Ruby master - Feature #14386

Add option to let Kernel.#system raise error instead of returning false

01/23/2018 05:39 PM - k0kubun (Takashi Kokubun)

Status:	Closed
Priority:	Normal
Assignee:	
Target version:	

Description

I sometimes write code like:

```
system('git pull origin master') || raise('Failed to execute: git pull origin master')
system('bundle check || bundle install') || raise(
'Failed to execute: bundle check || bundle install')
```

Using rake, we can simplify the above code to the following one, but there's no way to do that outside Rakefile. (Note that I just want to do the same thing as "bash -e", the error message is actually not important.)

```
sh 'git pull origin master'
sh 'bundle check || bundle install'
```

If we add the following option, we can simplify such code even when we're not on a rake task. I'm not sure whether it's a good name or not though.

```
system 'git pull origin master', assert_status: true
system 'bundle check || bundle install', assert_status: true
# => RuntimeError: Command failed with status (1): bundle check || bundle install
```

Associated revisions

Revision 62025 - 01/24/2018 02:11 PM - k0kubun (Takashi Kokubun)

process.c: add :exception option to Kernel.#system

to raise error when it fails.

[Feature 14386] [GH-1795]

Revision 1659a166 - 02/02/2018 11:03 AM - kazu

Use more verbose status in error messages

of system with exception: true like Process::Status#inspect
[Feature #14386] [ruby-core:85013]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@62158 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 62158 - 02/02/2018 11:03 AM - znz (Kazuhiro NISHIYAMA)

Use more verbose status in error messages

of system with exception: true like Process::Status#inspect
[Feature #14386] [ruby-core:85013]

Revision 62158 - 02/02/2018 11:03 AM - kazu

Use more verbose status in error messages

of system with exception: true like Process::Status#inspect
[Feature #14386] [ruby-core:85013]

History

#1 - 01/23/2018 05:52 PM - shevegen (Robert A. Heiler)

If I understood it correctly, you mean to pass in an optional hash
to be the second argument to system() right?

I have nothing against it. I think it would change existing behaviour
though, if I read http://ruby-doc.org/core-2.5.0/Kernel.html#method-i-system
correctly, so perhaps this may have to be targeted for 3.0, if
matz approves of it? Though it may be that I misunderstood the suggestion.

To the name, I think assert_status is ok, even if it is a bit verbose.
I can't think of a shorter one though... the one word variants I can
think of are less meaningful than assert_status. I could think of
assert_runtime or assert_raise or something like that but all of these
variants are not great really ... so assert_status seems somewhat ok to
me.

#2 - 01/23/2018 06:57 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

If an option would be used to make system raise instead of returning a boolean, then it would make more sense to return the combined err + out output instead of true or false. That could be quite useful and simpler to use than Open3's capture2e for some scenarios. Or maybe a new method could be suggested instead of overloading system.

#3 - 01/24/2018 12:04 AM - k0kubun (Takashi Kokubun)

> I have nothing against it. I think it would change existing behaviour
> though, if I read http://ruby-doc.org/core-2.5.0/Kernel.html#method-i-system
> correctly

I don't get what you mean. Do you mean existing program may be passing :assert_status in the second (or the third if env is given) Hash?

It's already taking the same options as Process.spawn, and I'm not going to change Process.spawn too here. So it would make a little hard to understand the behavior of Kernel.#system's options. But I can't imagine why it becomes breaking.

> If an option would be used to make system raise instead of returning a boolean, then it would make more sense to return the combined err + out output instead of true or false.

I agree that returning true would not make sense and such method would be convenient too. But the behavior is not consistent with the current Kernel.#system behavior. So probably it's better to use the different method name or to add a new option to Kernel.#`.

#4 - 01/24/2018 07:11 AM - akr (Akira Tanaka)

I don't like the keyword name, "assert_status".

I think "exception" is better.
It is consistent with read_nonblock, etc.

