Ruby trunk - Feature #7106

FileUtils.touch should allow touching the symlink itself rather than the file the link points to

10/04/2012 07:32 PM - cirrusthinking (Alessandro Diaferia)

Status:	Open	
Priority:	Normal	
Assignee:		
Target version:		

Description

Most systems support POSIX lutimes sys call that allows a symlink to be touched. I'd like to see this feature implemented. Unfortunately I'd like to see this feature on Solaris as soon as possible but I cannot see lutimes syscall supported there.

History

#1 - 10/11/2012 02:33 AM - kwilczynski (Krzysztof Wilczynski)

Hey,

Definitely a nice to have feature in the standard library :)

And so... few thoughts / ideas ...

On Linux, it would require both the glibc >= 2.6 and kernel >= 2.6.22 since lutimes() on linux uses the utimensat() system-call (and then do_times() which provides actual interface) with the AT_SYMLINK_NOFOLLOW flag set (defined in fcntl.h). In other words, it might cause problems to some people running older / ancient kernels or having old older glibc. Kernel and glibc would also have to support nanoseconds in the "timespec" struct (e.g. tv_nsec, etc) which is what a similar bug touches on: http://bugs.ruby-lang.org/issues/7109

On *BSD, I guess both kernel and libc support it (at least as per the documentation).

Then, on Solaris / OpenSolaris (or anything that does not have necessary capability), we could create a new temporary symbolic link (with safe unpredictable and randomly generated name) and then move it over to clobber old symbolic link rely on the sheer fact that rename()[1] on most Unix-alike systems should be atomic (it is most of the time guaranteed to be such on the same file system) and should be fast (and in a case of any errors we do not loose original link; excluding critical I/O errors or sudden power down, etc).

On Windows... I have no idea :)

Anyway, hope that helps a little :)

1. http://linux.die.net/man/2/rename

KW

#2 - 10/19/2012 03:54 AM - kwilczynski (Krzysztof Wilczynski)

- File 0001-Add-support-for-lutimes-3-to-file.c-as-File-lutime.patch added

- File 0002-Add-support-for-File-lutimes-on-Windows.patch added

- File 0003-Add-support-for-File-lutimes-to-FileUtils.patch added

Hey,

Small updates regarding this :) Support for nanoseconds in Linux kernel for VFS and stat(2) was added around 2.5.38[1]. Then, adding support of lutimes(3) was already mentioned here: http://bugs.ruby-lang.org/issues/4052

Said that, I have made three small patches that try to address lack of File::lutimes and support for symbolic links in FileUtils::touch. There is no support for the trick involving rename(2) mentioned previously (don't think such measures are needed), plus implementation is agnostic from whether an underlying architecture supports nanoseconds for atime, mtime and ctime, or not -- this was addressed to some extent already in the code (although, I am not sure about FAT on Windows, as support for any of these there seems to be rather odd).

Anyhow, I would be very grateful is somebody could review them, especially the Windows one as I am not sure if this would be the correct way of handling how to update access and modification time on Windows. This is my first contribution and I am sure it is far from being perfect, but I did run make test-all on many different versions of Linux distributions with different kernel versions in order to make sure that it works as expected. I need to install NetBSD and FreeBSD and test it there.

I hope that helps :)

1. http://lwn.net/Articles/10634/

KW

#3 - 10/24/2012 10:39 AM - kwilczynski (Krzysztof Wilczynski)

- File 0001-Add-support-for-lutimes.patch added

Hey,

I am adding one large patch, as per Aaron Patterson's request :-)

KW

#4 - 10/25/2012 11:57 AM - usa (Usaku NAKAMURA)

- File 0002-Add-support-for-File-lutimes-on-Windows-updated.patch added

Updated a patch for Windows because the old one includes many errors :)

#5 - 11/06/2012 01:40 PM - luislavena (Luis Lavena)

- Status changed from Open to Assigned

- Assignee set to luislavena (Luis Lavena)

- Target version changed from 1.9.3 to 2.0.0

#6 - 11/07/2012 06:51 AM - luislavena (Luis Lavena)

- Status changed from Assigned to Feedback

=begin

I've combined your patch and Usa's Windows modification (plus other misses) and put it here:

https://gist.github.com/4022459

Also, I've created the following files on a drive:

- foo: directory
- READ: a real file

And created the following type of links:

- bar: symlinkd (directory symlink) of foo
- a-link: symlink of READ
- HARD: hardlink of READ
- junc: a junction point of foo

```
V:>dir
Volume in drive V is RAMDISK
Volume Serial Number is 9A3E-37F6

Directory of V:\

06/11/2012  12:57 a.m.         a-link [READ]
06/11/2012  12:54 a.m.        bar [foo]
05/11/2012  09:25 p.m.            foo
06/11/2012  12:56 a.m.                7 HARD
06/11/2012  01:02 a.m.        junc [V:\foo]
06/11/2012  12:56 a.m.                7 READ
06/11/2012  12:49 a.m.            ruby2
```

As you can see from the output, all have different timestamps

Now, from IRB:

```
V:>irb
irb(main):001:0> File.mtime "foo"
=> 2012-11-05 21:25:54 -0300
irb(main):002:0> File.mtime "bar"
=> 2012-11-06 00:54:22 -0300
irb(main):003:0> File.mtime "junc"
=> 2012-11-06 01:02:03 -0300
```

On Windows, access of modification time gets the link modification time and not the target.

