Exception is caught in rescue above ensure

Given the following code:

```ruby
def foo
  i = 0
  3.times do |n|
    begin
      puts "a"
      i += 1
      next if i > 1
      puts "b"
      rescue => e
        puts "rescue in foo, caught #{e}"
      ensure
        puts "ensure in foo, yield from foo: #{n}"  
        yield n
      end
    end
  end
  begin
    x = 0
    foo do |o|
      puts o
      x += 1
      raise "test" if x > 1
      puts "done yielding"
    end
    rescue => e
      puts "rescue outside, caught #{e}"
    ensure
      puts "ensure outside"
    end
  end
end
```

The output is:

```
a
b
ensure in foo, yield from foo: 0
0
done yielding
a
ensure in foo, yield from foo: 1
1
rescue in foo, caught test
ensure in foo, yield from foo: 1
1
rescue outside, caught test
ensure outside
```

So the exception raised within the yielded block is caught in the rescue block above the ensure block which yielded. That sounds wrong to me. Or is it intended? The issue seems to be with the usage of next. Also, exception is caught inside AND outside as it
seems that the ensure block ends up being called twice ?!

If I change the code to this:

```ruby
def foo
  i = 0
  3.times do |n|
    begin
      puts "a"
      i += 1
      # next if i > 1
      puts "b"
      rescue => e
        puts "rescue in foo, caught #{e}"
        ensure
          puts "ensure in foo, yield from foo: #{n}"
          yield n
        end
    end
  end
begin
  x = 0
  foo do |o|
    puts o
    x += 1
    raise "test" if x > 1
    puts "done yielding"
  end
rescue => e
  puts "rescue outside, caught #{e}"
ensure
  puts "ensure outside"
end
```

Then the output is:

```
  a
  b
  ensure in foo, yield from foo: 0
  0
  done yielding
  a
  b
  ensure in foo, yield from foo: 1
  1
  rescue outside, caught test
  ensure outside
```

I would have expected this output also when using next as above.

Related Issues:

- Related to Ruby master - Bug #16618: Ensure called twice when raise in ensure

Associated revisions

Revision 609de71f - 04/22/2021 02:33 AM - ko1 (Koichi Sasada)

- fix raise in exception with jump

  add_ensure_iseq() adds ensure block to the end of
  jump such as next/redo/return. However, if the rescue
  cause are in the body, this rescue catches the exception
  in ensure clause.
ensure
raise
end

In this case, R should not be executed, but executed without this patch.

Fixes [Bug #13930]
Fixes [Bug #16618]

A part of tests are written by @jeremyevans https://github.com/ruby/ruby/pull/4291

History

#1 - 09/22/2017 02:59 PM - msauter (Michael Sauter)
- Description updated

#2 - 09/22/2017 03:00 PM - msauter (Michael Sauter)
- Description updated

#3 - 09/22/2017 03:02 PM - msauter (Michael Sauter)
- Description updated

#4 - 09/25/2017 12:52 PM - shyouhei (Shyouhei Urabe)
- Description updated

#5 - 10/28/2018 09:13 AM - wanabe (_ wanabe)
- File bug13930.patch added
- File bug13930.disasm added

I think it's due to compile_next() and add_ensure_iseq().

The following is a reduced script result.

```ruby
$ ruby -e 'iseq = RubyVM::InstructionSequence.compile("1.times do; begin; p :loop; next; rescue; p :NG_rescue; ensure; raise "test"; end; end rescue p $!"); File.write("bug13930.disasm", iseq.disasm); iseq.eval'
```

`:loop
`:NG_rescue
#<RuntimeError: test>

Attached "bug13930.disasm" is the disasm result.
"catch type: rescue st: 0001 ed: 0019" may be wrong and it might be a good to be "ed: 0006" or "ed: 0008". Because "0009 putself" is the receiver of "0012 opt_send_without_block " in ensure clause.

I wrote "bug13930.patch" but this is incorrect because:

1. The patch has a performance issue because of throw instruction.
   - I haven't measured benchmark yet and I don't know how.
2. "next in while loop" pattern should be also fixed.
3. compile_break() / compile_redo() / compile_return() will have same issue because they use add_ensure_iseq().
   - But I can't fix them because of "break from proc-closure" error.

#6 - 12/29/2019 10:38 AM - hsbt (Hiroshi SHIBATA)
- Tags set to wip-patch, core

#7 - 02/10/2020 02:27 AM - jeremyevans0 (Jeremy Evans)
- Related to Bug #16618: Ensure called twice when raise in ensure added

#8 - 03/19/2021 11:51 PM - jeremyevans0 (Jeremy Evans)
- Assignee set to ko1 (Koichi Sasada)
- Status changed from Open to Assigned

Thanks to wanabe (_ wanabe)'s insight that this issue is related to the scope of the rescue catch entry, I came up with a pull request that avoids the issues wanabe (_ wanabe) identified in his patch: https://github.com/ruby/ruby/pull/4291. Instead of trying to fix things cleanly (which I don't know how to do), just adjust the scope of the rescue catch entry so it doesn't cover the ensure block. This approach fixes the original example and wanabe (_ wanabe)'s simplified example in this issue, as well as all failing examples in #16618. The heuristic this approach uses may not fix all issues, and it may cause issues, but with my limited knowledge of the compiler I cannot be certain of either. However, the pull request does pass all existing tests.
#9 - 04/22/2021 02:33 AM - ko1 (Koichi Sasada)
- Status changed from Assigned to Closed

Applied in changeset git|609024e71f0af3e8ba3f22b993e4444e2e5bb05709.

---

fix raise in exception with jump

add_ensure_iseq() adds ensure block to the end of
jump such as next/redo/return. However, if the rescue
cause are in the body, this rescue catches the exception
in ensure clause.

iter do
next
rescue
ensure
raise
end

In this case, R should not be executed, but executed without this patch.

Fixes [Bug #13930]
Fixes [Bug #16618]

A part of tests are written by @jeremyevans. https://github.com/ruby/ruby/pull/4291

---

**Files**

<table>
<thead>
<tr>
<th>File</th>
<th>Size</th>
<th>Date</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>bug13930.disasm</td>
<td>3.62 KB</td>
<td>10/28/2018</td>
<td>wanabe</td>
</tr>
<tr>
<td>bug13930.patch</td>
<td>492 Bytes</td>
<td>10/28/2018</td>
<td>wanabe</td>
</tr>
</tbody>
</table>