Ruby master - Feature #12010

Exclude dot and dotdot from Dir#each

01/20/2016 02:17 AM - naruse (Yui NARUSE)

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

Description

Dir#each and Dir#read (including Dir.entries, Dir.foreach and other methods) return "." and ".." at first.
But through the all real use case "." and ".." are useless.
How about excluding them?

```
diff --git a/dir.c b/dir.c
index 193b5be..4c23a2d 100644
--- a/dir.c
+++ b/dir.c
@@ -699,6 +699,8 @@ fundamental_encoding_p(rb_encoding *enc)
 #else
 # define READDIR(dir, enc) readdir((dir))
 #endif
+#define DIR_IS_DOT_OR_DOTDOT(dp) ((dp)->d_name[0] == '.' && \
+    ((dp)->d_name[1] == '\0' || ((dp)->d_name[1] == '.' && (dp)->d_name[2] == '\0')))

 /*
  *  call-seq:
@@ -720,13 +722,12 @@ dir_read(VALUE dir)

     GetDIR(dir, dirp);
     errno = 0;
-    if ((dp = READDIR(dirp->dir, dirp->enc)) != NULL) {
-    return rb_external_str_new_with_enc(dp->d_name, NAMLEN(dp), dirp->enc);
-    }
-    else {
-    if (errno != 0) rb_sys_fail(0);
-    return Qnil;        /* end of stream */
+    while ((dp = READDIR(dirp->dir, dirp->enc)) != NULL) {
+    if (!DIR_IS_DOT_OR_DOTDOT(dp))
+        return rb_external_str_new_with_enc(dp->d_name, NAMLEN(dp), dirp->enc);
     }
+    if (errno != 0) rb_sys_fail(0);
+    return Qnil;        /* end of stream */
 }

 /*
@@ -764,6 +765,7 @@ dir_each(VALUE dir)
     const char *name = dp->d_name;
     size_t namlen = NAMLEN(dp);
     VALUE path;
+    if (DIR_IS_DOT_OR_DOTDOT(dp)) continue;
 #if NORMALIZE_UTF8PATH
     if (norm_p && has_nonascii(name, namlen) &&
         !NIL_P(path = rb_str_normalize_ospath(name, namlen))) {
diff --git a/test/pathname/test_pathname.rb b/test/pathname/test_pathname.rb
index 2690a3f..33f0d44 100644
--- a/test/pathname/test_pathname.rb
+++ b/test/pathname/test_pathname.rb
@@ -1238,7 +1238,7 @@ def test_entries
     with_tmpchdir('rubytest-pathname') {|dir|
       open("a", "w") {}
       open("b", "w") {}
-      assert_equal([Pathname("."), Pathname(".."), Pathname("a"), Pathname("b")], Pathname(".").e
ntries.sort)
```

```
+       assert_equal([Pathname("a"), Pathname("b")], Pathname(".").entries.sort)
      }
    end

@@ -1248,7 +1248,7 @@ def test_each_entry
        open("b", "w") {}
        a = []
        Pathname(".").each_entry {|v| a << v }
-       assert_equal([Pathname("."), Pathname(".."), Pathname("a"), Pathname("b")], a.sort)
+       assert_equal([Pathname("a"), Pathname("b")], a.sort)
      }
    end

@@ -1278,7 +1278,7 @@ def test_opendir
        Pathname(".").opendir {|d|
          d.each {|e| a << e }
        }
-       assert_equal([".", "..", "a", "b"], a.sort)
+       assert_equal(["a", "b"], a.sort)
      }
    end

diff --git a/test/ruby/test_dir.rb b/test/ruby/test_dir.rb
index 0cc5a6a..d3f6602 100644
--- a/test/ruby/test_dir.rb
+++ b/test/ruby/test_dir.rb
@@ -186,7 +186,7 @@ def test_glob_recursive

    def assert_entries(entries)
      entries.sort!
-     assert_equal(%w(. ..) + ("a".."z").to_a, entries)
+     assert_equal(("a".."z").to_a, entries)
    end

    def test_entries
```

Related issues:

Related to Ruby master - Feature #10121: Dir.empty?	**Closed**
Related to Ruby master - Feature #13969: Dir#each_child	**Closed**

History

#1 - 01/20/2016 02:25 AM - naruse (Yui NARUSE)

- Description updated

#2 - 01/20/2016 02:45 AM - normalperson (Eric Wong)

naruse@airemix.jp wrote:

> Dir#each and Dir#read) (including Dir.entries, Dir.foreach and other methods) return "." and ".." at first.
> But through the all real use case "." and ".." are useless.
> How about excluding them?

