Ruby - Feature #1961

Kernel#__dir__

08/19/2009 11:57 PM - yhara (Yutaka HARA)

Status:	Closed
Priority:	Normal
Assignee:	matz (Yukihiro Matsumoto)
Target version:	2.0.0

Description

=begin
= Proposal

Kernel#**dir**
returns the value of File.dirname(**FILE**)

According to the google code search, about 60% of uses of **FILE**
are of the form File.dirname(**FILE**). Ruby 1.9.2 provides
require_relative for this problem; but File.dirname(**FILE**) is
not always used for requiring ruby scripts, but also for reading
data files. **dir** helps these cases.

(Note: my proposal does not include **dir_dir** this time :-)
It should be discussed in another threads)
Related ticket: http://redmine.ruby-lang.org/issues/show/642

= Problem

File.dirname(**FILE**) is frequently used and too long.

= Analysis

There are 222 uses of **FILE** listed by the google code search,
classified into these categories:

(A) 30.6% (68) are used with require and File.dirname.
In Ruby 1.9.2, this case is supported by require_relative.

(B) 31.1% (69) are with File.dirname, but not with require.
For example, reading data files of unit tests.

(C) 21.6% (48) are the idiom, $0 == **FILE**.

B is as many as A (and even more than C), so it is reasonable to
add a shortcut for File.dirname(**FILE**) in addition to require_relative.

- code: http://gist.github.com/170336
- result: http://route477.net/files/__file__.html

= Solutions

(1) add a toplevel constant **DIR**
pros: looks like **FILE**
cons: adding new keyword

(2) add Kernel#**DIR**
pros: no new keyword
cons: it should have a lower-case name (like 1.9's **method**),
because it is not a constant but a method.

(3) add Kernel#**dir**
pros: no new keyword, and it is clearer than (4) that it has
some associations with **FILE**.

(4) make **FILE** to the default argument of File.dirname
pros: no new keyword nor new method
cons: it is not clear that 'File.dirname' is expanded to
the path of directory of **FILE**.

= Conclusion

I think (3) (Kernel#**dir**) is the best.

Thanks,

yhara (Yutaka HARA)
http://route477.net/
=end

Related issues:

Related to Ruby - Feature #642: __DIR__	**Rejected**	**10/14/2008**
Has duplicate Ruby - Feature #3346: __DIR__ revisted	**Closed**	**05/26/2010**

History

#1 - 08/20/2009 12:20 PM - authorNari (Narihiro Nakamura)

=begin
Hi.

I think nice idea and interesting proposal.

The following patch add Kernel#**dir**.
(from Nobuyoshi Nakada, thanks)

Index: prelude.rb

--- prelude.rb (revision 24591)
+++ prelude.rb (working copy)
@@ -24,12 +24,19 @@ class Thread
end

-def require_relative(relative_feature)

- c = caller.first
  +def **dir**(*args)
- c = caller(*args).first*
  e = c.rindex(/:\d+:in /)
  file = $`
  if /\A((.))/ =~ file # eval, etc.
- raise LoadError, "require_relative is called in #{$1}"
- file = yield($1)
  end
- absolute_feature = File.expand_path(File.join(File.dirname(file), relative_feature))
- File.dirname(file)
  +end

+def require_relative(relative_feature)

- dir = **dir**(2) {|file|
- raise LoadError, "require_relative is called in #{file}"
- }
- absolute_feature = File.expand_path(File.join(dir, relative_feature))
  require absolute_feature
  end

=end

#2 - 08/30/2009 12:00 AM - matz (Yukihiro Matsumoto)

=begin
Hi,

In message "Re: [ruby-core:24982] [Feature #1961] Kernel#**dir**"
on Wed, 19 Aug 2009 23:57:07 +0900, Yutaka HARA redmine@ruby-lang.org writes:

|= Proposal
|
| Kernel#**dir**
| returns the value of File.dirname(**FILE**)
|
|According to the google code search, about 60% of uses of **FILE**
|are of the form File.dirname(**FILE**). Ruby 1.9.2 provides
|require_relative for this problem; but File.dirname(**FILE**) is
|not always used for requiring ruby scripts, but also for reading
|data files. **dir** helps these cases.

