Ruby master - Misc #14190

What are the semantics of $SAFE?

12/15/2017 04:29 PM - Eregon (Benoit Daloze)

Status:	Open	
Priority:	Normal	
Assignee:		

Description

$SAFE is documented in many places as thread-local, but it seems more than that.
For example:

```
# a.rb
$SAFE=1
p $SAFE
require "#{Dir.pwd.untaint}/b.rb"

# b.rb
p [:in_b, $SAFE]
```

gives:

```
$ ruby -r./a -e 'p $SAFE'
1
[:in_b, 0]
0
```

So in b and in -e, $SAFE is 0.
Is it file-based somehow?

I was trying to understand what
https://github.com/ruby/ruby/blob/7c4306e6e9c3c4a255f4ad20134c1832dbe45ba2/test/rubygems/test_gem.rb#L9-L13
is supposed to do.
Does it make sense? What does it do?
It seems the test_* methods in that file actually read $SAFE as 0, not 1.

History

#1 - 12/15/2017 10:28 PM - mame (Yusuke Endoh)

I'm not familiar with $SAFE, but the scope seems finer:

```
$ ruby -e 'f = proc { $SAFE = 1; p [:in_proc, $SAFE] }; f.call; p [:out_of_proc, $SAFE]'
[:in_proc, 1]
[:out_of_proc, 0]
```