

Ruby trunk - Feature #14250

Make `\$\$SAFE` process global state and allow to set 0 again

12/27/2017 05:01 PM - ko1 (Koichi Sasada)

Status:	Closed	
Priority:	Normal	
Assignee:	ko1 (Koichi Sasada)	
Target version:	2.6	
Description		
<p>\$\$SAFE > 1 is removed from Ruby 2.3 and there are some opinion to remove \$\$SAFE feature ([Feature #5455]). There are several reason, but the biggest reason I think is nobody use \$\$SAFE correctly.</p> <p>Also current \$\$SAFE is thread/proc local information and it hurts performance (we need to restore \$\$SAFE information just after returning proc, even if returning by exception).</p> <p>Matz said \$\$SAFE == 1 is similar to warning and it is not a security feature, but one of the programming tool we can use to improve our program (\$SAFE == 3 was for sandbox, security feature).</p> <p>From this perspective, Matz approved us the followings:</p> <ul style="list-style-type: none">• \$\$SAFE is process global, not a Proc local state.• We can set \$\$SAFE == 0 when \$\$SAFE == 1. <p>I think we can't make big project with the above changes (how to make multi-thread programming with this \$\$SAFE?), but \$\$SAFE seems for small project (so-called <i>scripting</i>). Anyway if nobody use it, no problem on these changes.</p> <p>I will commit this change soon. Please try new spec and point out any problem you got.</p> <p>Thanks, Koichi</p>		
Related issues:		
Related to Ruby trunk - Feature #14256: Deprecate \$\$SAFE support in ERB and le...		Closed
Related to Ruby trunk - Bug #14353: \$\$SAFE should stay at least thread-local f...		Open

Associated revisions

Revision c39bdb79 - 12/28/2017 08:09 PM - ko1 (Koichi Sasada)

\$\$SAFE as a process global state. [Feature #14250]

- vm_core.h (rb_vm_t): move rb_execution_context_t::safe_level to rb_vm_t::safe_level_ because \$\$SAFE is a process (VM) global state.
- vm_core.h (rb_proc_t): remove rb_proc_t::safe_level because Proc objects don't need to keep \$\$SAFE at the creation. Also make is_from_method and is_lambda as 1 bit fields.
- cont.c (cont_restore_thread): no need to keep \$\$SAFE for Continuation.
- eval.c (ruby_cleanup): use rb_set_safe_level_force() instead of access vm->safe_level_ directly.
- eval_jump.c: End procs END{} doesn't keep \$\$SAFE.
- proc.c (proc_dup): removed and introduce rb_proc_dup in vm.c.
- safe.c (rb_set_safe_level): don't check \$\$SAFE 1 -> 0 changes.
- safe.c (safe_setter): use rb_set_safe_level().
- thread.c (rb_thread_safe_level): Thread#safe_level returns \$\$SAFE. It should be obsolete.
- transcode.c (load_transcoder_entry): rb_safe_level() only returns

0 or 1 so that this check is not needed.

- vm.c (vm_proc_create_from_captured): don't need to keep \$SAFE for Proc.
- vm.c (rb_proc_create): renamed to proc_create.
- vm.c (rb_proc_dup): moved from proc.c.
- vm.c (vm_invoke_proc): do not need to set and restore \$SAFE for Proc#call.
- vm_eval.c (rb_eval_cmd): rename a local variable to represent clearer meaning.
- lib/drb/drb.rb: restore \$SAFE.
- lib/erb.rb: restore \$SAFE, too.
- test/lib/leakchecker.rb: check \$SAFE == 0 at the end of tests.
- test/rubygems/test_gem.rb: do not set \$SAFE = 1.
- bootstraptest/test_proc.rb: catch up this change.
- spec/ruby/optional/capi/string_spec.rb: ditto.
- test/bigdecimal/test_bigdecimal.rb: ditto.
- test/fiddle/test_func.rb: ditto.
- test/fiddle/test_handle.rb: ditto.
- test/net/imap/test_imap_response_parser.rb: ditto.
- test/pathname/test_pathname.rb: ditto.
- test/readline/test_readline.rb: ditto.
- test/ruby/test_file.rb: ditto.
- test/ruby/test_optimization.rb: ditto.
- test/ruby/test_proc.rb: ditto.
- test/ruby/test_require.rb: ditto.
- test/ruby/test_thread.rb: ditto.
- test/rubygems/test_gem_specification.rb: ditto.
- test/test_tempfile.rb: ditto.
- test/test_tmpdir.rb: ditto.
- test/win32ole/test_win32ole.rb: ditto.
- test/win32ole/test_win32ole_event.rb: ditto.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@61510 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 61510 - 12/28/2017 08:09 PM - ko1 (Koichi Sasada)

