Users of F_SETLKW may block the entire VM via IO#fcntl, release the GVL so other operations may continue.

Associated revisions

Revision c0359f81 - 03/04/2011 04:38 PM - kosaki (Motohiro KOSAKI)

- io.c (io_cntl, nogvl_io_cntl): IO.fcntl() and IO.ioctl() release GVL during calling kernel interface.
  Suggested by Eric Wong. [ruby-core:35417][Bug #4463]

- test/ruby/test_io.rb (TestIO#test_fcntl_lock): add new test for IO.fcntl().

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@31025 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 31025 - 03/04/2011 04:38 PM - kosaki (Motohiro KOSAKI)

- io.c (io_cntl, nogvl_io_cntl): IO.fcntl() and IO.ioctl() release GVL during calling kernel interface.
  Suggested by Eric Wong. [ruby-core:35417][Bug #4463]

- test/ruby/test_io.rb (TestIO#test_fcntl_lock): add new test for IO.fcntl().

Revision 31025 - 03/04/2011 04:38 PM - kosaki (Motohiro KOSAKI)

- io.c (io_cntl, nogvl_io_cntl): IO.fcntl() and IO.ioctl() release GVL during calling kernel interface.
  Suggested by Eric Wong. [ruby-core:35417][Bug #4463]

- test/ruby/test_io.rb (TestIO#test_fcntl_lock): add new test for IO.fcntl().

Revision 31025 - 03/04/2011 04:38 PM - kosaki (Motohiro KOSAKI)

- io.c (io_cntl, nogvl_io_cntl): IO.fcntl() and IO.ioctl() release GVL during calling kernel interface.
Suggested by Eric Wong. [ruby-core:35417][Bug #4463]

- test/ruby/test_io.rb (TestIO#test_fcntl_lock): add new test for IO.fcntl().

Revision 31025 - 03/04/2011 04:38 PM - kosaki (Motohiro KOSAKI)

- io.c (io_cntl, nogvl_io_cntl): IO.fcntl() and IO.ioctl() release GVL during calling kernel interface.
  Suggested by Eric Wong. [ruby-core:35417][Bug #4463]

- test/ruby/test_io.rb (TestIO#test_fcntl_lock): add new test for IO.fcntl().

Revision 31025 - 03/04/2011 04:38 PM - kosaki (Motohiro KOSAKI)

- io.c (io_cntl, nogvl_io_cntl): IO.fcntl() and IO.ioctl() release GVL during calling kernel interface.
  Suggested by Eric Wong. [ruby-core:35417][Bug #4463]

- test/ruby/test_io.rb (TestIO#test_fcntl_lock): add new test for IO.fcntl().

Revision 31025 - 03/04/2011 04:38 PM - kosaki (Motohiro KOSAKI)

- io.c (io_cntl, nogvl_io_cntl): IO.fcntl() and IO.ioctl() release GVL during calling kernel interface.
  Suggested by Eric Wong. [ruby-core:35417][Bug #4463]

- test/ruby/test_io.rb (TestIO#test_fcntl_lock): add new test for IO.fcntl().

History

#1 - 03/04/2011 12:23 AM - kosaki (Motohiro KOSAKI)

- ruby -v changed from ruby 1.9.3dev (2011-03-03 trunk 31011) [x86_64-linux] to -

=begin

Issue #4463 has been reported by Eric Wong.

Bug #4463: [PATCH] release GVL for fcntl() for operations that may block
http://redmine.ruby-lang.org/issues/4463

Author: Eric Wong
Status: Open
Priority: Normal
Assignee:
Category: core
Target version: 1.9.x
ruby -v: ruby 1.9.3dev (2011-03-03 trunk 31011) [x86_64-linux]

Users of F_SETLKW may block the entire VM via IO#fcntl,
release the GVL so other operations may continue.

=end

09/22/2021
Yeah.
It looks reasonable request. :)
=end

#2 - 03/05/2011 02:24 AM - kosaki (Motohiro KOSAKI)
=begin
Hi
2011/3/3 KOSAKI Motohiro kosaki.motohiro@gmail.com:

Issue #4463 has been reported by Eric Wong.

