an embedded Ruby interpreter doesn't get the full Ruby environment unless it calls `ruby_process_options()` (which is not documented)

The documented way to embed a Ruby interpreter is to call:

```ruby
RUBY_INIT_STACK;
ruby_init();
ruby_init_loadpath();
```

However, this leaves the Ruby environment incomplete. As an example, the following program:

```ruby
#include
int main(int argc, char *argv[])
{
  RUBY_INIT_STACK;
  ruby_init();
  ruby_init_loadpath();

  rb_eval_string("p Mutex.new.methods");

  return 0;
}
```

prints:


Whereas running "ruby -e 'p Mutex.new.methods'" produces:


Note that "synchronize" is missing from the former. This is because `ruby_init_prelude()` has not been called - that's what adds synchronize() to Mutex.

A workaround is to call `ruby_process_options()` as in the following:

```ruby
#include
int main(int argc, char *argv[])
{
  RUBY_INIT_STACK;
  ...
ruby_init();
ruby_init_loadpath();
static char args[] = { "ruby", "/dev/null"};
ruby_process_options(2, args);

rb_eval_string("p Mutex.new.methods");

return 0;
}

This seems very clumsy!

Some possible solutions are:

- call ruby_init_prelude() from ruby_init()
- change the linkage of ruby_init_prelude() to be non-static and have the Ruby embedded program call it explicitly (requires a documentation change).

Related issues:
Related to Ruby master - Feature #3731: Easier Embedding API for Ruby

## History

### #1 - 11/24/2012 05:32 PM - mame (Yusuke Endoh)
- Tracker changed from Bug to Feature
- Status changed from Open to Assigned
- Assignee set to ko1 (Koichi Sasada)

I think that this is a feature request, not a bug. Moving the feature tracker. Related to #3731.

--
Yusuke Endoh mame@tsg.ne.jp

### #2 - 11/24/2012 05:32 PM - mame (Yusuke Endoh)
- Target version set to 2.6

### #3 - 01/31/2017 09:31 AM - ko1 (Koichi Sasada)
- Status changed from Assigned to Closed

Continue it at #3731.