If Ruby were a new language, yes. But I think it is too risky, now.

```
+#define DIR_IS_DOT_OR_DOTDOT(dp) ((dp)->d_name[0] == '.' && \
+ ((dp)->d_name[1] == '\0' || ((dp)->d_name[1] == '.' && (dp)->d_name[2] == '\0')))
```

Anyways, I prefer we reduce macro usage and use static inline functions
to avoid potential side effects, extra '\' and parentheses.

#3 - 01/20/2016 03:11 AM - nobu (Nobuyoshi Nakada)

- Description updated

d_name might not be NUL-terminated, use NAMLEN() on such platforms.

```
diff --git a/dir.c b/dir.c
index 193b5be..28a1e79 100644
--- a/dir.c
+++ b/dir.c
@@ -21,6 +21,7 @@
 #include <unistd.h>
 #endif

+#undef HAVE_DIRENT_NAMLEN
 #if defined HAVE_DIRENT_H && !defined _WIN32
 # include <dirent.h>
 # define NAMLEN(dirent) strlen((dirent)->d_name)
@@ -30,6 +31,7 @@
 #else
 # define dirent direct
 # define NAMLEN(dirent) (dirent)->d_namlen
+# define HAVE_DIRENT_NAMLEN 1
 # if HAVE_SYS_NDIR_H
 #  include <sys/ndir.h>
 # endif
@@ -699,6 +701,26 @@ fundamental_encoding_p(rb_encoding *enc)
 #else
 # define READDIR(dir, enc) readdir((dir))
 #endif
+static int
+to_be_skipped(const struct dirent *dp)
+{
+    const char *name = dp->d_name;
+    if (name[0] != '.') return FALSE;
+#ifdef HAVE_DIRENT_NAMLEN
+    switch (NAMLEN(dp)) {
+      case 2:
+   if (name[1] != '.') return FALSE;
+      case 1:
+   return TRUE;
+      default:
+    }
+#else
+    if (!name[1]) return TRUE;
+    if (name[1] != '.') return FALSE;
+    if (!name[2]) return TRUE;
+#endif
+    return FALSE;
+}

 /*
  *  call-seq:
@@ -720,13 +742,12 @@ dir_read(VALUE dir)

     GetDIR(dir, dirp);
     errno = 0;
-    if ((dp = READDIR(dirp->dir, dirp->enc)) != NULL) {
-   return rb_external_str_new_with_enc(dp->d_name, NAMLEN(dp), dirp->enc);
-    }
-    else {
-   if (errno != 0) rb_sys_fail(0);
-   return Qnil;        /* end of stream */
+    while ((dp = READDIR(dirp->dir, dirp->enc)) != NULL) {
+   if (!to_be_skipped(dp))
+       return rb_external_str_new_with_enc(dp->d_name, NAMLEN(dp), dirp->enc);
     }
+    if (errno != 0) rb_sys_fail(0);
+    return Qnil;       /* end of stream */
 }

 /*
@@ -762,8 +783,10 @@ dir_each(VALUE dir)
     IF_NORMALIZE_UTF8PATH(norm_p = need_normalization(dirp->dir, RSTRING_PTR(dirp->path)));
     while ((dp = READDIR(dirp->dir, dirp->enc)) != NULL) {
    const char *name = dp->d_name;
-   size_t namlen = NAMLEN(dp);
+   size_t namlen;
    VALUE path;
+   if (to_be_skipped(dp)) continue;
+   namlen = NAMLEN(dp);
```

```
#if NORMALIZE_UTF8PATH
    if (norm_p && has_nonascii(name, namlen) &&
        !NIL_P(path = rb_str_normalize_ospath(name, namlen))) {
```

#4 - 01/20/2016 10:26 AM - naruse (Yui NARUSE)

Nobuyoshi Nakada wrote:

> d_name might not be NUL-terminated, use NAMLEN() on such platforms.