Bug #4463: [PATCH] release GVL for fcntl() for operations that may block
http://redmine.ruby-lang.org/issues/4463

Author: Eric Wong
Status: Open
Priority: Normal
Assignee:
Category: core
Target version: 1.9.x
ruby -v: ruby 1.9.3dev (2011-03-03 trunk 31011) [x86_64-linux]

Users of F_SETLKW may block the entire VM via IO#fcntl,
release the GVL so other operations may continue.

Yeah.
It looks reasonable request. :)

Hi

I've commited slightly modified version today (r31025).
The difference is,

1) All IO.fcntl() and IO.iocntl() release GVL instead only SETLCKW. because,
A) if a user are using network filesystem, almost all fcntl need network
communication. iow, they can be blocked.
B) We are sure ioctl() has similar issue. But, we don't have any knowledge
which ioctl can be blocked. It is strongly dependend a
platform and a device.
2) Added small test. It is based on your Fcntl::Flock patch.

Thanks.
=end

#3 - 03/05/2011 02:24 AM - normalperson (Eric Wong)
=begin
KOSAKI Motohiro kosaki.motohiro@gmail.com wrote:

Hi

I've commited slightly modified version today (r31025).
The difference is,

1) All IO.fcntl() and IO.iocntl() release GVL instead only SETLCKW. because,
A) if a user are using network filesystem, almost all fcntl need network
communication. iow, they can be blocked.
B) We are sure ioctl() has similar issue. But, we don't have any knowledge
which ioctl can be blocked. It is strongly dependend a
platform and a device.
2) Added small test. It is based on your Fcntl::Flock patch.

Agreed on both points.

2) Added small test. It is based on your Fcntl::Flock patch.

Any chance of that patch making it into trunk? I'd be happy to make
any changes/improvements necessary (+docs, too). Thanks again.
Hi

I've committed slightly modified version today (r31025).
The difference is,

1) All IO.fcntl() and IO.iocntl() relese GVL instead only SETLCKW. because,
   
   A  Â A) if a user are using network filesystem, almost all fcntl need network
   A  Â A  Â communication. low, they can be blocked.
   A  Â B) We are sure ioctl has similar issue. But, we don't have any knowledge
   A  Â A  Â which ioctl can be blocked. It is strongly dependend a
   platform and a device.

Agreed on both points.

thank you.

2) Added small test. It is based on your Fcntl::Flock patch.

   Any chance of that patch making it into trunk? Â I'd be happy to make
   any changes/improvements necessary (+docs, too). Â Thanks again.

Umm..
I don't like its interface so much. your flock object don't mange any lock
state. it's merely wrapper of argument of fcntl. your interface mean we need
two line every lock operation. eg.

        lock

I agree it's currently too verbose.

I tried to keep io.c the same so I used a String subclass. Maybe I
should just modify teach io.c to deal with Hash/Array arguments? I do
worry about placing more burden on io.c for portability reasons, though
POSIX file locks might be very common by now...

To shorten interface, maybe Fcntl::Flock[] can return an array for splat
and take symbol args (like new Socket):

        f.fcntl *Fcntl::Flock[:F_SETLKW, :F_WRLCK, SEEK_SET, 0, 0]

Or maybe even:

        f.fcntl *Fcntl::Flock[:SETLKW, :WRLCK, :SET, 0, 0]

but I personally prefer array or hash capsulation. eg

        f.fcntl Fcntl::F_SETLKW, [Fcntl::F_WRLK, SEEK_SET, 0, 0]
or

ffcntl Fcntl::F_SETLKW, [:l_type => Fcntl::F_WRLK]

Yes, I like the Hash one but requires modifying io.c with potentially unportable code to support.

If we use non-String, maybe just call fcntl(2) inside ext/fcntl/fcntl.c internally and forget about IO::fcntl in io.c entirely:

Fcntl::Flock[:WRLCK, :SET, 0, 0].lock(io)
Fcntl::Flock[:WRLCK, :SET, 0, 0].try_lock(io)
Fcntl::Flock[:SET, 0, 0].unlock(io)

Or even:

Fcntl::Flock[:WRLCK, :SET, 0, 0].lock(io)
Fcntl::Flock[:WRLCK, :SET, 0, 0].try_lock(io)
Fcntl::Flock[:SET, 0, 0].unlock(io)

That would allow us to do something stateful like:

Fcntl::Flock[:WRLCK, :SET, 0, 0].lock(io)
Fcntl::Flock[:WRLCK, :SET, 0, 0].try_lock(io)
Fcntl::Flock[:SET, 0, 0].unlock(io)

