Backport193 - Backport #5888

JSON unittest fails

01/12/2012 09:12 PM - vo.x (Vit Ondruch)

Status:	Closed	
Priority:	Normal	
Assignee:	naruse (Yui NARUSE)	

Description

Hello,

When building Ruby 1.9.3 (as well as 2.0.0), the make check spits following errors:

```
1) Failure:
test_parse_values(TC_JSON) [/builddir/build/BUILD/ruby-1.9.3-p0/test/json/test_json.rb:174]:
<["\"\b\n\r\t\u0000\u001F"]> expected but was
<["\"\b\n\r\t\xA8\xA8"]>.
2) Failure:
test_parser_reset(TC_JSON) [/builddir/build/BUILD/ruby-1.9.3-p0/test/json/test_json.rb:291]:
<{"a"=>2,
"b"=>3.141,
"c"=>"c",
"d"=>[1, "b", 3.14],
"e"=>{"foo"=>"bar"},
"g"=>"\"\u0000\u001F",
"h"=>1000.0,
"i"=>0.001}> expected but was
<{"a"=>2,
"b"=>3.141,
"c"=>"c",
"d"=>[1, "b", 3.14],
"e"=>{"foo"=>"bar"},
"g"=>"\"\xA0\xA0",
"h"=>1000.0,
"i"=>0.001}>.
3) Failure:
test_fast_generate(TC_JSONGenerate) [/builddir/build/BUILD/ruby-1.9.3-p0/test/json/test_json_generate.rb:78]:
<{"a"=>2,
"b"=>3.141,
"c"=>"c",
"d"=>[1, "b", 3.14],
"e"=>{"foo"=>"bar"},
"g"=>"\"\u0000\u001F",
"h"=>1000.0,
"i"=>0.001}> expected but was
<{"a"=>2,
"b"=>3.141,
"c"=>"c",
"d"=>[1, "b", 3.14],
"e"=>{"foo"=>"bar"},
"g"=>"\"\xA0\xA0",
"h"=>1000.0,
"i"=>0.001}>.
4) Failure:
test_generate(TC_JSONGenerate) [/builddir/build/BUILD/ruby-1.9.3-p0/test/json/test_json_generate.rb:47]:
<{"a"=>2,
"b"=>3.141,
"c"=>"c",
"d"=>[1, "b", 3.14],
"e"=>{"foo"=>"bar"},
"g"=>"\"\u0000\u001F",
"h"=>1000.0,
"i"=>0.001}> expected but was
<{"a"=>2,
```

"b"=>3.141,
"c"=>"c",
"d"=>[1, "b", 3.14],
"e"=>{"foo"=>"bar"},
"g"=>"\"\xA0\xA0",
"h"=>1000.0,
"i"=>0.001}>.
5) Failure:
test_generate_pretty(TC_JSONGenerate) [/builddir/build/BUILD/ruby-1.9.3-p0/test/json/test_json_generate.rb:61]:
<{"a"=>2,
"b"=>3.141,
"c"=>"c",
"d"=>[1, "b", 3.14],
"e"=>{"foo"=>"bar"},
"g"=>"\"\u0000\u001F",
"h"=>1000.0,
"i"=>0.001}> expected but was
<{"a"=>2,
"b"=>3.141,
"c"=>"c",
"d"=>[1, "b", 3.14],
"e"=>{"foo"=>"bar"},
"g"=>"\"\xA0\xA0",
"h"=>1000.0,
"i"=>0.001}>.
6) Failure:
test_own_state(TC_JSONGenerate) [/builddir/build/BUILD/ruby-1.9.3-p0/test/json/test_json_generate.rb:92]:
<{"a"=>2,
"b"=>3.141,
"c"=>"c",
"d"=>[1, "b", 3.14],
"e"=>{"foo"=>"bar"},
"g"=>"\"\u0000\u001F",
"h"=>1000.0,
"i"=>0.001}> expected but was
<{"a"=>2,
"b"=>3.141,
"c"=>"c",
"d"=>[1, "b", 3.14],
"e"=>{"foo"=>"bar"},
"g"=>"\"\xA0\xA0",
"h"=>1000.0,
"i"=>0.001}>.
7) Failure:
test_chars(TC_JSONUnicode) [/builddir/build/BUILD/ruby-1.9.3-p0/test/json/test_json_unicode.rb:57]:
<"\x00"> expected but was
<"\xA0">.
8) Failure:
test_unicode(TC_JSONUnicode) [/builddir/build/BUILD/ruby-1.9.3-p0/test/json/test_json_unicode.rb:20]:
<["\u00A9 \u2260 \u20AC! \u0001"]> expected but was
<["\u00A9 \u2260 \u20AC! \xD8"]>.