I accept the proposal. If a patch comes, I'd love to merge it in.

```
      matz.
```

=end

#3 - 08/30/2009 02:57 AM - judofyr (Magnus Holm)

=begin
Wouldn't it be a little confusing to remember that **FILE** is uppercase and
dir is downcase? Doesn't sound very POLS to me...

//Magnus Holm

On Wed, Aug 19, 2009 at 16:57, Yutaka HARA redmine@ruby-lang.org wrote:

> Feature #1961: Kernel#**dir**
> http://redmine.ruby-lang.org/issues/show/1961
>
> Author: Yutaka HARA
> Status: Open, Priority: Normal
>
> = Proposal
>
> Kernel#**dir**
> returns the value of File.dirname(**FILE**)
>
> According to the google code search, about 60% of uses of **FILE**
> are of the form File.dirname(**FILE**). Ruby 1.9.2 provides
> require_relative for this problem; but File.dirname(**FILE**) is
> not always used for requiring ruby scripts, but also for reading
> data files. **dir** helps these cases.
>
> (Note: my proposal does not include **dir_dir** this time :-)
> It should be discussed in another threads)
> Related ticket: http://redmine.ruby-lang.org/issues/show/642
>
> = Problem
>
> File.dirname(**FILE**) is frequently used and too long.
>
> = Analysis
>
> There are 222 uses of **FILE** listed by the google code search,
> classified into these categories:
>
> (A) 30.6% (68) are used with require and File.dirname.
> In Ruby 1.9.2, this case is supported by require_relative.
>
> (B) 31.1% (69) are with File.dirname, but not with require.
> For example, reading data files of unit tests.
>
> (C) 21.6% (48) are the idiom, $0 == **FILE**.
>
> B is as many as A (and even more than C), so it is reasonable to
> add a shortcut for File.dirname(**FILE**) in addition to require_relative.
>
> - code: http://gist.github.com/170336

- result: http://route477.net/files/__file__.html

= Solutions

(1) add a toplevel constant **DIR**
pros: looks like **FILE**
cons: adding new keyword

(2) add Kernel#**DIR**
pros: no new keyword
cons: it should have a lower-case name (like 1.9's **method**),
because it is not a constant but a method.

(3) add Kernel#**dir**
pros: no new keyword, and it is clearer than (4) that it has
some associations with **FILE**.

(4) make **FILE** to the default argument of File.dirname
pros: no new keyword nor new method
cons: it is not clear that 'File.dirname' is expanded to
the path of directory of **FILE**.

= Conclusion

I think (3) (Kernel#**dir**) is the best.

Thanks,

yhara (Yutaka HARA)
http://route477.net/

---

http://redmine.ruby-lang.org

Wouldn't it be a little confusing to remember that **FILE** is uppercase and **dir** is downcase? Doesn't sound very POLS to me...

//Magnus Holm

On Wed, Aug 19, 2009 at 16:57, Yutaka HARA <redmine@ruby-lang.org> wrote:

Feature #1961: Kernel#**dir**

http://redmine.ruby-lang.org/issues/show/1961

Author: Yutaka HARA

Status: Open, Priority: Normal

= Proposal

Kernel#**dir**

returns the value of File.dirname(**FILE**)

According to the google code search, about 60% of uses of **FILE**

are of the form File.dirname(**FILE**). Ruby 1.9.2 provides

require_relative for this problem; but File.dirname(**FILE**) is

not always used for requiring ruby scripts, but also for reading

data files. **dir** helps these cases.

