, [x29,#24]
   0x000000000057d99c <+20>:    ldr     x0, [x0,#24]
   0x000000000057d9a0 <+24>:    ldr     w1, [x29,#20]
   0x000000000057d9a4 <+28>:    str     w1, [x0,#336]
   0x000000000057d9a8 <+32>:    ldr     x0, [x29,#24]
   0x000000000057d9ac <+36>:    ldr     x0, [x0,#24]
   0x000000000057d9b0 <+40>:    add     x0, x0, #0x10
   0x000000000057d9b4 <+44>:    ldr     x1, [x0,#8]
   0x000000000057d9b8 <+48>:    ldr     x29, [x0]
=> 0x000000000057d9bc <+52>:    ldr     x0, [x0,#16]
   0x000000000057d9c0 <+56>:    mov     sp, x0
   0x000000000057d9c4 <+60>:    br      x1

when called from rb_iterate0, where the bogus x29 value has been fetched from the jmp_buf at +48.

A watchpoint on that memory shows it being set to the bogus value here in rb_iterate0:

   0x0000000000595e50 <+96>:    str     x0, [sp,#264]
   0x0000000000595e54 <+100>:   ldr     x0, [sp,#232]
   0x0000000000595e58 <+104>:   ldr     x0, [x0,#24]
   0x0000000000595e5c <+108>:   str     x0, [sp,#592]
   0x0000000000595e60 <+112>:   add     x0, sp, #0x108
   0x0000000000595e64 <+116>:   add     x0, x0, #0x10
   0x0000000000595e68 <+120>:   add     x1, sp, #0x260
   0x0000000000595e6c <+124>:   str     x1, [x0]
=> 0x0000000000595e70 <+128>:   adrp    x1, 0x595000 <raise_method_missing+544>

28300        struct rb_vm_tag _tag;
28301        _tag.state = RUBY_TAG_NONE;
```

```
        _tag.tag = ((VALUE)RUBY_Qundef);
        _tag.prev = _ec->tag;
        ;
        state = (__builtin_setjmp((_tag.buf)) ? rb_ec_tag_state((_ec))
                                              : ((void)(_ec->tag = &_tag), 0));
```