Please note that I am building Ruby using GCC 4.7 on Fedora Rawhide. I have not seen this errors before with GCC 4.6

Associated revisions

Revision c358a4cb - 01/15/2012 06:52 AM - naruse (Yui NARUSE)

- ext/json/parser/parser.rl (json_string_unescape): workaround fix for over optimization of GCC 4.7. [ruby-core:42085] [Bug #5888] http://gcc.gnu.org/bugzilla/show_bug.cgi?id=51862

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@34306 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 34306 - 01/15/2012 06:52 AM - naruse (Yui NARUSE)

- ext/json/parser/parser.rl (json_string_unescape): workaround fix for over optimization of GCC 4.7. [ruby-core:42085] [Bug #5888] http://gcc.gnu.org/bugzilla/show_bug.cgi?id=51862

Revision 34306 - 01/15/2012 06:52 AM - naruse (Yui NARUSE)

- ext/json/parser/parser.rl (json_string_unescape): workaround fix for over optimization of GCC 4.7. [ruby-core:42085] [Bug #5888] http://gcc.gnu.org/bugzilla/show_bug.cgi?id=51862

Revision 34306 - 01/15/2012 06:52 AM - naruse (Yui NARUSE)

- ext/json/parser/parser.rl (json_string_unescape): workaround fix for over optimization of GCC 4.7. [ruby-core:42085] [Bug #5888] http://gcc.gnu.org/bugzilla/show_bug.cgi?id=51862

Revision 34306 - 01/15/2012 06:52 AM - naruse (Yui NARUSE)

- ext/json/parser/parser.rl (json_string_unescape): workaround fix for over optimization of GCC 4.7. [ruby-core:42085] [Bug #5888] http://gcc.gnu.org/bugzilla/show_bug.cgi?id=51862

Revision 34306 - 01/15/2012 06:52 AM - naruse (Yui NARUSE)

- ext/json/parser/parser.rl (json_string_unescape): workaround fix for over optimization of GCC 4.7. [ruby-core:42085] [Bug #5888] http://gcc.gnu.org/bugzilla/show_bug.cgi?id=51862

Revision 34306 - 01/15/2012 06:52 AM - naruse (Yui NARUSE)

- ext/json/parser/parser.rl (json_string_unescape): workaround fix for over optimization of GCC 4.7. [ruby-core:42085] [Bug #5888] http://gcc.gnu.org/bugzilla/show_bug.cgi?id=51862

Revision 63da304f - 02/05/2012 03:06 PM - naruse (Yui NARUSE)

merge revision(s) 34306:

```
    * ext/json/parser/parser.rl (json_string_unescape): workaround fix
      for over optimization of GCC 4.7. [ruby-core:42085] [Bug #5888]
      http://gcc.gnu.org/bugzilla/show_bug.cgi?id=51862
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_1_9_3@34432 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 34432 - 02/05/2012 03:06 PM - naruse (Yui NARUSE)

merge revision(s) 34306:

```
* ext/json/parser/parser.rl (json_string_unescape): workaround fix
  for over optimization of GCC 4.7. [ruby-core:42085] [Bug #5888]
  http://gcc.gnu.org/bugzilla/show_bug.cgi?id=51862
```

Revision e80d0a96 - 04/15/2013 05:58 AM - naruse (Yui NARUSE)

merge revision(s) 34306:

```
    * ext/json/parser/parser.rl (json_string_unescape): workaround fix
      for over optimization of GCC 4.7. [ruby-core:42085] [Bug #5888]
      http://gcc.gnu.org/bugzilla/show_bug.cgi?id=51862
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_1_9_2@40305 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 01/12/2012 09:48 PM - vo.x (Vit Ondruch)

Simpler reproducer is:

ruby -rjson -v -e "20.times { puts JSON.parse '[\"\u0000\"]'}"

ruby 1.9.3p0 (2011-10-30) [x86_64-linux]


h
P
8

As you can see, I obtain 20 time different character. That is weird.

