Ruby master - Feature #10181

New method File.openat()

08/28/2014 09:45 PM - technorama (Technorama Ltd.)

Status:	Open	
Priority:	Normal	
Assignee:		
Target version:		

Description

The purpose of the openat() function is to enable opening files in directories other than the current working directory without exposure to race conditions. Any part of the path of a file could be changed in parallel to a call to open(), resulting in unspecified behavior. By opening a file descriptor for the target directory and using the openat() function it can be guaranteed that the opened file is located relative to the desired directory.

openat() is part of POSIX.1-2008.

Compatibility:
Linux kernel >= 2.6.16
FreeBSD >= 7.0
OpenBSD >= 5.0
NetBSD >= 6.1.4
MacOS/X no

Pull request: https://github.com/ruby/ruby/pull/706

Related issues:

Related to Ruby master - Feature #2324: Dir instance methods for relative path	**Assigned**

History

#1 - 08/28/2014 09:55 PM - normalperson (Eric Wong)

I like this feature.

If matz approves, I assume you also want to add other *at functions?
e.g. fstatat, renameat, unlinkat, mkdirat, etc.

#2 - 08/28/2014 11:34 PM - nobu (Nobuyoshi Nakada)

- Related to Feature #2324: Dir instance methods for relative path added

#3 - 08/29/2014 08:03 AM - normalperson (Eric Wong)

Joel VanderWerf joelvanderwerf@gmail.com wrote:

> On 08/28/2014 02:53 PM, Eric Wong wrote:
>
> > I like this feature.
> >
> > If matz approves, I assume you also want to add other *at functions?
> > e.g. fstatat, renameat, unlinkat, mkdirat, etc.
>
> Hm, that suggests...
>
> Dir.at(...).open(...)
> Dir.at(...).fstat(...)

How would that be implemented?

I don't see it working...

The reason for *at functions is the file descriptor points to the
same file (directory) handle across multiple functions; in other words
it's a way to avoid race conditions by creating a private reference
to a container object (an FS directory)

The file descriptor points to the same directory regardless of whether
it's renamed (moved) or not.

One can think of FS operations as operations on Ruby hashes.
In your example, it might be like the following, assuming
"fs" is a giant hash protected by OS-wide locks:

```
# Dir.at(dirname).open("foo")
fs[dirname]["foo"]  # open("/dirname/foo", ...)
                          # another thread may replace/remove
                          # root[dirname] here
# Dir.at(dirname).open("bar")
fs[dirname]["bar"]  # open("/dirname/bar", ...)
```

We cannot guarantee Dir.at(dirname) / fs[dirname] returns
the same value twice when called in succession.

openat lets you work like this:

```
dh = fs[dirname] # dh = opendir(dirname)
dh["foo"] # openat(fileno(dh), "foo", ...)
dh["bar"] # openat(fileno(dh), "bar", ...)
...
```

Other threads can remove/replace/rename fs[dirname] with another
directory, but the directory handle from the initial lookup
remains valid to the thread which opened it.

#4 - 08/29/2014 08:23 AM - normalperson (Eric Wong)

Joel VanderWerf joelvanderwerf@gmail.com wrote:

> On 08/29/2014 12:55 AM, Eric Wong wrote:
>
> > Joel VanderWerf joelvanderwerf@gmail.com wrote:
> >
> > > On 08/28/2014 02:53 PM, Eric Wong wrote:
> > >
> > > > I like this feature.
> > > >
> > > > If matz approves, I assume you also want to add other *at functions?
> > > > e.g. fstatat, renameat, unlinkat, mkdirat, etc.
> > >
> > > Hm, that suggests...
> > >
> > > Dir.at(...).open(...)
> > > Dir.at(...).fstat(...)
> >
> > How would that be implemented?
>
> Couldn't Dir.at(...) return an object that wraps the fd of the dir?

Yes, but it would need to cache the same object every time it's called
given that arg for a given thread. Then it might not detect when that
thread might actually want a different FD/object, and the cache can fill
up or expire and we still end up with unpredictable behavior.