```
If I'm reading this right, the __builtin_longjmp rb_ec_tag_jump (in rb_iterate0) is attempting to restore x29
from the jmp_buf, but the __builtin_setjmp in rb_iterate0 isn't actually saving x29 there, and hence x29 gets
corrupted at the longjmp, deep in the callstack, leading to an eventual crash when vm_call_opt_call eventually
 tries to use x29.
```

#4 - 02/23/2018 08:24 AM - vo.x (Vit Ondruch)

This appears to be longstanding GCC issue on aarch64, which was recently exposed by change in GCC defaults 1. However, the GCC upstream is questioning usage of __builtin_setjmp/__builtin_longjmp. Can somebody provide answer to this question 2:

> I notice that in our builds, RUBY_SETJMP is using "__builtin_setjmp", rather
> than "setjmp".
>
> This seems to come from the "configure" check; the rpm build log has:
> "checking for setjmp type... __builtin_setjmp"
>
> Is this the default for upstream Ruby?
>
> A GCC upstream developer notes:
>
> > To me any use of __builtin_setjmp/__builtin_longjmp is almost always incorrect.
> > (https://gcc.gnu.org/bugzilla/show_bug.cgi?id=84521#c3)

#5 - 02/23/2018 08:52 AM - wanabe (_ wanabe)

- Related to Bug #9710: __builtin_setjmp/longjmp causes SEGV with mingw added

#6 - 02/26/2018 12:38 AM - wanabe (_ wanabe)

tl;dr:

- rb_ec_tag_jump() causes SEGV on aarch64 when it is built with -fomit-frame-pointer optimization that is the default of -O1 of gcc-8.0.1-0.13.
  - It is confirmed that -fno-omit-frame-pointer optflags can avoid the issue.
  - Also --with-setjmp-type=setjmp will do, but no one has yet confirmed.
- Andrew Pinski, one of gcc maintainer, says "To me any use of __builtin_setjmp/__builtin_longjmp is almost always incorrect."

vo.x (Vit Ondruch) wrote:

> This appears to be longstanding GCC issue on aarch64, which was recently exposed by change in GCC defaults [1]. However, the GCC upstream is questioning usage of __builtin_setjmp/__builtin_longjmp. Can somebody provide answer to this question [2]:
>
> > I notice that in our builds, RUBY_SETJMP is using "__builtin_setjmp", rather
> > than "setjmp".
> >
> > This seems to come from the "configure" check; the rpm build log has:
> > "checking for setjmp type... __builtin_setjmp"
> >
> > Is this the default for upstream Ruby?

Yes, it is, as far as I know since r15871 ruby-core:16086.

Nakada-san, how do you think about this issue?

#7 - 02/26/2018 12:36 PM - nobu (Nobuyoshi Nakada)

It sounds good to fix by --with-setjmp-type, if it works.

#8 - 02/26/2018 09:58 PM - vo.x (Vit Ondruch)

I tried --with-setjmp-type=setjmp and the build passed. But I have no idea what is the performance impact.

Nevertheless, there are two more comments from GCC people. The first is 1:

> Jeff Law 2018-02-26 17:01:13 CET

> In general I would not expect applications to make direct use of the builtin setjmp/longjmp routines within GCC. They should instead be using the OS provided setjmp/longjmp.
>
> The builtin setjmp/longjmp are primarily for the use of the exception handling system on a small number of targets that do not have sufficient unwinding mechansisms.

Based on this, I'd say you should reconsider change of the defaults.

On the other hand, GCC people already reverted the change responsible for the issue [2](#):

> Dave Malcolm 2018-02-26 17:30:10 CET
>
> The bug was worked around upstream today by commit r257984 (by changing the default on aarch64 back to -fno-omit-frame-pointer i.e. keep the frame pointer):
> https://gcc.gnu.org/viewcvs/gcc?view=revision&revision=257984

But they will continue to discuss what is the right solution here ...

#9 - 02/27/2018 07:40 AM - nobu (Nobuyoshi Nakada)

vo.x (Vit Ondruch) wrote:

> I tried --with-setjmp-type=setjmp and the build passed. But I have no idea what is the performance impact.

Thank you for checking.

> Nevertheless, there are two more comments from GCC people. The first is [1]:
>
> > Jeff Law 2018-02-26 17:01:13 CET
> >
> > In general I would not expect applications to make direct use of the builtin setjmp/longjmp routines within GCC. They should instead be using the OS provided setjmp/longjmp.
> >
> > The builtin setjmp/longjmp are primarily for the use of the exception handling system on a small number of targets that do not have sufficient unwinding mechansisms.

Yes, they are used for the exception handling in Ruby :)

#10 - 04/03/2018 11:29 PM - wanabe (_ wanabe)

- Status changed from Open to Third Party's Issue

GCC upstream changed aarch64 default behaviour on its revision 257984.
https://gcc.gnu.org/viewcvs/gcc/trunk/gcc/common/config/aarch64/aarch64-common.c?r1=257984&r2=257983&pathrev=257984

Fundamental frame pointer corruption may be fixed some day.
https://gcc.gnu.org/ml/gcc-patches/2018-03/msg00668.html

So ruby don't have to implement a workaround about the issue.
Even if someone encounters this, updating gcc latest or -fno-omit-frame-pointer or --with-setjmp-type=setjmp may help.

Thanks.

#11 - 04/09/2018 08:41 AM - vo.x (Vit Ondruch)

Just forwarding one remark from RH Bugzilla [0](#):

```
--- Comment #44 from Dave Malcolm <dmalcolm@redhat.com> ---
> should we keep the "setjmp" for AArch64? I am asking,
> since it seems that Ruby upstream is not going to change anything [1], so we
> should somehow officially resolve/close the issue.
>
>
>
> [1] https://bugs.ruby-lang.org/issues/14480#note-10

Note that the workaround in the gcc rpm is papering over the issue, albeit a
long-standing one: that __builtin_setjmp on aarch64 doesn't properly save the
frame pointer, leading to clobbering of the frame pointer when
__builtin_longjmp is used, hence leading to issues when used in conjunction
```

```
with -fomit-frame-pointer.