#2 - 01/12/2012 10:56 PM - vo.x (Vit Ondruch)

When building with -O0, everything seems to work.

#3 - 01/13/2012 05:32 PM - Anonymous

The flaw seems to be in ext/json/parser/parser.c - I compiled only this one with -O0 and the others with -O2 and everything worked fine.
When I compile everything with -O2 and use

valgrind --track-origins=yes -v --read-var-info=yes ruby -rjson -v -e "puts JSON.parse '[\"\u0000\"]'"

Part of the valgrind output is quite suspicious:

```
==3350== Conditional jump or move depends on uninitialised value(s)
==3350==    at 0x4EAB4F5: rb_io_puts (io.c:6223)
==3350==    by 0x4EAB6AE: io_puts_ary (io.c:6174)
==3350==    by 0x4F853EF: exec_recursive_i (thread.c:3940)
==3350==    by 0x4F85A43: exec_recursive (thread.c:3991)
==3350==    by 0x4EAB42C: rb_io_puts (io.c:6217)
==3350==    by 0x4F787C1: vm_call0 (vm_eval.c:79)
==3350==    by 0x4F7BFC2: vm_call_method (vm_insnhelper.c:404)
==3350==    by 0x4F72191: vm_exec_core (insns.def:1015)
==3350==    by 0x4F777B7: vm_exec (vm.c:1220)
==3350==    by 0x4F7E7ED: rb_iseq_eval_main (vm.c:1461)
==3350==    by 0x4E89E79: ruby_exec_internal (eval.c:204)
==3350==    by 0x4E8A92C: ruby_exec_node (eval.c:251)
==3350==  Uninitialised value was created by a stack allocation
==3350==    at 0xD3E1430: JSON_parse_string (parser.rl:497)
==3350==
==3350== Syscall param write(buf) points to uninitialised byte(s)
==3350==    at 0x524E00D: ??? (in /lib64/libpthread-2.15.so)
==3350==    by 0x4EA43B3: io_flush_buffer_sync (io.c:640)
==3350==    by 0x4EA9D15: rb_io_fptr_cleanup (io.c:3490)
==3350==    by 0x4EABCB0: rb_io_fptr_finalize (io.c:3579)
==3350==    by 0x4E99939: finalize_list (gc.c:2944)
==3350==    by 0x4E9D301: rb_gc_call_finalizer_at_exit (gc.c:3075)
==3350==    by 0x4E8C15A: ruby_cleanup (eval.c:147)
==3350==    by 0x4E8C33E: ruby_run_node (eval.c:244)
==3350==    by 0x40086A: main (main.c:38)
==3350==  Address 0xd3039b0 is 0 bytes inside a block of size 8,192 alloc'd
==3350==    at 0x4C284CD: malloc (vg_replace_malloc.c:236)
==3350==    by 0x4E9C454: vm_xmalloc (gc.c:764)
==3350==    by 0x4EAA3F0: io_binwrite (io.c:839)
==3350==    by 0x4EAA67D: io_write (io.c:945)
==3350==    by 0x4F787C1: vm_call0 (vm_eval.c:79)
==3350==    by 0x4F7905F: rb_funcall (vm_eval.c:235)
==3350==    by 0x4EAB475: rb_io_puts (io.c:6222)
==3350==    by 0x4EAB6AE: io_puts_ary (io.c:6174)
==3350==    by 0x4F853EF: exec_recursive_i (thread.c:3940)
==3350==    by 0x4F85A43: exec_recursive (thread.c:3991)
==3350==    by 0x4EAB42C: rb_io_puts (io.c:6217)
==3350==    by 0x4F787C1: vm_call0 (vm_eval.c:79)
==3350==  Uninitialised value was created by a stack allocation
==3350==    at 0xD3E1430: JSON_parse_string (parser.rl:497)
```

Mainly the last uninitialized value it particularly interesting, yet I'm not sure what exactly is causing the problem.