What is the platform?
POSIX says it is NUL-terminated.
http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/dirent.h.html

#5 - 01/20/2016 12:33 PM - Eregon (Benoit Daloze)

Yui NARUSE wrote:

> Dir#each and Dir#read (including Dir.entries, Dir.foreach and other methods) return "." and ".." at first.
> But through the all real use case "." and ".." are useless.
> How about excluding them?

Strongly agreed, I had this on my "things to report" list forever but never got around to report it.

I am unsure of the potential compatibility issues.
I guess code ignoring "." and ".." will behave just fine though.

P.S.: "." and ".." are not necessarily at first it seems, at least on Linux.

#6 - 01/21/2016 02:10 AM - nobu (Nobuyoshi Nakada)

Yui NARUSE wrote:

> What is the platform?

Win32 was in my mind, but opendir_internal() terminates it.
So it "should respect d_namlen if available".

#7 - 01/21/2016 02:28 AM - naruse (Yui NARUSE)

Nobuyoshi Nakada wrote:

> Yui NARUSE wrote:
>
> > What is the platform?
>
> Win32 was in my mind, but opendir_internal() terminates it.
> So it "should respect d_namlen if available".

It sounds "don't need to respect".

#8 - 01/21/2016 03:46 AM - nobu (Nobuyoshi Nakada)

- Related to Feature #10121: Dir.empty? added

#9 - 01/21/2016 03:50 AM - duerst (Martin Dürst)

Hello Eric,

On 2016/01/20 11:43, Eric Wong wrote:

> naruse@airemix.jp wrote:
>
> > Dir#each and Dir#read) (including Dir.entries, Dir.foreach and other methods) return "." and ".." at first.
> > But through the all real use case "." and ".." are useless.
> > How about excluding them?
>
> If Ruby were a new language, yes. But I think it is too risky, now.

Can somebody do a code search for this? I know AKR is good at that (but
I don't want to ask him to do this).

> > +#define DIR_IS_DOT_OR_DOTDOT(dp) ((dp)->d_name[0] == '.' && \
>
> - ((dp)->d_name[1] == '\0' || ((dp)->d_name[1] == '.' && (dp)->d_name[2] == '\0')))
>
> > Anyways, I prefer we reduce macro usage and use static inline functions
> > to avoid potential side effects, extra '\' and parentheses.

Are inline functions now an accepted concept in C? It seems they are
available from C99, but then so are // comments, and they are not yet
accepted in Ruby because of some old compilers.

Also, are there C compilers that inline functions as part of
optimization, even if they aren't marked as such in the source?

While I very much understand your point re. potential side effects,
using inline functions also may save space at the cost of time, in
particular for long macros. Is that also one of the reasons you favor them?

Regards, Martin.

Unsubscribe: ruby-core-request@ruby-lang.org?subject=unsubscribe
http://lists.ruby-lang.org/cgi-bin/mailman/options/ruby-core

#10 - 01/21/2016 04:25 AM - akr (Akira Tanaka)

- File dir-entries-usages.txt added

I searched Dir.entries on gems.

It seems there are size/length invocations on the result of Dir.entries.

Note that I used https://github.com/akr/gem-codesearch

#11 - 01/21/2016 05:02 AM - normalperson (Eric Wong)

"Martin J. Dürst" duerst@it.aoyama.ac.jp wrote:

> On 2016/01/20 11:43, Eric Wong wrote:
>
> > naruse@airemix.jp wrote:
> >
> > > Dir#each and Dir#read) (including Dir.entries, Dir.foreach and other methods) return "." and ".." at first.
> > > But through the all real use case "." and ".." are useless.
> > > How about excluding them?
> >
> > If Ruby were a new language, yes. But I think it is too risky, now.
>
> Can somebody do a code search for this? I know AKR is good at that
> (but I don't want to ask him to do this).

I just found some in yahns which I wrote:
http://yhbt.net/yahns.git/plain/extras/autoindex.rb

Easily fixable, but we also don't know what kinds of code people
have in private.