\$SAFE as a process global state. [Feature #14250]

- vm_core.h (rb_vm_t): move rb_execution_context_t::safe_level to rb_vm_t::safe_level_ because \$SAFE is a process (VM) global state.
- vm_core.h (rb_proc_t): remove rb_proc_t::safe_level because Proc objects don't need to keep \$SAFE at the creation. Also make is_from_method and is_lambda as 1 bit fields.
- cont.c (cont_restore_thread): no need to keep \$SAFE for Continuation.
- eval.c (ruby_cleanup): use rb_set_safe_level_force() instead of access

vm->safe_level_ directly.

- eval_jump.c: End procs END{} doesn't keep \$SAFE.
- proc.c (proc_dup): removed and introduce rb_proc_dup in vm.c.
- safe.c (rb_set_safe_level): don't check \$SAFE 1 -> 0 changes.
- safe.c (safe_setter): use rb_set_safe_level().
- thread.c (rb_thread_safe_level): Thread#safe_level returns \$SAFE. It should be obsolete.
- transcode.c (load_transcoder_entry): rb_safe_level() only returns 0 or 1 so that this check is not needed.
- vm.c (vm_proc_create_from_captured): don't need to keep \$SAFE for Proc.
- vm.c (rb_proc_create): renamed to proc_create.
- vm.c (rb_proc_dup): moved from proc.c.
- vm.c (vm_invoke_proc): do not need to set and restore \$SAFE for Proc#call.
- vm_eval.c (rb_eval_cmd): rename a local variable to represent clearer meaning.
- lib/drb/drb.rb: restore \$SAFE.
- lib/erb.rb: restore \$SAFE, too.
- test/lib/leakchecker.rb: check \$SAFE == 0 at the end of tests.
- test/rubygems/test_gem.rb: do not set \$SAFE = 1.
- bootstraptest/test_proc.rb: catch up this change.
- spec/ruby/optional/capi/string_spec.rb: ditto.
- test/bigdecimal/test_bigdecimal.rb: ditto.
- test/fiddle/test_func.rb: ditto.
- test/fiddle/test_handle.rb: ditto.
- test/net/imap/test_imap_response_parser.rb: ditto.
- test/pathname/test_pathname.rb: ditto.
- test/readline/test_readline.rb: ditto.
- test/ruby/test_file.rb: ditto.
- test/ruby/test_optimization.rb: ditto.
- test/ruby/test_proc.rb: ditto.
- test/ruby/test_require.rb: ditto.
- test/ruby/test_thread.rb: ditto.
- test/rubygems/test_gem_specification.rb: ditto.
- test/test_tempfile.rb: ditto.
- test/test_tmpdir.rb: ditto.
- test/win32ole/test_win32ole.rb: ditto.
- test/win32ole/test_win32ole_event.rb: ditto.