Same goes for atime and ctime:

```
irb(main):004:0> ["foo", "bar", "junc"].each { |folder| puts File.atime(folder) }
2012-11-05 21:25:54 -0300
2012-11-06 00:54:22 -0300
2012-11-06 01:02:03 -0300

irb(main):006:0> ["foo", "bar", "junc"].each { |folder| puts File.ctime(folder) }; nil
2012-11-05 21:23:45 -0300
2012-11-06 00:54:22 -0300
2012-11-06 01:02:03 -0300
```

Now, doing (({File.utime})) on the symlinkd modifies the target:

```
irb(main):016:0> a = File.mtime("foo")
=> 2012-11-07 00:54:22 -0300
irb(main):017:0> a = File.atime("bar")
=> 2012-11-06 00:54:22 -0300
irb(main):018:0> File.utime(a, a, "bar")
=> 1
irb(main):019:0> a = File.atime("bar")
=> 2012-11-06 00:54:22 -0300
irb(main):020:0> a = File.atime("foo")
=> 2012-11-06 00:54:22 -0300
```

But not the symlink, which is expected.

Problem is now that the code around (({rb_file_s_lutime})) avoids it getting defined on Windows, even when functions around it were faked. (it checks for (({HAVE_LUTIMES})) and that doesn't exists on Windows)

I did a minor tweak and got that passing (is included in my patch), and now:

```
V:>irb
irb(main):001:0> a = File.atime("bar")
=> 2012-11-06 00:54:22 -0300
irb(main):002:0> b = a + 3600
=> 2012-11-06 01:54:22 -0300
irb(main):003:0> File.lutime(b, b, "bar")
=> 1
irb(main):004:0> File.atime("bar")
=> 2012-11-06 01:54:22 -0300
irb(main):005:0> File.atime("foo")
=> 2012-11-06 00:54:22 -0300
```

Symlinks can only be created by administrators, but modifying the timestamps (access, modification) is totally possible.

It also works transparently with junction points (which normal users can create too):

```
irb(main):008:0> File.atime("junc")
=> 2012-11-06 01:02:03 -0300
irb(main):009:0> File.atime("foo")
=> 2012-11-06 00:54:22 -0300
irb(main):010:0> File.lutime(b, b, "junc")
=> 1
irb(main):011:0> File.atime("foo")
=> 2012-11-06 00:54:22 -0300
irb(main):012:0> File.atime("junc")
=> 2012-11-06 01:54:22 -0300
```

And file symlinks (previous were directory symlinks, are two different kind of links):

```
irb(main):018:0> File.atime("READ")
=> 2012-11-06 00:56:43 -0300
irb(main):019:0> File.atime("a-link")
=> 2012-11-06 00:57:07 -0300
irb(main):020:0> c = _ + (4 * 3600)
=> 2012-11-06 04:57:07 -0300
irb(main):021:0> File.lutime(c, c, "a-link")
=> 1
irb(main):022:0> File.atime("READ")
=> 2012-11-06 00:56:43 -0300
irb(main):023:0> File.atime("a-link")
=> 2012-11-06 04:57:07 -0300
```

The only one that has no effect are hardlinks, but because the file itself is the same (which is expected)

```
irb(main):024:0> File.atime("READ")
```

```
=> 2012-11-06 00:56:43 -0300
irb(main):025:0> File.atime("HARD")
=> 2012-11-06 00:56:43 -0300
irb(main):026:0> d = _ + (9 * 3600)
=> 2012-11-06 09:56:43 -0300
irb(main):027:0> File.lutime(d, d, "HARD")
=> 1
irb(main):028:0> File.atime("READ")
=> 2012-11-06 09:56:43 -0300
irb(main):029:0> File.atime("HARD")
=> 2012-11-06 09:56:43 -0300
```

All this was performed with a normal command prompt (without admin rights)

Now, (({File.lutime})) seems to work properly, but the modification to (({FileUtils})) might not work.

As you see, FileUtils relies on (({File.symlink?})) to combine with (({:nofollow})) option and then use either (({lutime})) or (({utime})).