> > Anyways, I prefer we reduce macro usage and use static inline functions
> > to avoid potential side effects, extra '\' and parentheses.
>
> Are inline functions now an accepted concept in C? It seems they are
> available from C99, but then so are // comments, and they are not
> yet accepted in Ruby because of some old compilers.

It seems so, we already have static inlines everywhere in Ruby.

I guess AC_C_INLINE takes care of the portability inside configure.in

'//' comments can't be worked around using CPP on old compilers.

Fwiw, git and Linux kernel both reject '//' despite using
static inlines heavily; Linux also uses C99 struct initializers,
but git does not as git targets more compilers than Linux.

> Also, are there C compilers that inline functions as part of
> optimization, even if they aren't marked as such in the source?

Yes, actually I often favor plain 'static' to let the compiler
more make decisions. In my experience with gcc; single-use
'static' functions always get inlined (but I don't look at giant
functions much). Usually, smaller icache footprints are
better for overall performance(*).

gcc will also complain about unused 'static', but not 'static inline'
in header files.

> While I very much understand your point re. potential side effects,
> using inline functions also may save space at the cost of time, in
> particular for long macros. Is that also one of the reasons you
> favor them?

Yes, space, too. I prefer to trust the compiler as much as possible
in the hopes they can make better decisions than me (or may eventually
do so, perhaps with things like PGO).

Functions also get type-checked at compile time. Compile-time
type-checking is nice in C :)

(*) Of course, we default to -O3 in which makes our icache footprint
huge. I've tried with -O2 in the past and unfortunately our
benchmarks got slower. It seems there's some hotspots in the
VM core loop that benefit from -O3...

Unsubscribe: ruby-core-request@ruby-lang.org?subject=unsubscribe
http://lists.ruby-lang.org/cgi-bin/mailman/options/ruby-core

#12 - 01/21/2016 07:47 AM - naruse (Yui NARUSE)

Akira Tanaka wrote:

> I searched Dir.entries on gems.
>
> It seems there are size/length invocations on the result of Dir.entries.
>
> Note that I used https://github.com/akr/gem-codesearch

Thanks, I hadn't thought up such use case.
It sounds need a treatment at least.

Could you check about Dir.foreach?

#13 - 01/21/2016 07:58 AM - naruse (Yui NARUSE)

Eric Wong wrote:

> "Martin J. Dürst" duerst@it.aoyama.ac.jp wrote:
>
> > On 2016/01/20 11:43, Eric Wong wrote:
> >
> > naruse@airemix.jp wrote:
> >
> > > Dir#each and Dir#read) (including Dir.entries, Dir.foreach and other methods) return "." and ".." at first.
> > > But through the all real use case "." and ".." are useless.
> > > How about excluding them?
> >
> > If Ruby were a new language, yes. But I think it is too risky, now.

> Can somebody do a code search for this? I know AKR is good at that
> (but I don't want to ask him to do this).

I just found some in yahns which I wrote:
http://yhbt.net/yahns.git/plain/extras/autoindex.rb

Hmm...
Emulating directory listing sounds a valid use case...

#14 - 01/22/2016 03:38 AM - kddeisz (Kevin Deisz)

Seems like a good place for an option in these methods, but default it to include to make migration easier.

#15 - 01/23/2016 08:42 AM - shevegen (Robert A. Heiler)

> default it to include to make migration easier

I dunno. I never found a usecase for '.' and '..' so I would have no migration need at all since I did not use it. And in the old cases where I used it, I always ended up filtering away the '.' and '..' since I have no idea what to do with it.

But I also do not use Dir.entries much at all either since many years. I fell in love with Dir[] and have been using that ever since, respectively Dir['*'] or any other filter. I assume that perhaps Dir.entries() is not used that much in "modern" ruby code.

I did a grep in the rack-1.6.4 source, just to use something as reference :-)

4 Instances of Dir[], one in lib/rack/directory.rb, two in rack.gemspec,
one in a Rakefile.

And no instance of Dir.entries hehe

#16 - 03/01/2018 01:13 AM - nobu (Nobuyoshi Nakada)

- Related to Feature #13969: Dir#each_child added

#17 - 03/01/2018 01:13 AM - nobu (Nobuyoshi Nakada)

This can be closed by #13969?

Files

dir-entries-usages.txt	304 KB	01/21/2016	akr (Akira Tanaka)