Ruby 1.8 - Bug #4277

Compiler optimizations may lead to premature GC in C code

01/14/2011 11:20 AM - qz (Quet Zal)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	
ruby -v:	ruby 1.8.7 (2010-12-23 patchlevel 330) [x86_64-linux]

Description

=begin
It is several months as I had problems with GzipReader that comes with ruby. When processing large amount of files (300k+) there were always some files that threw exceptions, but they looked perfectly good and unpacked with gunzip just fine.
Today ruby just segfaulted and that made me to do some investigations....

Let's start with problem:

Program received signal SIGSEGV, Segmentation fault.
0x00007ffff6f3f1fb in memcpy () from /lib/libc.so.6
(gdb) bt
#0 0x00007ffff6f3f1fb in memcpy () from /lib/libc.so.6
#1 0x00007ffff7b7a131 in str_buf_cat (str=140737317158040, ptr=0xee7430 , len=2048)
at /usr/include/bits/string3.h:52
#2 0x00007ffff59a65bc in zstream_append_input (z=0x707dc0, src=0xee7430 , len=) at zlib.c:604
#3 0x00007ffff59a6600 in gzfile_read_raw_ensure (gz=0x707dc0, size=10) at zlib.c:1879
#4 0x00007ffff59a7bb6 in gzfile_read_header (obj=140737317158120, io=140737317158160) at zlib.c:2004
#5 rb_gzreader_initialize (obj=140737317158120, io=140737317158160) at zlib.c:2941
#6 0x00007ffff7b1a21e in rb_call0 (klass=140737316755120, recv=140737317158120, id=2961, oid=2961, argc=1,
argv=0x7fffffffb360,
body=0x7ffff5c58aa8, flags=) at eval.c:5928
#7 0x00007ffff7b1a3d6 in rb_call (klass=140737316755120, recv=140737317158120, mid=2961, argc=1, argv=0x7fffffffb360,
scope=1, self=6)
at eval.c:6176
#8 0x00007ffff7b1ad4b in rb_funcall2 (recv=, mid=, argc=, argv=)
at eval.c:6312
#9 0x00007ffff7b1ade4 in rb_obj_call_init (obj=140737317158120, argc=1, argv=0x7fffffffb360) at eval.c:7825
#10 0x00007ffff7b472d2 in rb_class_new_instance (argc=1, argv=0x7fffffffb360, klass=) at object.c:1644
#11 0x00007ffff59a9436 in rb_gzfile_s_wrap (argc=7294720, argv=0xee7430, klass=2048) at zlib.c:2347
#12 0x00007ffff59a94d6 in gzfile_s_open (argc=1, argv=0x7fffffffb360, klass=140737316755120, mode=0x7ffff59aa306 "rb") at
zlib.c:2377
#13 0x00007ffff7b1a21e in rb_call0 (klass=140737316755080, recv=140737316755120, id=7681, oid=7681, argc=1,
argv=0x7fffffffb360,
body=0x7ffff5c58b48, flags=) at eval.c:5928
#14 0x00007ffff7b1a3d6 in rb_call (klass=140737316755080, recv=140737316755120, mid=7681, argc=1, argv=0x7fffffffb360,
scope=0, self=140737354023600)
at eval.c:6176
#15 0x00007ffff7b145e0 in rb_eval (self=140737354023600, n=) at eval.c:3506
#16 0x00007ffff7b17c0c in rb_eval (self=140737354023600, n=) at eval.c:3236
#17 0x00007ffff7b175e1 in rb_eval (self=140737354023600, n=) at eval.c:3322
#18 0x00007ffff7b1a130 in rb_call0 (klass=140737354033760, recv=140737354023600, id=10921, oid=10921, argc=0,
argv=0x7fffffffc130,
body=0x7ffff7fc1d50, flags=) at eval.c:6079
#19 0x00007ffff7b1a3d6 in rb_call (klass=140737354033760, recv=140737354023600, mid=10921, argc=3, argv=0x7fffffffc130,
scope=1,
self=140737354023600) at eval.c:6176
#20 0x00007ffff7b1470e in rb_eval (self=140737354023600, n=) at eval.c:3521
#21 0x00007ffff7b159e8 in rb_eval (self=140737354023600, n=) at eval.c:3701
#22 0x00007ffff7b1864c in rb_yield_0 (val=140737317162800, self=140737354023600, klass=0, flags=, avalue=0) at eval.c:5095
#23 0x00007ffff6892a78 in each_hash (argc=0, argv=0x0, obj=140737316750640) at mysql.c:1174
#24 0x00007ffff7b1a21e in rb_call0 (klass=140737353547960, recv=140737316750640, id=11089, oid=11089, argc=0, argv=0x0,
body=0x7ffff7f6e0d8,

```
flags=) at eval.c:5928
#25 0x00007ffff7b1a3d6 in rb_call (klass=140737353547960, recv=140737316750640, mid=11089, argc=0, argv=0x0, scope=0,
self=140737354023600)
at eval.c:6176
#26 0x00007ffff7b145e0 in rb_eval (self=140737354023600, n=) at eval.c:3506
#27 0x00007ffff7b17c0c in rb_eval (self=140737354023600, n=) at eval.c:3236
#28 0x00007ffff7b26dc5 in ruby_exec_internal () at eval.c:1654
#29 0x00007ffff7b26e05 in ruby_exec () at eval.c:1674
#30 0x00007ffff7b26e30 in ruby_run () at eval.c:1684
#31 0x0000000000400943 in main (argc=2, argv=0x7fffffffd7d8, envp=) at main.c:48
(gdb)

(gdb) frame 3
#3  0x00007ffff59a6600 in gzfile_read_raw_ensure (gz=0x707dc0, size=10) at zlib.c:1879
1879            zstream_append_input2(&gz->z, str);
(gdb) list
1874        VALUE str;
1875
1876        while (NIL_P(gz->z.input) || RSTRING(gz->z.input)->len < size) {
1877            str = gzfile_read_raw(gz);
1878            if (NIL_P(str)) return Qfalse;
1879            zstream_append_input2(&gz->z, str);
1880        }
1881        return Qtrue;
1882    }
1883
(gdb)

(gdb) frame 2
#2  0x00007ffff59a65bc in zstream_append_input (z=0x707dc0, src=0xee7430 , len=) at zlib.c:604
604             rb_str_buf_cat(z->input, src, len);
(gdb) list
599     {
600         if (len <= 0) return;
601
602         if (NIL_P(z->input)) {
603             z->input = rb_str_buf_new(len);
604             rb_str_buf_cat(z->input, src, len);
605             RBASIC(z->input)->klass = 0;
606         }
607         else {
608             rb_str_buf_cat(z->input, src, len);
(gdb)
```