If upstream Ruby want to use __builtin_setjmp as a performance optimization,
that's up to them, I guess, but it's relying on none of the code ever using or
interacting with -fomit-frame-pointer (until PR target/84521 is properly
fixed).
```

As for Ruby on AArch64 Fedora, I'll keep using "setjmp" unless somebody has some convincing arguments :)

#12 - 04/10/2018 12:04 AM - wanabe (_ wanabe)

- Status changed from Third Party's Issue to Open

Sorry for the confusion, I am not one of "upstream".
So I revert the issue status and there are no upstream opinion.
I am sorry again.

#13 - 01/05/2021 01:24 AM - mame (Yusuke Endoh)

- Related to Bug #17511: Segmentation fault when compiled with -O2 or higher on ARM Android added

#14 - 01/05/2021 04:39 AM - mame (Yusuke Endoh)

Looks like the status of __builtin_setjmp is very complicated:

The documentation of GCC says:

> GCC provides the built-in functions __builtin_setjmp and __builtin_longjmp which are similar to, but not interchangeable with, the C library functions setjmp and longjmp. The built-in versions are used internally by GCC's libraries to implement exception handling on some targets. You should use the standard C library functions declared in <setjmp.h> in user code instead of the builtins.
>
> An important caveat is that GCC arranges to save and restore only those registers known to the specific architecture variant being compiled for. This can make __builtin_setjmp and __builtin_longjmp more efficient than their library counterparts in some cases, but it can also cause incorrect and mysterious behavior when mixing with code that uses the full register set.

So, in general, it looks good to avoid using __builtin_setjmp/__builtin_longjmp in Ruby.

It was introduced for a performance reason on Mac OS X in 2008.

http://blade.nagaokaut.ac.jp/cgi-bin/vframe.rb/ruby/ruby-core/16023?15980-16545+split-mode-vertical

I'm unsure if this performance improvement on macOS is still significant. Anyone is interested in measuring the peformance?

Actually, using __builtin_setjmp has been very troublesome. It is explicitly disabled on some architectures/conditions:

- AIX: d612c44dad8a5d4c373b0c4bc08a99e3dc06498e
- ppc64* LInux: 67fcbf9328614379fee28bed83ae6749ce7a3dda
- when -fcf-protection is used: #15307
- GCC 4 or before: #10272

And looks like we need to disable it on Android too. #17511

But interestingly, it is explicitly enabled on mingw64 because it was apparently needed to fix an issue: #15348.

#15 - 02/23/2021 10:35 AM - xtkoba (Tee KOBAYASHI)

The main issue might be resolved by the following commit to GCC:
https://gcc.gnu.org/git/?p=gcc.git&a=commit;h=25403c416e5f12d681d1fc45a8789d19ab40297f
(see also https://gcc.gnu.org/bugzilla/show_bug.cgi?id=84521#c29)

IMHO, the use of __builtin_{set,long}jmp should be opt-in. They are probably safe to use in recent GCC versions, but may not in Clang/LLVM (or other compilers than GCC).

#16 - 08/23/2023 11:23 PM - jeremyevans0 (Jeremy Evans)

I submitted a pull request to disable __builtin_setjmp for Linux aarch64: https://github.com/ruby/ruby/pull/8272

#17 - 09/22/2023 09:49 AM - vo.x (Vit Ondruch)

Please note that on Fedora this was not issue for past ~5 years. This has beend workarounded on GCC side:

https://bugzilla.redhat.com/show_bug.cgi?id=1545239#c42

And it seems that the GCC folks position is that this should behave consistently everywhere. So I am not sure why the PR above extends the special casing of some platforms instead of removing it.

#18 - 06/06/2024 10:46 PM - jeremyevans (Jeremy Evans)

- Status changed from Open to Closed

Applied in changeset git|029d92b8988d26955d0622f0cbb8ef3213200749.

Disable __builtin_setjmp usage on linux-aarch64

It is questionable whether __builtin_setjmp should default to yes
at all, but since it appears to still have problems on this platform,
it seems safest to disable it.

Fixes [Bug #14480]

#19 - 06/07/2024 08:34 AM - vo.x (Vit Ondruch)

I don't understand why this change was applied. This should be either enabled everywhere or disabled everywhere, not enabled on some random platforms. Please note that this should not be issue on aarch64 for a while.

Or was the issue exhibited somewhere recently? The commit does not elaborate about it.

#20 - 01/30/2025 12:15 AM - thesamesam (Sam James)

vo.x (Vit Ondruch) wrote in #note-19:

> I don't understand why this change was applied. This should be either enabled everywhere or disabled everywhere, not enabled on some random platforms. Please note that this should not be issue on aarch64 for a while.
>
> Or was the issue exhibited somewhere recently? The commit does not elaborate about it.

We had a report of this in Gentoo today on amd64: https://bugs.gentoo.org/949016. Using setjmp worked. Having read over the discussions with the GCC developers, I don't think __builtin_setjmp should ever be used if setjmp is available.

Files

Dockerfile	573 Bytes	02/21/2018	wanabe (_ wanabe)