Ruby trunk - Feature #9785

Feature Proposal: Dir.chdir Thread Safety

04/29/2014 02:13 PM - schneems (Richard Schneeman)

Status:	Rejected
Priority:	Normal
Assignee:	
Target version:	2.2.0

Description

I am proposing that Dir.chdir with a block be local to the current thread and any threads that are created inside of that block. FileUtils.cd and FileUtils.chdir should also behave the same way.

Currently Dir.chdir will change the directory for the entire process. This makes writing a program that modifies different directories in threads very difficult. Here is some ruby code that demonstrates the problem:

```
# /tmp/code.rb

require 'fileutils'

FileUtils.mkdir_p("/tmp/foo")
FileUtils.mkdir_p("/tmp/bar")


threads = []
threads << Thread.new do
  Dir.chdir("/tmp/foo") do
    puts "Thread in Dir.chdir('/tmp/foo') pwd: #{`pwd`}"
  end
end


threads << Thread.new do
  puts "Thread without Dir.chdir         pwd: #{`pwd`}"
end

threads.map(&:join)
```

When you run it you get different results:

```
$ ruby /tmp/code.rb
Thread without Dir.chdir         pwd: /tmp
Thread in Dir.chdir('/tmp/foo') pwd: /private/tmp/foo

$ ruby /tmp/code.rb
Thread in Dir.chdir('/tmp/foo') pwd: /private/tmp/foo
Thread without Dir.chdir         pwd: /private/tmp/foo
```

This is because Dir.chdir is not limited to the scope of the block but rather changes the working directory globally for the entire process including different threads.

Threads in MRI are very good for reading and writing to the disk, however many times a programmer wishes to read or write to disk they will want to use Dir.chdir. The current behavior of Dir.chdir prevents a programmer from changing directory inside of threads and can be very confusing for anyone who does not know this behavior.

For a better programming experience either we can make Dir.chdir thread aware, or introduce a new way to change the directory inside of a new thread such as Dir.threadsafe_chdir, I believe the first option is the best.

History

#1 - 04/29/2014 03:24 PM - schneems (Richard Schneeman)

It's come to my attention that this is fairly hardcoded into the OS (changing CWD is a per-process operation rather than a per-thread one). I do not have a proposed implementation for how to change directory within a thread, perhaps we could take ideas from another language allows this functionality if there are any.

#2 - 04/29/2014 03:38 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

If forking is an option for you, it would allow you to use chdir blocks the way you want I think.

#3 - 04/29/2014 04:49 PM - normalperson (Eric Wong)

richard.schneeman@gmail.com wrote:

> It's come to my attention that this is fairly hardcoded into the OS (changing CWD is a per-process operation rather than a per-thread one). I do not have a proposed implementation for how to change directory within a thread, perhaps we could take ideas from another language allows this functionality if there are any.

Right, this is one of the reasons the *at family of syscalls (openat, renameat, etc...) was introduced into POSIX.

Adding support for those might be good idea. However, OS support outside Linux/Solaris is probably still limited at the moment.

Linux also allows unsetting the CLONE_FS flag for cloned threads, but that's completely unportable.

#4 - 04/30/2014 04:19 AM - nobu (Nobuyoshi Nakada)

MVM branch has incomplete per-thread cwd, some methods are not implemented however, e.g., File#rename.

#5 - 05/01/2014 03:33 PM - schneems (Richard Schneeman)

I think maybe the openat and family of *at calls is close to my original proposal but does not help for executing a script inside of a chdir block: https://github.com/heroku/hatchet/commit/f882d8920525df6c1dda5fbd5494ce03aaa7c592#diff-c8c936aa2a8d587bef4a4232e0028ed9L63.

As my original proposal violates the basic assumptions of threads and CWD, I think this specific proposal can be closed. Maybe when support becomes better for those functions or if someone has a better idea of how to utilize them we can open up a new issue. Here is a related discussion from 2008 that I found: https://www.ruby-forum.com/topic/165079

#6 - 05/01/2014 10:01 PM - akr (Akira Tanaka)

- Status changed from Open to Rejected

:chdir option for spawn(), system() and IO.popen() is usable to specify the current directory of the child process without changing the current process of the parent process.

```
% pwd
/home/akr
% ruby -e 'system("pwd", :chdir => "/tmp")'
/tmp
```

#7 - 05/02/2014 03:47 AM - nobu (Nobuyoshi Nakada)

And if you want to discard output from the child process:

```
system(command, *args, chdir: dir, out: IO::NULL) # discard stdout only

system(command, *args, chdir: dir, out: IO::NULL, err: [:child, :out]) # also stderr
```