Ruby - Bug #17623

irb starts with some local variables already defined

02/12/2021 09:27 AM - UlyssesZhan (Ulysses Zhan)

Status:	Closed		
Priority:	Normal		
Assignee:	marcandre (Marc-Andre Lafortune)		
Target version:			
ruby -v:	ruby 3.0.0p0 (2020-12-25 revision 95aff21468) [x64-mingw32]	**Backport:**	2.5: UNKNOWN, 2.6: UNKNOWN, 2.7: UNKNOWN, 3.0: UNKNOWN

Description

```
irb(main):001:0> a = 1
=> 1
irb(main):002:0> def f = a
=> :f
irb(main):003:0> f
=> "D:/Ruby30-x64/lib/ruby/gems/3.0.0/gems/irb-1.3.0/exe/irb"
```

I have not idea what it means.
The codes just work fine outside irb.
Maybe it is a bug.

Associated revisions

Revision 3503c94af501e38164613ef8347174a27346828a - 09/09/2021 09:37 PM - Marc-Andre Lafortune

[ruby/irb] Avoid loading files' local variables [Bug #17623]

https://github.com/ruby/irb/commit/b12f0cb8e2

Revision 3503c94af501e38164613ef8347174a27346828a - 09/09/2021 09:37 PM - Marc-Andre Lafortune

[ruby/irb] Avoid loading files' local variables [Bug #17623]

https://github.com/ruby/irb/commit/b12f0cb8e2

Revision 3503c94a - 09/09/2021 09:37 PM - Marc-Andre Lafortune

[ruby/irb] Avoid loading files' local variables [Bug #17623]

https://github.com/ruby/irb/commit/b12f0cb8e2

History

#1 - 02/12/2021 11:06 AM - mame (Yusuke Endoh)

I cannot reproduce the issue.

```
$ ruby -v
ruby 3.0.0p0 (2020-12-25 revision 95aff21468) [x86_64-linux]

$ irb
irb(main):001:0* a = 1
=> 1
irb(main):002:1* def f = a
=> :f
irb(main):003:0> f
Traceback (most recent call last):
        5: from /home/mame/local/bin/irb:23:in `<main>'
        4: from /home/mame/local/bin/irb:23:in `load'
        3: from /home/mame/local/lib/ruby/gems/3.0.0/gems/irb-1.3.0/exe/irb:11:in `<top (required)>'
        2: from (irb):3:in `<main>'
        1: from (irb):2:in `f'
NameError (undefined local variable or method `a' for main:Object)
irb(main):004:0>

$ ruby
a = 1
```

```
def f = a
f
-:2:in `f': undefined local variable or method `a' for main:Object (NameError)
        from -:3:in `<main>'
```

Does your .irbrc have something?

#2 - 02/12/2021 03:30 PM - xtkoba (Tee KOBAYASHI)

It reproduces with Ruby 3.0.0-1 (x64) from RubyInstaller, and it seems that f has already been defined when the first prompt (irb(main):001:0>) appears.

It does not reproduce with my own x64-mingw32 build.

#3 - 02/12/2021 04:08 PM - xtkoba (Tee KOBAYASHI)

I found that f is indeed defined in RubyInstaller's irb.cmd:

```
--- original/irb.cmd
+++ RubyInstaller/irb.cmd
@@ -28,7 +28,7 @@
 end

 if Gem.respond_to?(:activate_bin_path)
-load Gem.activate_bin_path('irb', 'irb', version)
+f = Gem.activate_bin_path('irb', 'irb', version); require 'irbrc_predefiner'; load f
 else
 gem "irb", version
 load Gem.bin_path("irb", "irb", version)
```

The same goes for str or version with every environment:

```
irb(main):001:0> str
=> nil
irb(main):002:0> version
=> ">= 0.a"
irb(main):003:0>
```

I have no idea whether it is a bug or not that the local variables defind in irb command are visible from IRB.

#4 - 02/12/2021 04:49 PM - Eregon (Benoit Daloze)

I think that might be caused by https://bugs.ruby-lang.org/issues/9580#change-88660 cc @marcandre (Marc-Andre Lafortune)

#5 - 02/19/2021 10:05 PM - jeremyevans0 (Jeremy Evans)

- Status changed from Open to Assigned

- Assignee set to marcandre (Marc-Andre Lafortune)

@Eregon (Benoit Daloze) is correct. This issue first started in Ruby 3.0. It does not occur when the irb --context-mode is 1, 2, or 3 (Ruby 2.7 default), but it does occur when --context-mode is 4 (Ruby 3.0 default) or 0. Between Ruby 2.7 and 3.0, the default irb --context-mode switched from 3 to 4 to fix #9580.

It seems undesirable to me that irb would have local variables predefined other than _. While the f local variable is RubyInstaller specific, the str and version local variables are not. So I think this should be considered a bug.

Assigning to @marcandre (Marc-Andre Lafortune), since he made the --context-mode change in irb. One possible solution is to accept the new irb behavior, and modify the binstubs to not expose or create local variables. An easy way to do that:

```
proc do
  # previous binstub code
end.call
```

The problem with that approach is it adds a stack frame. An alternative approach that doesn't add a stack frame is to switch to a much uglier approach using instance variables (which can be removed, unlike local variables):

```
require 'rubygems'

@version = ">= 0.a"

@str = ARGV.first
if @str
  @str = @str.b[/\A_(.*)_\z/, 1]
  if @str and Gem::Version.correct?(@str)
```

```
    @version = @str
    ARGV.shift
  end
end
remove_instance_variable(:@str)

if Gem.respond_to?(:activate_bin_path)
load Gem.activate_bin_path('irb', 'irb', @version.tap{remove_instance_variable(:@version)})
else
gem "irb", @version
load Gem.bin_path("irb", "irb", @version.tap{remove_instance_variable(:@version)})
end
```

#6 - 03/18/2021 01:50 PM - marcandre (Marc-Andre Lafortune)

- Subject changed from endless def can access to outer local variables and lead to unexpected result to irb starts with some local variables already defined

I think this is a nice solution: https://github.com/ruby/irb/pull/206

#7 - 09/09/2021 09:37 PM - Anonymous

- Status changed from Assigned to Closed

Applied in changeset git|3503c94af501e38164613ef8347174a27346828a.

[ruby/irb] Avoid loading files' local variables [Bug #17623]

https://github.com/ruby/irb/commit/b12f0cb8e2