#5 - 08/29/2014 08:44 AM - nobu (Nobuyoshi Nakada)

I don't think it is possible to emulate openat family by FD in user space.
So adding rb_cloexec_open2() is a bad idea, IMHO, not only its name.

#6 - 08/29/2014 08:53 AM - normalperson (Eric Wong)

Joel VanderWerf joelvanderwerf@gmail.com wrote:

> On 08/29/2014 01:21 AM, Eric Wong wrote:
>
> > Joel VanderWerf joelvanderwerf@gmail.com wrote:
> >
> > > On 08/29/2014 12:55 AM, Eric Wong wrote:

> Joel VanderWerf joelvanderwerf@gmail.com wrote:
>
> > On 08/28/2014 02:53 PM, Eric Wong wrote:
> >
> > > I like this feature.
> > >
> > > If matz approves, I assume you also want to add other *at functions?
> > > e.g. fstatat, renameat, unlinkat, mkdirat, etc.
> >
> > Hm, that suggests...
> >
> > Dir.at(...).open(...)
> > Dir.at(...).fstat(...)
>
> How would that be implemented?

Couldn't Dir.at(...) return an object that wraps the fd of the dir?

Yes, but it would need to cache the same object every time it's called given that arg for a given thread. Then it might not detect when that thread might actually want a different FD/object, and the cache can fill up or expire and we still end up with unpredictable behavior.

What if you always used it like this:

d = Dir.at()
d.open()
d.fstat()

so it's up to the caller to decide explicitly when to use the same object or not. It reflects the underlying fd-based API, doesn't it?

OK, that would work. However it ends up creating a new object type and overloading of names, making it harder to review code, I think.
I prefer this:

```
d = Dir.open(..)
d.openat(...)
d.fstatat(..)
```

#7 - 08/29/2014 09:03 AM - normalperson (Eric Wong)

nobu@ruby-lang.org wrote:

> I don't think it is possible to emulate openat family by FD in user space.
> So adding rb_cloexec_open2() is a bad idea, IMHO, not only its name.

Right, we cannot emulate openat; this needs kernel support.

Also, File.new(dir) may not be portable enough for non-Linux. I think this should be based on Dir class instead (using Dir.open(dir), as discussed with Joel).

#8 - 09/03/2014 04:58 AM - funny_falcon (Yura Sokolov)

If you can reuse result of opendir(dirname) why you couldn't reuse result of Dir.at(dirname) ?
29.08.2014 11:55 пользователь "Eric Wong" normalperson@yhbt.net написал:

> Joel VanderWerf joelvanderwerf@gmail.com wrote:
>
> > On 08/28/2014 02:53 PM, Eric Wong wrote:
> >
> > > I like this feature.
> > >
> > > If matz approves, I assume you also want to add other *at functions?
> > > e.g. fstatat, renameat, unlinkat, mkdirat, etc.

> Hm, that suggests...
>
> Dir.at(...).open(...)
> Dir.at(...).fstat(...)

How would that be implemented?

I don't see it working...

The reason for *at functions is the file descriptor points to the
same file (directory) handle across multiple functions; in other words
it's a way to avoid race conditions by creating a private reference
to a container object (an FS directory)

The file descriptor points to the same directory regardless of whether
it's renamed (moved) or not.

One can think of FS operations as operations on Ruby hashes.
In your example, it might be like the following, assuming
"fs" is a giant hash protected by OS-wide locks:

```
# Dir.at(dirname).open("foo")
fs[dirname]["foo"]  # open("/dirname/foo", ...)
                          # another thread may replace/remove
                          # root[dirname] here
# Dir.at(dirname).open("bar")
fs[dirname]["bar"]  # open("/dirname/bar", ...)
```

We cannot guarantee Dir.at(dirname) / fs[dirname] returns
the same value twice when called in succession.

openat lets you work like this:

```
dh = fs[dirname] # dh = opendir(dirname)
dh["foo"] # openat(fileno(dh), "foo", ...)
dh["bar"] # openat(fileno(dh), "bar", ...)
...
```

Other threads can remove/replace/rename fs[dirname] with another
directory, but the directory handle from the initial lookup
remains valid to the thread which opened it.