#5 - 01/24/2018 07:14 AM - matz (Yukihiro Matsumoto)

Agree with adding keyword argument to specify raising an exception (exception:true sounds reasonable).

Matz.

#6 - 01/24/2018 09:00 AM - k0kubun (Takashi Kokubun)

I created a patch for exception: true. https://github.com/ruby/ruby/pull/1795

#7 - 01/24/2018 02:15 PM - k0kubun (Takashi Kokubun)

- Status changed from Open to Closed

Applied in changeset r62025.

For rosenfeld's suggestion, open3 would be a good place to have the functionality.

#8 - 01/25/2018 02:31 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

Thanks, Takashi, but very often in my code, when dealing with output and error management of external commands, I end up wrapping them in a utility method of my own to properly handle errors and control flow anyway, so I'm not sure I'm interested in creating an issue for that :) I just mentioned the return value here because it made sense to me when I read the proposal, not because I care too much about it ;)

But thanks anyway for mentioning where I should suggest such method addition if I wanted to :) And thanks for the patch, as I think I'll have some use for it in some of my code.

#9 - 01/27/2018 04:21 AM - znz (Kazuhiro NISHIYAMA)

Process::Status#inspect includes signaled information, etc.
How about add such information instead of exit status number only in error messages?

examples: https://gist.github.com/znz/b3c081d0e63d87af6402e27f514f2f15

#10 - 01/27/2018 04:29 AM - k0kubun (Takashi Kokubun)

That sounds good.

#11 - 01/30/2018 02:56 PM - znz (Kazuhiro NISHIYAMA)

How about using part after pid of Process::Status#inspect?

```
% ruby -ve 'begin;system(%q(ruby -e "exit(false)"), exception: true);rescue => e;p e;p Process.last_status;end
'
ruby 2.6.0dev (2018-01-30 trunk 62111) [x86_64-darwin16]
#<RuntimeError: Command failed with exit 1: ruby -e "exit(false)">
#<Process::Status: pid 38365 exit 1>
% ruby -ve 'begin;system(%q(ruby -e "Process.kill(:TERM, Process.pid)"), exception: true);rescue;p $!;p $?;end
'
ruby 2.6.0dev (2018-01-30 trunk 62111) [x86_64-darwin16]
#<RuntimeError: Command failed with SIGTERM (signal 15): ruby -e "Process.kill(:TERM, Process.pid)">
#<Process::Status: pid 38389 SIGTERM (signal 15)>

diff --git a/process.c b/process.c
index 8b8268e9f1..6ad9216564 100644
--- a/process.c
+++ b/process.c
@@ -548,7 +548,8 @@ pst_pid(VALUE st)
 static void
 pst_message(VALUE str, rb_pid_t pid, int status)
 {
-    rb_str_catf(str, "pid %ld", (long)pid);
+    if (pid != (rb_pid_t)-1)
+        rb_str_catf(str, "pid %ld", (long)pid);
     if (WIFSTOPPED(status)) {
 	int stopsig = WSTOPSIG(status);
 	const char *signame = ruby_signal_name(stopsig);
@@ -4090,8 +4091,10 @@ rb_f_system(int argc, VALUE *argv)
     status = PST2INT(rb_last_status_get());
     if (status == EXIT_SUCCESS) return Qtrue;
     if (eargp->exception) {
-        rb_raise(rb_eRuntimeError, "Command failed with status (%d): %s",
-                 WEXITSTATUS(status), RSTRING_PTR(eargp->invoke.sh.shell_script));
+        VALUE str = rb_str_buf_new(0);
+        pst_message(str, (rb_pid_t)-1, status);
+        rb_raise(rb_eRuntimeError, "Command failed with%"PRIsVALUE": %s",
+                 str, RSTRING_PTR(eargp->invoke.sh.shell_script));
     }
     else {
         return Qfalse;
```

#12 - 01/31/2018 11:47 AM - k0kubun (Takashi Kokubun)

+1. It seems more consistent. Please do so.