Or even:

Fcntl::Flock[:WRLCK, :SET, 0, 0].lock(io)
Fcntl::Flock[:WRLCK, :SET, 0, 0].try_lock(io)
Fcntl::Flock[:SET, 0, 0].unlock(io)

That would allow us to do something stateful like:

Fcntl::Flock[:WRLCK, :SET, 0, 0].lock(io)
Fcntl::Flock[:WRLCK, :SET, 0, 0].try_lock(io)
Fcntl::Flock[:SET, 0, 0].unlock(io)

That would allow us to do something stateful like:

Fcntl.lock(io, :WRLCK, :SET, 0, 0)
Fcntl.try_lock(io, :WRLCK, :SET, 0, 0)
Fcntl.unlock(io, :SET, 0, 0)

Fcntl::Flock[:WRLCK, :SET, 0, 0].lock(io)
Fcntl::Flock[:WRLCK, :SET, 0, 0].try_lock(io)
Fcntl::Flock[:SET, 0, 0].unlock(io)

That would allow us to do something stateful like:

Fcntl::Flock[:WRLCK, :SET, 0, 0].lock(io)
Fcntl::Flock[:WRLCK, :SET, 0, 0].try_lock(io)
Fcntl::Flock[:SET, 0, 0].unlock(io)

That would allow us to do something stateful like:

Fcntl.synchronize(io, :WRLCK, :SET, 0, 0) do
  # ...
end

I dislike all caps, even, taking hints from pthread_rwlock_*.:

Fcntl::Flock[:WRLCK, :SET, 0, 0].lock(io)
Fcntl::Flock[:WRLCK, :SET, 0, 0].try_lock(io)
Fcntl::Flock[:SET, 0, 0].unlock(io)

Fcntl::Flock[:WRLCK, :SET, 0, 0].lock(io)
Fcntl::Flock[:WRLCK, :SET, 0, 0].try_lock(io)
Fcntl::Flock[:SET, 0, 0].unlock(io)

Fcntl::Flock[:WRLCK, :SET, 0, 0].lock(io)
Fcntl::Flock[:WRLCK, :SET, 0, 0].try_lock(io)
Fcntl::Flock[:SET, 0, 0].unlock(io)

That would allow us to do something stateful like:

Fcntl.synchronize(io, :WRLCK, :SET, 0, 0) do
  # ...
end

But, of course, I'm not against if matz ack yours. So I recommend you describe the detailed interface to matz instead only just attached a patch. It's best practice to persuade very busy person. :)

Thanks again for the feedback. So many ways to do this interface, but just anything but Array#pack sounds good to me :)
That would allow us to do something stateful like:

```ruby
Fcntl.synchronize(io, :WRLCK, :SET, 0, 0) do
  # ...
end
```

Following up, I went with something along these lines here.

http://redmine.ruby-lang.org/issues/4464

Simple use case to lock the whole file is just:

```ruby
Fcntl::Lock.synchronize(file) do
  # ...
end
```

--
Eric Wong
=end

#8 - 04/12/2011 08:17 PM - kosaki (Motohiro KOSAKI)
=begin
Issue #4463 has been reported by Eric Wong.

Bug #4463: [PATCH] release GVL for fcntl() for operations that may block
http://redmine.ruby-lang.org/issues/4463

Author: Eric Wong
Status: Open
Priority: Normal
Assignee:
Category: core
Target version: 1.9.x
ruby -v: ruby 1.9.3dev (2011-03-03 trunk 31011) [x86_64-linux]

Users of F_SETLKW may block the entire VM via IO#fcntl,
release the GVL so other operations may continue.

Yeah.
It looks reasonable request. ;)
=end

#9 - 04/12/2011 08:17 PM - kosaki (Motohiro KOSAKI)
=begin
Hi

2011/3/3 KOSAKI Motohiro kosaki.motohiro@gmail.com:

Issue #4463 has been reported by Eric Wong.

Bug #4463: [PATCH] release GVL for fcntl() for operations that may block
http://redmine.ruby-lang.org/issues/4463

Author: Eric Wong
Status: Open
Priority: Normal
Assignee:
Category: core
Target version: 1.9.x
ruby -v: ruby 1.9.3dev (2011-03-03 trunk 31011) [x86_64-linux]

Users of F_SETLKW may block the entire VM via IO#fcntl,
release the GVL so other operations may continue.