#4 - 01/13/2012 05:56 PM - naruse (Yui NARUSE)

Could you compare disassembled result with -O0, and try #pragma gcc optimize ("O0")?

#5 - 01/13/2012 07:52 PM - Anonymous

- File gcc-o0 added

- File gcc-o2 added

- File pragma-gcc-o0 added

Here are the disassembled results for function JSON_parse_string for:

- compilation with gcc -O0
- compilation with pragma GCC optimize ("O0")
- compilation with gcc -O2 The first two work properly, the last one doesn't.

#6 - 01/13/2012 11:41 PM - Anonymous

After few hours of debugging, it seems that the flaw may actually be in the json_string_unescape function (in the same file, deeper in the callstack).
When compiled with -O0, it shows nothing suspicious, but with -O2, it shows some possible problems:

1344 unescape_len = convert_UTF32_to_UTF8(buf, ch);
convert_UTF32_to_UTF8 (ch=0, buf=0x7fffffffdd00 "\270",) at parser.c:46

-- The incomplete sequence is the thing that is weird. The "ch" variable is created using some byte shifting in the json_string_unescape function, which may be the cause (if the optimization does something with the byte shifting).

#7 - 01/15/2012 08:46 AM - naruse (Yui NARUSE)

Bohuslav Kabrda wrote:

> After few hours of debugging, it seems that the flaw may actually be in the json_string_unescape function (in the same file, deeper in the callstack).
> When compiled with -O0, it shows nothing suspicious, but with -O2, it shows some possible problems:
>
> 1344 unescape_len = convert_UTF32_to_UTF8(buf, ch);
> convert_UTF32_to_UTF8 (ch=0, buf=0x7fffffffdd00 "\270",) at parser.c:46
>
> -- The incomplete sequence is the thing that is weird. The "ch" variable is created using some byte shifting in the json_string_unescape function, which may be the cause (if the optimization does something with the byte shifting).

Incomplete sequence in buf is not the root of this issue because its content and the length are explicitly
specified in convert_UTF32_to_UTF8().
But yes, the bug is here.

The root of this issue is optimization for convert_UTF32_to_UTF8().
With -O2 convert_UTF32_to_UTF8() is inlined to json_string_unescape.
If ch < 0x7F, it runs as

```
0x00000008034017a0 <+1120>:  cmp    $0x7f,%rsi                  # if (ch <= 0x7F) {
0x00000008034017a4 <+1124>:  jbe    0x80340180c
0x0000000803401811 <+1233>:  jmp    0x8034017b4
0x00000008034017b4 <+1140>:  lea    0x20(%rsp),%rsi
0x00000008034017b9 <+1145>:  jmpq   0x8034016c0    # return
```

```
0x000000080340180c <+1228>:  mov    $0x1,%edx                    # int len = 1;
```

So it doesn't change buf.
I concluded this issue is caused by gcc 4.7's optimization; it wrongly optimizes out assignment to buf.

Its workaround can be following:

```
diff --git a/ext/json/parser/parser.c b/ext/json/parser/parser.c
index d1d14c7..f3bf50c 100644
--- a/ext/json/parser/parser.c
+++ b/ext/json/parser/parser.c
@@ -40,8 +40,9 @@ static UTF32 unescape_unicode(const unsigned char *p)
 return result;
 }

-static int convert_UTF32_to_UTF8(char *buf, UTF32 ch)
+static int convert_UTF32_to_UTF8(char *out, UTF32 ch)
 {
-    volatile char *buf = out;  int len = 1;  if (ch <= 0x7F) {      buf[0] = (char) ch;
```

#8 - 01/15/2012 10:29 AM - naruse (Yui NARUSE)

- Status changed from Open to Assigned

- Assignee set to naruse (Yui NARUSE)

filed it to GCC's tracker.

