Ruby master - Bug #8364

Vim's if_ruby feature sometimes crash.

05/03/2013 08:50 PM - Anonymous

Status:	Third Party's Issue		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.0.0p0 (2013-02-24) [x86_64-darwin11.4.2]	**Backport:**	1.9.3: UNKNOWN, 2.0.0: UNKNOWN

Description

=begin
This problem was originally reported to vim-jp project. See
((URL:https://github.com/vim-jp/issues/issues/372)) (Japanese discussion).

(()) is an text editor and it supports embedded
ruby interface so that we can customize Vim with Ruby. On Mac, ruby
causes crash.

Steps to reproduce:
$ vim
:function! F()
: ruby nil
:endfunction
:call F()
:ruby 10000.times { 'x' * 10000 }
Vim: Caught deadly signal ABRT
Vim: Finished.

Version is:
Mac OS X 10.7.5
ruby-2.0.0-p0
Vim 7.3.918

Maybe the following is minimum C code to reproduce crash.

```
/* rubyembed.c */
#include

void rbinit() {
int dummy_argc = 2;
char *dummy_argv[] = {"foo", "-e0"};
RUBY_INIT_STACK;
ruby_init();
ruby_process_options(dummy_argc, dummy_argv);
}

void rbexe(const char *src) {
int state;
rb_eval_string_protect(src, &state);
if (state)
ruby_error_print();
}

void init() {
int eatstack[1024]; /* perhaps you need more to crash */
rbinit();
}

int main(int argc, char **argv) {
init();
rbexe("10000.times { 'x' * 10000 }");
```

```
return 0;
}

$ cd ~/tmp/ruby-2.0.0-p0
$ ./configure --prefix=$HOME/tmp/opt && make install
$ cc rubyembed.c -I$HOME/tmp/opt/include/ruby-2.0.0 -I$HOME/tmp/opt/include/ruby-2.0.0/x86_64-darwin11.4.2
-L$HOME/tmp/opt/lib -lruby-static
$ ./a.out
eval:1: [BUG] gc_sweep(): unknown data type 0x0(0x007fb2590651e0) 0x102000
ruby 2.0.0p0 (2013-02-24) [x86_64-darwin11.4.2]

-- Crash Report log information --------------------------------------------
See Crash Report log file under the one of following:
* ~/Library/Logs/CrashReporter
* /Library/Logs/CrashReporter
* ~/Library/Logs/DiagnosticReports
* /Library/Logs/DiagnosticReports
the more detail of.

-- Control frame information -----------------------------------------------
eval:1: [BUG] object allocation during garbage collection phase
ruby 2.0.0p0 (2013-02-24) [x86_64-darwin11.4.2]

-- Crash Report log information --------------------------------------------
See Crash Report log file under the one of following:
* ~/Library/Logs/CrashReporter
* /Library/Logs/CrashReporter
* ~/Library/Logs/DiagnosticReports
* /Library/Logs/DiagnosticReports
the more detail of.

-- Control frame information -----------------------------------------------
c:0005 p:---- s:0013 e:000012 CFUNC  :*
c:0004 p:0010 s:0009 e:000008 BLOCK  eval:1 [FINISH]
c:0003 p:---- s:0007 e:000006 CFUNC  :times
c:0002 p:0006 s:0004 e:000003 EVAL   eval:1 [FINISH]
c:0001 p:0000 s:0002 E:000d48 TOP    [FINISH]

eval:1:in <main>'
eval:1:intimes'
eval:1:in block in <main>'
eval:1:in*'

-- C level backtrace information -------------------------------------------
0   a.out                    0x000000010dc0b02b rb_vm_bugreport + 251
1   a.out                    0x000000010daa7af8 report_bug + 392
2   a.out                    0x000000010daa7dff rb_bug + 207
3   a.out                    0x000000010dac89f9 newobj + 297
4   a.out                    0x000000010dac910f rb_newobj_of + 31
5   a.out                    0x000000010db942a4 str_new + 68
6   a.out                    0x000000010db957ee rb_usascii_str_new + 30
7   a.out                    0x000000010db13aeb rb_id2str + 107
8   a.out                    0x000000010db13cb9 rb_id2name + 9
9   a.out                    0x000000010dc0adbb control_frame_dump + 1035
10  a.out                     0x000000010dc0afcb rb_vm_bugreport + 155
11  a.out                     0x000000010daa7af8 report_bug + 392
12  a.out                     0x000000010daa7dff rb_bug + 207
13  a.out                     0x000000010dac4803 slot_sweep + 659
14  a.out                     0x000000010dac7dcf garbage_collect + 623
15  a.out                     0x000000010dac82e5 vm_xmalloc + 149
16  a.out                     0x000000010db943a3 str_new + 323
17  a.out                     0x000000010db96966 rb_str_times + 118
18  a.out                     0x000000010dbecad9 vm_call_cfunc_with_frame + 761
19  a.out                     0x000000010dbf21bd vm_exec_core + 14301
20  a.out                     0x000000010dbf79d1 vm_exec + 2673
21  a.out                     0x000000010dc061fb rb_yield + 507
22  a.out                     0x000000010daf7b1d int_dotimes + 61
```

```
23  a.out                   0x000000010dbecad9 vm_call_cfunc_with_frame + 761
24  a.out                   0x000000010dc04f5c vm_call_method + 828
25  a.out                   0x000000010dbf0b43 vm_exec_core + 8547
26  a.out                   0x000000010dbf79d1 vm_exec + 2673
27  a.out                   0x000000010dbf8305 eval_string_with_cref + 693
28  a.out                   0x000000010daaddd8 rb_protect + 232
29  a.out                   0x000000010da6694f rbexe + 31
30  a.out                   0x000000010da669b5 main + 37
31  a.out                   0x000000010da668c4 start + 52

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

- Loaded script: -e

- Loaded features:

```
  0 enumerator.so
  1 /Users/yukihiro/tmp/opt/lib/ruby/2.0.0/x86_64-darwin11.4.2/enc/encdb.bundle
  2 /Users/yukihiro/tmp/opt/lib/ruby/2.0.0/x86_64-darwin11.4.2/enc/trans/transdb.bundle
  3 /Users/yukihiro/tmp/opt/lib/ruby/2.0.0/x86_64-darwin11.4.2/rbconfig.rb
  4 /Users/yukihiro/tmp/opt/lib/ruby/2.0.0/rubygems/compatibility.rb
  5 /Users/yukihiro/tmp/opt/lib/ruby/2.0.0/rubygems/defaults.rb
  6 /Users/yukihiro/tmp/opt/lib/ruby/2.0.0/rubygems/deprecate.rb
  7 /Users/yukihiro/tmp/opt/lib/ruby/2.0.0/rubygems/errors.rb
  8 /Users/yukihiro/tmp/opt/lib/ruby/2.0.0/rubygems/version.rb
  9 /Users/yukihiro/tmp/opt/lib/ruby/2.0.0/rubygems/requirement.rb
  10 /Users/yukihiro/tmp/opt/lib/ruby/2.0.0/rubygems/platform.rb
  11 /Users/yukihiro/tmp/opt/lib/ruby/2.0.0/rubygems/specification.rb
  12 /Users/yukihiro/tmp/opt/lib/ruby/2.0.0/rubygems/exceptions.rb
  13 /Users/yukihiro/tmp/opt/lib/ruby/2.0.0/rubygems/core_ext/kernel_gem.rb
  14 /Users/yukihiro/tmp/opt/lib/ruby/2.0.0/rubygems/core_ext/kernel_require.rb
  15 /Users/yukihiro/tmp/opt/lib/ruby/2.0.0/rubygems.rb
```

[NOTE]
You may have encountered a bug in the Ruby interpreter or extension libraries.
Bug reports are welcome.
For details: http://www.ruby-lang.org/bugreport.html

I am not sure that this is correct fix but above problem seems
disappeared with this.

```
diff --git a/thread_pthread.c b/thread_pthread.c
index 8953f5e..58b63b8 100644
--- a/thread_pthread.c
+++ b/thread_pthread.c
@@ -618,8 +618,10 @@ ruby_init_stack(volatile VALUE *addr
)
{
native_main_thread.id = pthread_self();
-#ifdef STACK_END_ADDRESS
+#if defined(STACK_END_ADDRESS)
native_main_thread.stack_start = STACK_END_ADDRESS;
+#elif defined(STACKADDR_AVAILABLE)
```

- native_main_thread.stack_start = pthread_get_stackaddr_np(pthread_self()); #else if (!native_main_thread.stack_start ||
  STACK_UPPER((VALUE *)(void *)&addr,

Or perhaps such code is not valid or something is wrong in Vim's if_ruby code?
=end

History

#1 - 05/04/2013 02:10 PM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Third Party's Issue

rbexe() also needs RUBY_INIT_STACK, in every interface function calling ruby from outside.

Your patch is not a portable fix.

You can't use Markdiwn here, RD only.

#2 - 05/04/2013 07:12 PM - Anonymous

Thank you for your advice.
I tried RUBY_INIT_STACK in rbexe(). But it still crash.

```
/* rubyembed.c */
#include

void rbinit() {
int dummy_argc = 2;
char *dummy_argv[] = {"foo", "-e0"};
RUBY_INIT_STACK;
ruby_init();
ruby_process_options(dummy_argc, dummy_argv);
}

void rbexe(const char *src) {
int state;
RUBY_INIT_STACK;
rb_eval_string_protect(src, &state);
if (state)
ruby_error_print();
}

void init() {
int eatstack[1024]; /* perhaps you need more to crash */
rbinit();
}

int main(int argc, char **argv) {
init();
rbexe("10000.times { 'x' * 10000 }");
return 0;
}
```

#3 - 05/05/2013 12:40 PM - nobu (Nobuyoshi Nakada)

=begin
I couldn't link your source because (({ruby_error_print})) is not exported.
It run successfully by replacing the call with (({printf()})).
=end

#4 - 05/05/2013 12:46 PM - nobu (Nobuyoshi Nakada)

- Description updated

#5 - 05/05/2013 01:19 PM - Anonymous

You are right. With -Wall flag, I got warning "implicit declaration of
function 'ruby_error_print'". But it is not related to crash. My
program still doesn't work after removing ruby_error_print() call.