(Note: my proposal does not include **dir_dir** this time :-)

It should be discussed in another threads)

Related ticket: http://redmine.ruby-lang.org/issues/show/642

= Problem

File.dirname(**FILE**) is frequently used and too long.

= Analysis

There are 222 uses of **FILE** listed by the google code search,

classified into these categories:

(A) 30.6% (68) are used with require and File.dirname.

In Ruby 1.9.2, this case is supported by require_relative.

(B) 31.1% (69) are with File.dirname, but not with require.

For example, reading data files of unit tests.

(C) 21.6% (48) are the idiom, $0 == **FILE**.

B is as many as A (and even more than C), so it is reasonable to

add a shortcut for File.dirname(**FILE**) in addition to require_relative.

- code: http://gist.github.com/170336
- result: http://route477.net/files/__file__.html

= Solutions

(1) add a toplevel constant __DIR__
pros: looks like __FILE__
cons: adding new keyword

(2) add Kernel#__DIR__
pros: no new keyword
cons: it should have a lower-case name (like 1.9's __method__),
because it is not a constant but a method.

(3) add Kernel#__dir__
pros: no new keyword, and it is clearer than (4) that it has
some associations with __FILE__.

(4) make __FILE__ to the default argument of File.dirname

pros: no new keyword nor new method
cons: it is not clear that 'File.dirname' is expanded to
the path of directory of __FILE__.

= Conclusion

I think (3) (Kernel#__dir__) is the best.

Thanks,
--
yhara (Yutaka HARA)
http://route477.net/

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

=end

#4 - 08/30/2009 03:51 AM - matz (Yukihiro Matsumoto)

=begin
Hi,

In message "Re: [ruby-core:25182] Re: [Feature #1961] Kernel#**dir**"
on Sun, 30 Aug 2009 02:57:44 +0900, Magnus Holm judofyr@gmail.com writes:

|Wouldn't it be a little confusing to remember that **FILE** is uppercase and
|**dir** is downcase? Doesn't sound very POLS to me...

FILE is a keyword and **dir** is a method. I'd like to emphasize
this difference but I'm open to hear opinion from others.

```
matz.
```

=end

#5 - 08/30/2009 04:31 AM - zenspider (Ryan Davis)

=begin

On Aug 29, 2009, at 12:10 , Hongli Lai wrote:

I think it should be **DIR** for consistency. There are no other __
things in Ruby
that are lowercase. I think whether **dir** is a keyword or a
function should be an
implementation detail.

```
nil.methods.grep(/__/)
=> ["send", "id"]
```

=end

#6 - 08/30/2009 05:56 AM - matz (Yukihiro Matsumoto)

=begin
Hi,

In message "Re: [ruby-core:25184] Re: [Feature #1961] Kernel#**dir**"
on Sun, 30 Aug 2009 04:10:55 +0900, Hongli Lai hongli@plan99.net writes:

|I think it should be **DIR** for consistency. There are no other __ things in Ruby
|that are lowercase. I think whether **dir** is a keyword or a function should be an
|implementation detail.

On 1.9, we have **method** and **callee** lowercase for the same reason.

```
matz.
```

=end

#7 - 08/30/2009 04:14 PM - naruse (Yui NARUSE)

=begin
Hongli Lai wrote:

> I think it should be **DIR** for consistency. There are no other __ things in Ruby
> that are lowercase. I think whether **dir** is a keyword or a function should be an
> implementation detail.

You should see solutions in the original post:

> (1) add a toplevel constant **DIR**
> pros: looks like **FILE**
> cons: adding new keyword
>
> (2) add Kernel#**DIR**
> pros: no new keyword
> cons: it should have a lower-case name (like 1.9's **method**),
> because it is not a constant but a method.
>
> (3) add Kernel#**dir**
> pros: no new keyword, and it is clearer than (4) that it has
> some associations with **FILE**.
>
> (4) make **FILE** to the default argument of File.dirname
> pros: no new keyword nor new method
> cons: it is not clear that 'File.dirname' is expanded to
> the path of directory of **FILE**.