#9 - 09/03/2014 06:20 AM - normalperson (Eric Wong)

We already have opendir (in the form of Dir.open), so would
Dir.at would be an alias of Dir.open?

I do not like aliases since they makes reading/searching code harder.

But I think we should use Dir.open instead of File.open/File.new for
openat, and also support Dir#fileno:
https://bugs.ruby-lang.org/issues/9880

#10 - 09/03/2014 07:31 AM - funny_falcon (Yura Sokolov)

> But I think we should use Dir.open instead of File.open/File.new for
> openat, and also support Dir#fileno:

Totally agree: it is reasonable to add methods to Dir object for manipulating files relative to directory

#11 - 09/03/2014 07:46 AM - akr (Akira Tanaka)

We should consider other *at functions, as well as openat.

renameat and linkat takes two file descriptors to specify directories.
Also, they may be a special value, AT_FDCWD.

How do we map

renameat(AT_FDCWD, "foo", newfd, "bar") and
renameat(oldfd, "foo", AT_FDCWD, "bar") ?

They are difficult to map Dir methods.

#12 - 09/03/2014 09:29 AM - normalperson (Eric Wong)

akr@fsij.org wrote:

> We should consider other *at functions, as well as openat.
>
> renameat and linkat takes two file descriptors to specify directories.
> Also, they may be a special value, AT_FDCWD.
>
> How do we map
> renameat(AT_FDCWD, "foo", newfd, "bar") and
> renameat(oldfd, "foo", AT_FDCWD, "bar") ?
>
> They are difficult to map Dir methods.

IO.copy_stream is similar, I think:

```
d1 = Dir.open("d1")
d2 = Dir.open("d2")

# allow d1/d2 to be Fixnum for fileno, too
Dir.renameat(d1, "foo", d2, "bar")
Dir.renameat(Dir::AT_FDCWD, "foo", d2.fileno, "bar")
```

However, the following defeats the purpose of renameat, so I am somewhat
against the following (both string args where FDs should be):

```
Dir.renameat("d1/", "foo", "d2/", "bar")
```

Maybe allowing string path for one (not both) FD arg is not too bad:

```
Dir.renameat(Dir::AT_FDCWD, "foo", "d2/", "bar")
Dir.renameat("d1/", "foo", d2.fileno, "bar")
```

#13 - 09/03/2014 02:55 PM - technorama (Technorama Ltd.)

The proposed Dir api must provide a way to open both files and directories in order to be useful.

New proposal:

d1 = Dir.open('d1') => aDir
d2 = d1.open('subdir') => aDir relative to d1
file = d2.open_file('file') => aFile relative to d2

I plan on handling all other *at functions in a gem so that all ruby implementations can share the same implementation. The only methods that must be included in mri are dir.open and dir.open_file due to limitations of the c api.

#14 - 09/05/2014 03:55 AM - funny_falcon (Yura Sokolov)

May be it is not a best option, but just to be considered:

```
d1 = Dir.open('d1') => aDir
d2 = d1.opendir('subdir') => aDir relative to d1
file = d2.open('file') => aFile relative to d2
d1.rename_at("foo", d2,"bar")
Dir::AT_FDCWD.rename_at("foo", d1, "bar")
```

#15 - 08/09/2015 09:55 AM - technorama (Technorama Ltd.)

No movement in almost a year. Will this proposal be accepted?

#16 - 10/22/2015 12:12 PM - shugo (Shugo Maeda)

Technorama Ltd. wrote:

> The proposed Dir api must provide a way to open both files and directories in order to be useful.
>
> New proposal:

d1 = Dir.open('d1') => aDir
d2 = d1.open('subdir') => aDir relative to d1
file = d2.open_file('file') => aFile relative to d2

I prefer the following style:

```
d1 = Dir.open("d1") #=> a Dir
d2 = Dir.openat(d1, "subdir") #=> a Dir relative to d1
file = File.openat(d2, "file") #=> a File relative to d2
```

Because it is clear that these methods call openat(2), and at is a preposition.
Other *at functions such as renameat() can be consistently implemented as singleton methods of File.