Yeah.

09/22/2021
6/13
Hi

I've committed slightly modified version today (r31025).
The difference is,

1) All IO.fcntl() and IO.iocntl() release GVL instead only SETLCKW. because,
A) if a user are using network filesystem, almost all fcntl need network communication. iow, they can be blocked.
B) We are sure ioctl() has similar issue. But, we don't have any knowledge which ioctl can be blocked. It is strongly dependend a platform and a device.
2) Added small test. It is based on your Fcntl::Flock patch.

Agreed on both points.

thank you.

2) Added small test. It is based on your Fcntl::Flock patch.

Any chance of that patch making it into trunk? I'd be happy to make any changes/improvements necessary (+docs, too). Thanks again.

--
Eric Wong

#11 - 04/12/2011 08:17 PM - kosaki (Motohiro KOSAKI)
=begin

I've committed slightly modified version today (r31025).
The difference is,

1) All IO.fcntl() and IO.iocntl() release GVL instead only SETLCKW. because,
A) if a user are using network filesystem, almost all fcntl need network communication. iow, they can be blocked.
B) We are sure ioctl() has similar issue. But, we don't have any knowledge which ioctl can be blocked. It is strongly dependend a platform and a device.

Agreed on both points.

thank you.

2) Added small test. It is based on your Fcntl::Flock patch.

Any chance of that patch making it into trunk? I'd be happy to make any changes/improvements necessary (+docs, too). Thanks again.
Umm...
I don't like its interface so much. your flock object don't mange any lock state. it's merely wrapper of argument of fcntl. your interface mean we need two line every lock operation. eg.

```ruby
lock
=end
```

#12 - 04/12/2011 08:17 PM - normalperson (Eric Wong)

begin
KOSAKI Motohiro kosaki.motohiro@gmail.com wrote:

Umm...
I don't like its interface so much. your flock object don't mange any lock state. it's merely wrapper of argument of fcntl. your interface mean we need two line every lock operation. eg.

```ruby
lock = Fcntl::Flock.new Fcntl::F_WRLCK
f.fcntl Fcntl::F_SETLKW, lock
```

I agree it's currently too verbose.
I tried to keep io.c the same so I used a String subclass. Maybe I should just modify teach io.c to deal with Hash/Array arguments? I do worry about placing more burden on io.c for portability reasons, though POSIX file locks might be very common by now...

To shorten interface, maybe Fcntl::Flock[] can return an array for splat and take symbol args (like new Socket):

```ruby
f.fcntl *Fcntl::Flock[:F_SETLKW, :F_WRLCK, :SEEK_SET, 0, 0]
```

Or maybe even:

```ruby
f.fcntl *Fcntl::Flock[:SETLKW, :WRLCK, :SET, 0, 0]
```

but I personally prefer array or hash capsulation. e.g

```ruby
f.fcntl Fcntl::F_SETLKW, [Fcntl::F_WRLK, SEEK_SET, 0, 0]
```

or

```ruby
f.fcntl Fcntl::F_SETLKW, { :l_type => Fcntl::F_WRLK }
```

Yes, I like the Hash one but requires modifying io.c with potentially unportable code to support.

If we use non-String, maybe just call fcntl(2) inside ext/fcntl/fcntl.c internally and forget about IO#fcntl in io.c entirely:

```ruby
Fcntl::Flock[:WRLCK, :SET, 0, 0].lock(io)
Fcntl::Flock[:WRLCK, :SET, 0, 0].try_lock(io)
Fcntl::Flock[:SET, 0, 0].unlock(io)
```

Or even:

```ruby
Fcntl.lock(io, :WRLCK, :SET, 0, 0)
Fcntl.try_lock(io, :WRLCK, :SET, 0, 0)
Fcntl.unlock(io, :SET, 0, 0)
Fcntl.getlock(io, :RDLCK, :SET, 0, 0) -> Fcntl::Flock object
```

That would allow us to do something stateful like:

```ruby
Fcntl.synchronize(io, :WRLCK, :SET, 0, 0) do
  # ...
end
```