And you can add cons to (3) add Kernel#**dir**
cons: lacking consistency for **FILE**

I think so too and I like (1).

--
NARUSE, Yui naruse@airemix.jp

=end

#8 - 08/30/2009 05:12 PM - zenspider (Ryan Davis)

=begin

On Aug 30, 2009, at 00:14 , NARUSE, Yui wrote:

> You should see solutions in the original post:
>
> > (1) add a toplevel constant **DIR**
> > pros: looks like **FILE**
> > cons: adding new keyword
> >
> > (2) add Kernel#**DIR**
> > pros: no new keyword
> > cons: it should have a lower-case name (like 1.9's **method**),
> > because it is not a constant but a method.
> >
> > (3) add Kernel#**dir**
> > pros: no new keyword, and it is clearer than (4) that it has
> > some associations with **FILE**.
> >
> > (4) make **FILE** to the default argument of File.dirname
> > pros: no new keyword nor new method
> > cons: it is not clear that 'File.dirname' is expanded to
> > the path of directory of **FILE**.
>
> And you can add cons to (3) add Kernel#**dir**
> cons: lacking consistency for **FILE**
>
> I think so too and I like (1).

If we have to have this (I'm against it in general) then I think #1
makes more sense wrt consistency. I don't see any real reason why it
shouldn't be a parse-time constant like **FILE** is and I think that
is what most people would expect.

=end

#9 - 09/28/2009 06:54 PM - axgle (xiong ai)

=begin
(1) add a toplevel constant **DIR**
+1
when you type "**DIR**",you type "shift **DIR**".
when you type "**dir**",you type "shift __[release shift] dir shift __"
so,I prefer **DIR** than **dir**
thanks
=end

#10 - 09/30/2009 04:19 AM - manveru (Michael Fellinger)

=begin
On Mon, Sep 28, 2009 at 11:54 AM, xiong ai redmine@ruby-lang.org wrote:

> Issue #1961 has been updated by xiong ai.
>
> ## (1) add a toplevel constant DIR
> ## +1
> ## when you type "DIR",you type "shift DIR".
> ## when you type "dir",you type "shift __[release shift] dir shift __"
> ## so,I prefer DIR than dir
> ## thanks
>
> http://redmine.ruby-lang.org/issues/show/1961
>
> ---
>
> http://redmine.ruby-lang.org

I would be for **DIR** for the ease of writing and because it
resembles **FILE**, neither would be constants if we take into account
that leading underscore is not a valid character for constants.
But whatever name will be chosen, it would be nice if **dir** could
take arguments that are passed to File::expand_path.
A possible implementation is:
http://github.com/manveru/ramaze/blob/master/lib/ramaze/snippets/object/__dir__.rb

--
Michael Fellinger

=end

#11 - 11/29/2009 10:04 AM - ujihisa (Tatsuhiro Ujihisa)

- Status changed from Open to Assigned

- Assignee set to matz (Yukihiro Matsumoto)

=begin

=end

#12 - 03/24/2010 03:10 AM - rogerdpack (Roger Pack)

=begin
My personal preference would be **DIR**

Here's a example patch that seems to work, if it's helpful at all.

Index: prelude.rb

--- prelude.rb (revision 27020)
+++ prelude.rb (working copy)
@@ -22,3 +22,12 @@
}
end
end
+
+# Kernel
+
+module Kernel

- def **DIR**
- filename = caller[0][/^(.*):/, 1]
- File.expand_path(File.dirname(filename))
- end
  +end

(gleaned from an old Ramaze copy).

I'd be happy to add tests or change the name or what not, as well, if so directed.
Thanks.
-rp
=end

#13 - 03/24/2010 03:34 AM - murphy (Kornelius Kalnbach)

=begin
On 23.03.10 19:10, Roger Pack wrote:

> My personal preference would be **DIR**
> ...which suggests that the method is a magic constant like **FILE**.
> While funny, I don't think it's a good idea.