What happens here is that str being passed to zstream_append_input, but is destroyed on its way to rb_str_buf_cat. It happens during GC that is triggered by rb_str_buf_new.
Ruby's GC scans memory (and maybe also checks registers?) for values that look like VALUEs (oh, god). And it can't seem to find one in this case. Let's see why...

```
text:0000000000003697                mov     rdi, rbx
.text:000000000000369A                call    gzfile_read_raw
.text:000000000000369F                cmp     rax, 4
.text:00000000000036A3                jnz     short loc_36A9
.text:00000000000036A5                mov     al, 0
.text:00000000000036A7                jmp     short loc_36CE
.text:00000000000036A9 ; ---------------------------------------------------------------------------
.text:00000000000036A9
.text:00000000000036A9 loc_36A9:                              ; CODE XREF: gzfile_read_raw_ensure+1Bj
.text:00000000000036A9                mov     rdx, [rax+10h]
.text:00000000000036AD                mov     rsi, [rax+18h]
.text:00000000000036B1                mov     rdi, r12
.text:00000000000036B4                call    zstream_append_input
```

VALUE is in rax, pointer and length get extracted to rdx and rsi, then rax is most likely overwritten as its 'scrap' register.
So when rb_str_buf_new is called (which in turn calls ruby_xmalloc, which in turn calls garbage_collect) original VALUE is nowhere to be found. Compiler was clever enough (sadly, for us) to optimize all memory stores.
We can force compiler to store that value in memory by declaring str in gzfile_read_raw_ensure as volatile. This indeed fixes the

problem.

This bug report goes without code to reproduce, because I'm not allowed to send our application and isolating proof-of-concept code is not too easy, but it must be clear what's going on and how to fix it.

I really hope that someone thinks of a way to check rest of ruby extensions' code for such problems.

Thanks for reading this wall of text ;)
=end