Revision 61510 - 12/28/2017 08:09 PM - ko1 (Koichi Sasada)

`$$SAFE` as a process global state. [Feature #14250]

- `vm_core.h (rb_vm_t)`: move `rb_execution_context_t::safe_level` to `rb_vm_t::safe_level` because `$$SAFE` is a process (VM) global state.
- `vm_core.h (rb_proc_t)`: remove `rb_proc_t::safe_level` because Proc objects don't need to keep `$$SAFE` at the creation. Also make `is_from_method` and `is_lambda` as 1 bit fields.
- `cont.c (cont_restore_thread)`: no need to keep `$$SAFE` for Continuation.
- `eval.c (ruby_cleanup)`: use `rb_set_safe_level_force()` instead of access `vm->safe_level` directly.
- `eval_jump.c`: End procs `END{}` doesn't keep `$$SAFE`.
- `proc.c (proc_dup)`: removed and introduce `rb_proc_dup` in `vm.c`.
- `safe.c (rb_set_safe_level)`: don't check `$$SAFE 1 -> 0` changes.
- `safe.c (safe_setter)`: use `rb_set_safe_level()`.
- `thread.c (rb_thread_safe_level)`: `Thread#safe_level` returns `$$SAFE`. It should be obsolete.
- `transcode.c (load_transcoder_entry)`: `rb_safe_level()` only returns 0 or 1 so that this check is not needed.
- `vm.c (vm_proc_create_from_captured)`: don't need to keep `$$SAFE` for Proc.
- `vm.c (rb_proc_create)`: renamed to `proc_create`.
- `vm.c (rb_proc_dup)`: moved from `proc.c`.
- `vm.c (vm_invoke_proc)`: do not need to set and restore `$$SAFE` for `Proc#call`.
- `vm_eval.c (rb_eval_cmd)`: rename a local variable to represent clearer meaning.
- `lib/drb/drb.rb`: restore `$$SAFE`.
- `lib/erb.rb`: restore `$$SAFE`, too.
- `test/lib/leakchecker.rb`: check `$$SAFE == 0` at the end of tests.
- `test/rubygems/test_gem.rb`: do not set `$$SAFE = 1`.
- `bootstraptest/test_proc.rb`: catch up this change.
- `spec/ruby/optional/capi/string_spec.rb`: ditto.
- `test/bigdecimal/test_bigdecimal.rb`: ditto.
- `test/fiddle/test_func.rb`: ditto.
- `test/fiddle/test_handle.rb`: ditto.
- `test/net/imap/test_imap_response_parser.rb`: ditto.
- `test/pathname/test_pathname.rb`: ditto.
- `test/readline/test_readline.rb`: ditto.
- `test/ruby/test_file.rb`: ditto.
- `test/ruby/test_optimization.rb`: ditto.
- `test/ruby/test_proc.rb`: ditto.
- `test/ruby/test_require.rb`: ditto.
- `test/ruby/test_thread.rb`: ditto.

- test/rubygems/test_gem_specification.rb: ditto.
- test/test_tempfile.rb: ditto.
- test/test_tmpdir.rb: ditto.
- test/win32ole/test_win32ole.rb: ditto.
- test/win32ole/test_win32ole_event.rb: ditto.

History

#1 - 12/27/2017 05:41 PM - shevegen (Robert A. Heiler)

Can not comment on \$SAFE but I personally have not used \$SAFE so far in like +10 years or so. I can only remember the pickaxe mentioning it, but I have not used it in any of my ruby code.

A bit off-topic but does anyone remember if `_why`'s old ruby sandbox (the online irb, I think), made use of it? For such projects, trivial ways to control how "safe" the ruby is, may be more useful. E. g. in any restricted environment such as that.

#2 - 12/28/2017 02:32 AM - mame (Yusuke Endoh)

- File `gems-using-safe.txt` added

FYI: by using [gem-codesearch](#), I have briefly searched the gems using \$SAFE:

```
$ csearch -f '*.rb' '^\s*[\s#].*\$SAFE *=' | wc -l
147
```

Much less than I thought... The full list is attached.