For example, BasicObject wouldn't have access to it.

[murphy]

=end

#14 - 03/24/2010 07:03 AM - Eregon (Benoit Daloze)

=begin

> (1) add a toplevel constant **DIR**
> pros: looks like **FILE**
> cons: adding new keyword
>
> (2) add Kernel#**DIR**
> pros: no new keyword
> cons: it should have a lower-case name (like 1.9's **method**),
> because it is not a constant but a method.
>
> (3) add Kernel#**dir**
> pros: no new keyword, and it is clearer than (4) that it has
> some associations with **FILE**.
> cons: lacking consistency for **FILE**
>
> (4) make **FILE** to the default argument of File.dirname
> pros: no new keyword nor new method
> cons: it is not clear that 'File.dirname' is expanded to
> the path of directory of **FILE**.

+1 for (1) and (3)
I first felt that **DIR** is more "natural" to me (because the script directory is something constant).
After a bit thinking, (3) looks fine too (to remember we are calling a function).
I think (2) is a bad mix and (4) does not really improve File.dirname(**FILE**)

B.D.
=end

#15 - 03/24/2010 06:29 PM - headius (Charles Nutter)

=begin

This is nice but would not be backward compatible with code that depends on **FILE** actually being a string (any code that splits on / or builds a path with + "../" for example. Maybe a subclass of String? Is there a down side to that?

class FileString < String
def dir
File.dirname(self)
end
end

- Charlie (mobile)

On Mar 23, 2010, at 5:53 PM, Evan Phoenix evan@fallingsnow.net wrote:

> Perhaps **FILE** should not be a raw String object, but rather a CurrentFile object. It could have a #to_str to return the normal String so it can be used like normal, but can provide methods like #dir, where **FILE**.dir == File.dirname(**FILE**).
>
> This solves the need for **DIR** in a very clean, object oriented way.
>
> - Evan Phoenix

=end

#16 - 03/25/2010 04:15 AM - Eregon (Benoit Daloze)

=begin

> Perhaps **FILE** should not be a raw String object, but rather a CurrentFile object. It could have a #to_str to return the normal String so it can be used like normal, but can provide methods like #dir, where **FILE**.dir == File.dirname(**FILE**).
>
> This solves the need for **DIR** in a very clean, object oriented way.

Sorry to not be according, but classes for one object and '**FILE**.dir' fear me.
The idea behind is everything but not bad anyway. Paths simply stored in String doesn't seem very appropriate.

Something like Pathname, or maybe just a Path class would be a better OO approach I think.
But I'm reluctant to the syntax of '**FILE**.meth'

B.D.

P.S.:
Is not Ruby the language where everything is an Object ?
I mean if we can do -1.abs, Path("/dir/file.ext").{basedir,dirname,dir} looks like a solution

> Perhaps __FILE__ should not be a raw String object, but rather a CurrentFile object. It could have a #to_str to return the normal String so it can be used like normal, but can provide methods like #dir, where __FILE__.dir == File.dirname(__FILE__).
>
> This solves the need for __DIR__ in a very clean, object oriented way.

Sorry to not be according, but classes for one object and '__FILE__.dir' fear me.
The idea behind is everything but not bad anyway. Paths simply stored in String doesn't seem very appropriate.

Something like Pathname, or maybe just a Path class would be a better OO approach I think.
But I'm reluctant to the syntax of '__FILE__.meth'

B.D.