For JSON, following is simple patch:

diff --git a/ext/json/parser/parser.c b/ext/json/parser/parser.c
index d1d14c7..1773616 100644
--- a/ext/json/parser/parser.c
+++ b/ext/json/parser/parser.c
@@ -1293,6 +1293,7 @@ static VALUE json_string_unescape(VALUE result, char *string, char *stringEnd)
{
char *p = string, *pe = string, *unescape;
int unescape_len;

- char buf[4];

  while (pe < stringEnd) {
  if (*pe == '\') {
  @@ -1325,7 +1326,6 @@ static VALUE json_string_unescape(VALUE result, char *string, char *stringEnd)
  if (pe > stringEnd - 4) {
  return Qnil;
  } else {

- [redacted]

```
UTF32 ch = unescape_unicode((unsigned char *) ++pe);
pe += 3;
if (UNI_SUR_HIGH_START == (ch & 0xFC00)) {
```

#9 - 01/15/2012 03:52 PM - naruse (Yui NARUSE)

- Status changed from Assigned to Closed

- % Done changed from 0 to 100

This issue was solved with changeset r34306.
Vit, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

- ext/json/parser/parser.rl (json_string_unescape): workaround fix for over optimization of GCC 4.7. [ruby-core:42085] [Bug #5888]
  http://gcc.gnu.org/bugzilla/show_bug.cgi?id=51862

#10 - 01/15/2012 04:59 PM - normalperson (Eric Wong)

Yui NARUSE naruse@airemix.jp wrote:

> filed it to GCC's tracker.

I don't believe this is a GCC bug: space for buf[4] can be reclaimed
as soon as execution finishes in the scope where buf[4] was declared.

> For JSON, following is simple patch:

This was always a bug, we were just lucky because GCC wasn't as
aggressive when popping the stack.

> diff --git a/ext/json/parser/parser.c b/ext/json/parser/parser.c
> index d1d14c7..1773616 100644
> --- a/ext/json/parser/parser.c
> +++ b/ext/json/parser/parser.c
> @@ -1293,6 +1293,7 @@ static VALUE json_string_unescape(VALUE result, char *string, char *stringEnd)
> {
> char *p = string, *pe = string, *unescape;
> int unescape_len;
>
> - char buf[4];
>
> while (pe < stringEnd) {
> if (*pe == '\') {
> @@ -1325,7 +1326,6 @@ static VALUE json_string_unescape(VALUE result, char *string, char *stringEnd)
> if (pe > stringEnd - 4) {
> return Qnil;
> } else {

- [redacted]

```
                UTF32 ch = unescape_unicode((unsigned char *) ++pe);
                pe += 3;
                if (UNI_SUR_HIGH_START == (ch & 0xFC00)) {
```

#11 - 01/15/2012 05:00 PM - naruse (Yui NARUSE)

Eric Wong wrote:

> Yui NARUSE naruse@airemix.jp wrote:
>
> > filed it to GCC's tracker.
>
> I don't believe this is a GCC bug: space for buf[4] can be reclaimed
> as soon as execution finishes in the scope where buf[4] was declared.

GCC answered the same thing.
http://gcc.gnu.org/bugzilla/show_bug.cgi?id=51862#c2

> > For JSON, following is simple patch:
>
> This was always a bug, we were just lucky because GCC wasn't as
> aggressive when popping the stack.

I made pull request https://github.com/flori/json/pull/115

#12 - 01/16/2012 05:00 AM - vo.x (Vit Ondruch)

Thank you guys. Can we backport it for 1.9.3 please?

#13 - 01/16/2012 04:30 PM - Anonymous

- File ruby-1.9.3-fix-json-parser.patch added

The attached patch is a backport for Ruby 1.9.3.

#14 - 01/16/2012 08:24 PM - naruse (Yui NARUSE)

- Tracker changed from Bug to Backport

- Project changed from Ruby master to Backport193

- Status changed from Closed to Open

#15 - 02/06/2012 12:06 AM - naruse (Yui NARUSE)

- Status changed from Open to Closed

This issue was solved with changeset r34432.
Vit, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

---

merge revision(s) 34306:

```
* ext/json/parser/parser.rl (json_string_unescape): workaround fix
  for over optimization of GCC 4.7. [ruby-core:42085] [Bug #5888]
  http://gcc.gnu.org/bugzilla/show_bug.cgi?id=51862
```

Files

gcc-o0	22.7 KB	01/13/2012	Anonymous
gcc-o2	16.1 KB	01/13/2012	Anonymous
pragma-gcc-o0	22.5 KB	01/13/2012	Anonymous
ruby-1.9.3-fix-json-parser.patch	1.65 KB	01/16/2012	Anonymous