Problem is, (({File.symlink?})) always return false on Windows:

```
irb(main):001:0> File.symlink?("foo")
=> false
irb(main):002:0> File.symlink?("bar")
=> false
irb(main):003:0> File.symlink?("a-link")
=> false
irb(main):004:0> File.symlink?("HARD")
=> false
irb(main):005:0> File.symlink?("junc")
```

While current Ruby can create hardlinks, it cannot determine a symlink (the code returns directly (({Qfalse})).

I think that is the final part to get your code into Ruby (at least working on Windows).

Thank you.
=end

#7 - 11/20/2012 04:08 AM - kwilczynski (Krzysztof Wilczynski)

Hey,

Thanks a million Luis! :-)

It would be amazing to get File#lutime added now so *nix / POSIX compliant environment will benefit from it, and then we can look (under a separate ticket) how to add File#symlink? for Windows platform (lack of it is lingering for quite some time now).

Again, thanks for help to everyone who helped :-)

KW

#8 - 11/20/2012 05:12 AM - luislavena (Luis Lavena)

- Assignee changed from luislavena (Luis Lavena) to usa (Usaku NAKAMURA)

Nakamura-san

Are you OK with making File#lutime return a NotImplementedError for now and we can work on next version making File#symlink? and associated work on Windows?

Thank you.

#9 - 11/20/2012 12:34 PM - usa (Usaku NAKAMURA)

- Target version changed from 2.0.0 to 2.6

Yes, OK.
Let's do it on next minor.

#10 - 11/20/2012 12:51 PM - usa (Usaku NAKAMURA)

- Status changed from Feedback to Assigned

#11 - 06/26/2013 06:08 AM - zzak (Zachary Scott)

usa: ping!

#12 - 07/08/2013 10:15 AM - luislavena (Luis Lavena)

Krzysztof Wilczynski, would you mind provide an updated patch that return NotImplementedError on Windows?

Feel free to send a pull request on GitHub and I'll get these bits merged.

Thank you.

#13 - 07/17/2013 11:05 AM - usa (Usaku NAKAMURA)

- Status changed from Assigned to Open

- Assignee deleted (usa (Usaku NAKAMURA))

Ah, I take charge of Windows versions, but I cannot judge whether we should accept this feature request or not.
fileutils doesn't have the maintainer now, can someone judge it?

#14 - 08/02/2013 10:24 AM - kwilczynski (Krzysztof Wilczynski)

When talking about accepting it ... two things to consider as patch has two parts, really:

Part 1 - Adding support for lutimes on *nix/POSIX operating systems[]*;
Part 2 - Changing FileUtils accordingly to accommodate for File getting new methods;

- which can be benefit them right away.

As Luis proven in the past, adding Windows (_WIN32 et al) support might be non-trivial. That having said, I can amend the patch to return NonImplementedError on Windows - not sure how much trouble it is to add support for Windows family (citation needed).

So, even if we consider FileUtils being frozen due to lack of maintainer and Windows implementation for entire Windows family being complex (at this point at least), then perhaps we could separate concerns here and include lutimes patch for *nix/POSIX and return NotImplementedError for Windows for the time being, and then get FileUtils and proper Windows support in check.

What do you guys think?

#15 - 08/05/2013 08:53 AM - zzak (Zachary Scott)

Can you just write a gem?

#16 - 08/11/2013 07:31 PM - kwilczynski (Krzysztof Wilczynski)

In the grand scheme of things FileUtils could be a gem (especially, since it lacks a maintainer now), yes. This probably would turn to be good for it, actually.

As per your suggestion, I am going to do my best and maintain this as a gem for any BSD/POSIX.1 compliant platform. Maybe even with Windows support later (with Luis' blessing).

Edit:

But I would really like to see these system calls added to Ruby. Some of them are almost 10 years old, and it breaks my heart to see Ruby missing them from the Core.

#17 - 11/27/2014 03:51 AM - kwilczynski (Krzysztof Wilczynski)

I would like to resurrect this - what do you think Luis and Usa-san? I am more happy to re-base against Ruby 2.1.5 or head, and try to solve other issues.

Just let me know what needs doing, I would really love to help with this :)

#18 - 12/25/2017 06:15 PM - naruse (Yui NARUSE)

- Target version deleted (2.6)

Files

0001-Add-support-for-lutimes-3-to-file.c-as-File-lutime.patch	6.86 KB	10/19/2012	kwilczynski (Krzysztof Wilczynski)
0002-Add-support-for-File-lutimes-on-Windows.patch	3.9 KB	10/19/2012	kwilczynski (Krzysztof Wilczynski)
0003-Add-support-for-File-lutimes-to-FileUtils.patch	1.92 KB	10/19/2012	kwilczynski (Krzysztof Wilczynski)
0001-Add-support-for-lutimes.patch	10.5 KB	10/24/2012	kwilczynski (Krzysztof Wilczynski)
0002-Add-support-for-File-lutimes-on-Windows-updated.patch	3.33 KB	10/25/2012	usa (Usaku NAKAMURA)