P.S.:
Is not Ruby the language where everything is an Object ?
I mean if we can do -1.abs, Path("/dir/file.ext").{basedir,dirname,dir} looks like a solution

=end

#17 - 03/25/2010 12:29 PM - naruse (Yui NARUSE)

=begin
2010/3/24 Charles Oliver Nutter headius@headius.com:

> This is nice but would not be backward compatible with code that depends on **FILE** actually being a string (any code that splits on / or builds a path with + "../" for example. Maybe a subclass of String? Is there a down side to that?
>
> class FileString < String
> def dir
> File.dirname(self)
> end
> end

Why don't you use Pathname?

note:
Including pathname lib to core is sometimes discussed in core members.
But it is not concrete yet.

--
NARUSE, Yui
naruse@airemix.jp

=end

#18 - 03/25/2010 03:02 PM - headius (Charles Nutter)

=begin
Actually, I proposed that it be a String subclass, so it actually would still be a String, but with some additional logic that people tend to rewrite themselves anyway.

- Charlie (mobile)

On Mar 24, 2010, at 1:35 PM, Caleb Clausen vikkous@gmail.com wrote:

> On 3/24/10, Charles Oliver Nutter headius@headius.com wrote:
>
> > This is nice but would not be backward compatible with code that depends on **FILE** actually being a string (any code that splits on /
> > or builds a path with + "../" for example. Maybe a subclass of String?
> > Is there a down side to that?
> >
> > class FileString < String
> > def dir
> > File.dirname(self)
> > end
> > end
>
> - Charlie (mobile)
>
> On Mar 23, 2010, at 5:53 PM, Evan Phoenix evan@fallingsnow.net wrote:
>
> > Perhaps **FILE** should not be a raw String object, but rather a CurrentFile object. It could have a #to_str to return the normal String so it can be used like normal, but can provide methods like #dir, where **FILE**.dir == File.dirname(**FILE**).
> >
> > This solves the need for **DIR** in a very clean, object oriented way.
>
> - Evan Phoenix

Please, please, no. Right now, the fact that we have **ENCODING** in

> 1.9 which resolves into a type of object that doesn't even exist in the 1.8 interpreter means that I'm already having trouble parsing 1.9's **ENCODING** correctly from within the 1.8 interpreter. I would much rather have these magic keywords resolve to simple values with universally available types: String, Integer, Array. It would have been so much easier for me if **ENCODING** were just a String.
>
> My preference would be to go ahead and add **DIR** as a new keyword with a regular string value. It would be some effort for my lexer and parser to support this, but not a lot. Having simple, regular, straightforward semantics for **DIR** is to my mind much preferable to making **FILE** be some kind of almost-a-String-but-not-quite.

=end

#19 - 03/27/2010 04:41 PM - headius (Charles Nutter)

=begin
On Fri, Mar 26, 2010 at 8:27 PM, Caleb Clausen vikkous@gmail.com wrote:

> Yes, you are right. I tend to blur the distinction between subclassing and extending objects at runtime in my own thinking. But whichever it is, I'd rather not see a new type in ruby's semantics just to support this feature. A new keyword seems so much more straightforward.

A new keyword is less backward-compatible than adding methods to a string instance (or a string subclass). 1.8 could simply make **FILE**'s string additionally def dir; File.dirname(self); end and it would be compatible

Of course, this is mostly academic for me...I'm just bikeshedding. **DIR** would be fine with me too, or a per-file hash so we don't have to add future keywords (similar to __SOURCE_LINES)

- Charlie

=end

#20 - 04/02/2010 08:28 AM - znz (Kazuhiro NISHIYAMA)

- Category set to core

- Target version set to 2.0.0

=begin

=end

#21 - 05/13/2010 12:32 AM - rogerdpack (Roger Pack)

=begin
Any feedback on my patch for **dir** (or Narihiro Nakamura's patch for **dir**)?
Still wishing for this.
Thanks.
-rp
=end

#22 - 08/03/2012 07:33 PM - yhara (Yutaka HARA)

This ticket should be closed since discussion is continued to #3346 ("**DIR** revisted").

#23 - 08/03/2012 08:05 PM - matz (Yukihiro Matsumoto)

- Description updated

- Status changed from Assigned to Closed