#3 - 12/28/2017 05:37 AM - k0kubun (Takashi Kokubun)

- Related to Feature #14255: Deprecate \$SAFE support in ERB added

#4 - 12/28/2017 05:53 AM - k0kubun (Takashi Kokubun)

- Related to deleted (Feature #14255: Deprecate \$SAFE support in ERB)

#5 - 12/28/2017 05:54 AM - k0kubun (Takashi Kokubun)

- Related to Feature #14256: Deprecate \$SAFE support in ERB and let ERB.new take keyword arguments for it added

#6 - 12/28/2017 08:09 PM - ko1 (Koichi Sasada)

- Status changed from Open to Closed

Applied in changeset [trunk|r61510](#).

\$SAFE as a process global state. [Feature [#14250](#)]

- `vm_core.h (rb_vm_t)`: move `rb_execution_context_t::safe_level` to `rb_vm_t::safe_level_` because \$SAFE is a process (VM) global state.
- `vm_core.h (rb_proc_t)`: remove `rb_proc_t::safe_level` because Proc objects don't need to keep \$SAFE at the creation. Also make `is_from_method` and `is_lambda` as 1 bit fields.
- `cont.c (cont_restore_thread)`: no need to keep \$SAFE for Continuation.
- `eval.c (ruby_cleanup)`: use `rb_set_safe_level_force()` instead of access `vm->safe_level_` directly.
- `eval_jump.c`: End procs `END{}` doesn't keep \$SAFE.
- `proc.c (proc_dup)`: removed and introduce `rb_proc_dup` in `vm.c`.
- `safe.c (rb_set_safe_level)`: don't check \$SAFE 1 -> 0 changes.

- safe.c (safe_setter): use rb_set_safe_level().
- thread.c (rb_thread_safe_level): Thread#safe_level returns \$SAFE. It should be obsolete.
- transcode.c (load_transcoder_entry): rb_safe_level() only returns 0 or 1 so that this check is not needed.
- vm.c (vm_proc_create_from_captured): don't need to keep \$SAFE for Proc.
- vm.c (rb_proc_create): renamed to proc_create.
- vm.c (rb_proc_dup): moved from proc.c.
- vm.c (vm_invoke_proc): do not need to set and restore \$SAFE for Proc#call.
- vm_eval.c (rb_eval_cmd): rename a local variable to represent clearer meaning.
- lib/drb/drb.rb: restore \$SAFE.
- lib/erb.rb: restore \$SAFE, too.
- test/lib/leakchecker.rb: check \$SAFE == 0 at the end of tests.
- test/rubygems/test_gem.rb: do not set \$SAFE = 1.
- bootstraptest/test_proc.rb: catch up this change.
- spec/ruby/optional/capi/string_spec.rb: ditto.
- test/bigdecimal/test_bigdecimal.rb: ditto.
- test/fiddle/test_func.rb: ditto.
- test/fiddle/test_handle.rb: ditto.
- test/net/imap/test_imap_response_parser.rb: ditto.
- test/pathname/test_pathname.rb: ditto.
- test/readline/test_readline.rb: ditto.
- test/ruby/test_file.rb: ditto.
- test/ruby/test_optimization.rb: ditto.
- test/ruby/test_proc.rb: ditto.
- test/ruby/test_require.rb: ditto.
- test/ruby/test_thread.rb: ditto.
- test/rubygems/test_gem_specification.rb: ditto.
- test/test_tempfile.rb: ditto.
- test/test_tmpdir.rb: ditto.
- test/win32ole/test_win32ole.rb: ditto.
- test/win32ole/test_win32ole_event.rb: ditto.

#7 - 01/13/2018 08:00 PM - Eregon (Benoit Daloze)

- Related to Bug #14353: \$SAFE should stay at least thread-local for compatibility added

Files

gems-using-safe.txt	15.1 KB	12/28/2017	name (Yusuke Endoh)
---------------------	---------	------------	---------------------