I dislike all caps, even, taking hints from pthread_rwlock_*:

```ruby
Fcntl.rdlock(io, :set, 0, 0)
```
fcntl.tryrdlock(io, :set, 0, 0)
fcntl.wrlock(io, :set, 0, 0)
fcntl.trywrlock(io, :set, 0, 0)
fcntl.unlock(io, :set, 0, 0)

fcntl.read_synchronize(io, :set, 0, 0) do
  # ...
end

fcntl.synchronize(io, :set, 0, 0) do
  # ...
end

But, of course, I'm not against if matz ack yours. So I recommend you
describe the detailed interface to matz instead only just attached a patch.
It's best practice to persuade very busy person. :)

Thanks again for the feedback. So many ways to do this interface,
but just anything but Array#pack sounds good to me :)

--
Eric Wong
=end

#13 - 04/12/2011 08:17 PM - normalperson (Eric Wong)
=begin
Eric Wong normalperson@yhbt.net wrote:

KOSAKI Motohiro kosaki.motohiro@gmail.com wrote:

  Umm..
  I don't like its interface so much. your flock object don't mange any lock
  state. it's merely wrapper of argument of fcntl. your interface mean we need
  two line every lock operation. eg.

  That would allow us to do something stateful like:

  fcntl.synchronize(io, :WRLCK, :SET, 0, 0) do
    # ...
  end

Following up, I went with something along these lines here.

http://redmine.ruby-lang.org/issues/4464

Simple use case to lock the whole file is just:

fcntl::Lock.synchronize(file) do
  # ...
end

--
Eric Wong
=end

#14 - 04/12/2011 08:18 PM - kosaki (Motohiro KOSAKI)
=begin

Issue #4463 has been reported by Eric Wong.

Bug #4463: [PATCH] release GVL for fcntl() for operations that may block
http://redmine.ruby-lang.org/issues/4463

Author: Eric Wong
Status: Open
Priority: Normal
Assignee:
Category: core
Users of F_SETLKW may block the entire VM via IO#fcntl, release the GVL so other operations may continue.

Yeah. It looks reasonable request. ;)

Hi

2011/3/3 KOSAKI Motohiro kosaki.motohiro@gmail.com:

Issue #4463 has been reported by Eric Wong.

Bug #4463: [PATCH] release GVL for fcntl() for operations that may block
http://redmine.ruby-lang.org/issues/4463

Author: Eric Wong
Status: Open
Priority: Normal
Assignee:
Category: core
Target version: 1.9.x
ruby -v: ruby 1.9.3dev (2011-03-03 trunk 31011) [x86_64-linux]

Users of F_SETLKW may block the entire VM via IO#fcntl, release the GVL so other operations may continue.

Yeah.
It looks reasonable request. ;)

Hi

I've committed slightly modified version today (r31025).
The difference is,

1) All IO.fcntl() and IO.iocntl() release GVL instead only SETLCKW. because,
   A) if a user are using network filesystem, almost all fcntl need network
      communication. iow, they can be blocked.
   B) We are sure ioctl() has similar issue. But, we don't have any knowledge
      which ioctl can be blocked. It is strongly depend end a
      platform and a device.

2) Added small test. It is based on your Fcntl::Flock patch.

Thanks.

Agreed on both points.
2) Added small test. It is based on your Fcntl::Flock patch.

Any chance of that patch making it into trunk? I'd be happy to make any changes/improvements necessary (+docs, too). Thanks again.

---
Eric Wong
=end

#17 - 04/12/2011 08:18 PM - kosaki (Motohiro KOSAKI)
=begin

I've committed slightly modified version today (r31025).
The difference is,

1) All IO.fcntl() and IO.iocntl() release GVL instead only SETLCKW. because,
   A) if a user are using network filesystem, almost all fcntl need network
      communication. iow, they can be blocked.
   B) We are sure ioctl() has similar issue. But, we don't have any knowledge
      which ioctl can be blocked. It is strongly dependend a platform and a device.

Agreed on both points.

thank you.

2) Added small test. It is based on your Fcntl::Flock patch.

Any chance of that patch making it into trunk? Â I'd be happy to make any changes/improvements necessary (+docs, too). Â Thanks again.

Umm..
I don't like its interface so much. your flock object don't mange any lock state. it's merely wrapper of argument of fcntl. your interface mean we need two line every lock operation. eg.

```
lock
```
=end

#18 - 04/12/2011 08:18 PM - normalperson (Eric Wong)
=begin

KOSAKI Motohiro kosaki.motohiro@gmail.com wrote:

Umm..
I don't like its interface so much. your flock object don't mange any lock state. it's merely wrapper of argument of fcntl. your interface mean we need two line every lock operation. eg.

```
lock = Fcntl::Flock.new Fcntl::F_WRLCK
f.fcntl Fcntl::F_SETLKW, lock
```

I agree it's currently too verbose.