```
/* rubyembed.c */
#include

void rbinit() {
int dummy_argc = 2;
char *dummy_argv[] = {"foo", "-e0"};
RUBY_INIT_STACK;
ruby_init();
ruby_process_options(dummy_argc, dummy_argv);
}

void rbexe(const char *src) {
int state;
RUBY_INIT_STACK;
rb_eval_string_protect(src, &state);
}
```

```
void init() {
int eatstack[1024]; /* perhaps you need more to crash */
rbinit();
}

int main(int argc, char **argv) {
init();
rbexe("10000.times { 'x' * 10000 }");
return 0;
}
```

#6 - 05/05/2013 06:54 PM - h.shirosaki (Hiroshi Shirosaki)

I don't know the rationale, but adding RUBY_INIT_STACK before init() seems to suppress the crash.

```
int main(int argc, char **argv) {
RUBY_INIT_STACK;
init();
rbexe("10000.times { 'x' * 10000 }");
return 0;
}
```

This is a possible fix for Vim. It seems to work on osx. Vim 7.3.923 with ruby trunk.

```
diff --git a/src/if_ruby.c b/src/if_ruby.c
--- a/src/if_ruby.c
+++ b/src/if_ruby.c
@@ -1384,8 +1384,14 @@ static void ruby_vim_init(void)
rb_define_method(cVimWindow, "width", window_width, 0);
rb_define_method(cVimWindow, "width=", window_set_width, 1);
rb_define_method(cVimWindow, "cursor", window_cursor, 0);
rb_define_method(cVimWindow, "cursor=", window_set_cursor, 1);

 rb_define_virtual_variable("$curbuf", buffer_s_current, 0);
 rb_define_virtual_variable("$curwin", window_s_current, 0);

}
+
+void vim_ruby_init(void) {
```

- /* should initialize machine stack early in main function */
- VALUE v;
- ruby_init_stack(&v); +} diff --git a/src/main.c b/src/main.c --- a/src/main.c +++ b/src/main.c @@ -187,16 +187,20 @@ main params.want_full_screen = TRUE; #ifdef FEAT_EVAL params.use_debug_break_level = -1; #endif #ifdef FEAT_WINDOWS params.window_count = -1; #endif

```
+#ifdef FEAT_RUBY
```

- vim_ruby_init(); +#endif + #ifdef FEAT_TCL vim_tcl_init(params.argv[0]); #endif

```
#ifdef MEM_PROFILE
atexit(vim_mem_profile_dump);
#endif

diff --git a/src/proto/if_ruby.pro b/src/proto/if_ruby.pro
--- a/src/proto/if_ruby.pro
+++ b/src/proto/if_ruby.pro
@@ -1,9 +1,10 @@
/* if_ruby.c */
int ruby_enabled __ARGS((int verbose));
void ruby_end __ARGS((void));
void ex_ruby __ARGS((exarg_T *eap));
void ex_rubydo __ARGS((exarg_T *eap));
void ex_rubyfile __ARGS((exarg_T *eap));
void ruby_buffer_free __ARGS((buf_T *buf));
void ruby_window_free __ARGS((win_T *win));
+void vim_ruby_init __ARGS((void));
/* vim: set ft=c : */
```

#7 - 05/06/2013 06:17 PM - Anonymous

Thank you for your advice.
You can close this issue.

#8 - 05/08/2013 04:10 PM - h.shirosaki (Hiroshi Shirosaki)

Maybe I found the reason why RUBY_INIT_STACK in rbexe() doesn't work.

th->machine_stack_start is set from native_main_thread.stack_start in ruby_init().

https://github.com/ruby/ruby/blob/ab750920b9dfdab1117bc39e14bb28195b2b9830/thread_pthread.c#L704
th->machine_stack_start = native_main_thread.stack_start;

RUBY_INIT_STACK itself changes native_main_thread.stack_start, but doesn't change th->machine_stack_start.
th->machine_stack_start is used by GC. So we should specify proper start stack address to include stack region of rbexe() before ruby_init().

I confirmed RUBY_INIT_STACK in rbexe() works as expected if setting th->machine_stack_start in RUBY_INIT_STACK by the following patch.

https://gist.github.com/shirosaki/5536902

#9 - 05/11/2013 04:04 PM - Anonymous

Hiroshi,

Thank you for fixing and reporting to vim_dev.