I tried to keep io.c the same so I used a String subclass. Maybe I should just modify teach io.c to deal with Hash/Array arguments? I do worry about placing more burden on io.c for portability reasons, though POSIX file locks might be very common by now...

To shorten interface, maybe Fcntl::Flock[] can return an array for splat and take symbol args (like new Socket):

```
f.fcntl *Fcntl::Flock[:F_SETLKW, :F_WRLCK, :SEEK_SET, 0, 0]
```

Or maybe even:
f.fcntl *Fcntl::Flock[:SETLKW, :WRLCK, :SET, 0, 0]

but I personally prefer array or hash capsulation. e.g

f.fcntl Fcntl::F_SETLKW, [Fcntl::F_WRLK, SEEK_SET, 0, 0]

or

f.fcntl Fcntl::F_SETLKW, { :d_type => Fcntl::F_WRLK }

Yes, I like the Hash one but requires modifying io.c with potentially unportable code to support.

If we use non-String, maybe just call fcntl(2) inside ext/fcntl/fcntl.c internally and forget about IO::fcntl in io.c entirely:

Fcntl::Flock[:WRLCK, :SET, 0, 0].lock(io)
Fcntl::Flock[:WRLCK, :SET, 0, 0].try_lock(io)
Fcntl::Flock[:SET, 0, 0].unlock(io)

Or even:

Fcntl.lock(io, :WRLCK, :SET, 0, 0)
Fcntl.try_lock(io, :WRLCK, :SET, 0, 0)
Fcntl.unlock(io, :SET, 0, 0)

Fcntl.getlock(io, :RDLCK, :SET, 0, 0) -> Fcntl::Flock object

That would allow us to do something stateful like:

Fcntl.synchronize(io, :WRLCK, :SET, 0, 0) do
  # ...
end

I dislike all caps, even, taking hints from pthread_rwlock_*:

Fcntl.rdlock(io, :set, 0, 0)
Fcntl.tryrdlock(io, :set, 0, 0)
Fcntl.wrlock(io, :set, 0, 0)
Fcntl.trywrlock(io, :set, 0, 0)
Fcntl.unlock(io, :set, 0, 0)

Fcntl.read_synchronize(io, :set, 0, 0) do
  # ...
end

Fcntl.synchronize(io, :set, 0, 0) do
  # ...
end

But, of course, I'm not against if matz ack yours. So I recommend you describe the detailed interface to matz instead only just attached a patch. It's best practice to persuade very busy person. :)

Thanks again for the feedback. So many ways to do this interface, but just anything but Array#pack sounds good to me :)

--
Eric Wong
=end

#19 - 04/12/2011 08:18 PM - normalperson (Eric Wong)

=end

But, of course, I'm not against if matz ack yours. So I recommend you describe the detailed interface to matz instead only just attached a patch. It's best practice to persuade very busy person. :)

Thanks again for the feedback. So many ways to do this interface, but just anything but Array#pack sounds good to me :)

--
Eric Wong
=end

#19 - 04/12/2011 08:18 PM - normalperson (Eric Wong)

=end

But, of course, I'm not against if matz ack yours. So I recommend you describe the detailed interface to matz instead only just attached a patch. It's best practice to persuade very busy person. :)

Thanks again for the feedback. So many ways to do this interface, but just anything but Array#pack sounds good to me :)

--
Eric Wong
=end
That would allow us to do something stateful like:

```ruby
Fcntl.synchronize(io, :WRLCK, :SET, 0, 0) do
  # ...
end
```

Following up, I went with something along these lines here.

http://redmine.ruby-lang.org/issues/4464

Simple use case to lock the whole file is just:

```ruby
Fcntl::Lock.synchronize(file) do
  # ...
end
```

--

Eric Wong
=end

<table>
<thead>
<tr>
<th>Files</th>
<th></th>
<th></th>
<th></th>
<th>03/03/2011</th>
<th>normalperson (Eric Wong)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001-release-GVL-for-fcntl-for-operations-that-may-block.patch</td>
<td>1.7 KB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>