

Ruby - Feature #1081

add File::write() convenience method

02/01/2009 05:46 AM - sunaku (Suraj Kurapati)

Status:	Closed	
Priority:	Normal	
Assignee:	mame (Yusuke Endoh)	
Target version:	1.9.3	
Description		
<pre>=begin</pre>		
Please add a File::write() convenience method to the core Ruby API.		
Currently, it is easier to read whole files than to write them:		
reading a whole file --- less effort		
<pre>text = File::read('foo.txt')</pre>		
writing a whole file --- more effort		
<pre>File::open('foo.txt', 'wb') { f f.write 'ruby!' }</pre>		
This imbalance can be corrected by adding a File::write method, such as the following, to the core Ruby API:		
<pre>class File def self.write path, data, mode = 'wb' open(path, mode) { f f.write data } end end</pre>		
Thanks for your consideration.		
<pre>=end</pre>		
Related issues:		
Related to Ruby - Bug #4846: Permission denied - /tmp/nonexisting		Closed
		06/07/2011

Associated revisions

Revision 7b3948f0 - 06/02/2011 07:30 AM - sorah (Sorah Fukumori)

- io.c: Add File.write, File.binwrite. [Feature #1081] [ruby-core:21701]
- test/ruby/test_io.rb: Test for File.write, File.binwrite.
- NEWS: News for above.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@31902 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 02/02/2009 12:02 PM - ko1 (Koichi Sasada)

- Assignee set to matz (Yukihiro Matsumoto)

=begin

=end

#2 - 02/02/2009 03:28 PM - matz (Yukihiro Matsumoto)

```
=begin
Hi,
```

In message "Re: [\[ruby-core:21701\]](#) [Feature [#1081](#)] add File::write() convenience method"
on Sun, 1 Feb 2009 05:46:57 +0900, Suraj Kurapati redmine@ruby-lang.org writes:

|Please add a File::write() convenience method to the core Ruby API.

|Currently, it is easier to read whole files than to write them:

```
| # reading a whole file --- less effort
| text = File::read('foo.txt')
|
| # writing a whole file --- more effort
| File::open('foo.txt', 'wb') {|f| f.write 'ruby!' }
```

open() has following API

open(path, mode="r", mode=0, ophash=nil)

write() has

IO#write(string)

What do you think proper API for combined File.open?

File::write(path, string, mode, opt)

or

File::write(string, path, mode, opt)

or whatever? The latter would conflict with some in-house definition
of File::write.

```
matz.
```

```
=end
```

#3 - 02/02/2009 04:10 PM - murphy (Kornelius Kalnbach)

```
=begin
```

Yukihiro Matsumoto wrote:

What do you think proper API for combined File.open?

File::write(path, string, mode, opt)

or

File::write(string, path, mode, opt)
we badly need keyword arguments.

[murphy]

```
=end
```

#4 - 02/03/2009 01:08 PM - sunaku (Suraj Kurapati)

```
=begin
```

Hi,

Sorry for my late response.

Matz wrote:

What do you think proper API for combined File.open?

File::write(path, string, mode, opt)

I prefer this order of parameters because it feels natural:

File::open('hello.txt', 'r').read()

```
File::read('hello.txt')
```

```
File::open('hello.txt', 'w').write('hello')  
File::write('hello.txt', 'hello')
```

In addition, the Facets library also prefers this order:

<http://facets.rubyforge.org/doc/api/core/classes/File.html#M000241>

or

```
File::write(string, path, mode, opt)
```

The latter would conflict with some in-house definition of File::write.

Then, to minimize disruption, let us avoid this latter order of parameters.

Thanks for your consideration.
=end

#5 - 04/02/2009 01:05 AM - rogerdpack (Roger Pack)

=begin

What do you think proper API for combined File.open?

```
File::write(path, string, mode, opt)
```

I also prefer this first way.
File.write('hello.txt', 'some stuff')

and default to mode 'wb' for all platforms
Thanks for looking into this. It would be quite useful.

=end

#6 - 09/22/2009 03:24 AM - sunaku (Suraj Kurapati)

=begin

To summarize Roger's comment, we prefer this API:

```
File::write(path, string, mode='wb', opt={})
```

Thanks for your consideration.
=end

#7 - 09/28/2009 06:40 PM - axgle (xiong ai)

=begin

```
File::write(path, string, mode='wb', opt={})
```

+1

=end

#8 - 11/24/2009 05:25 AM - rogerdpack (Roger Pack)

=begin

Hmm.

On second thought perhaps on windows it should be

```
File::write(path, string, mode='w', opt={})
```

And there should be

```
File::binwrite(path, string, mode='wb', opt={})
```

for consistency with 1.9's existing File::read and File::binread

Thoughts?
-r

=end

#9 - 01/03/2010 04:56 AM - sunaku (Suraj Kurapati)

```
=begin
That seems reasonable Roger. I agree:
```

```
File::write(path, string, mode='w', opt={})
File::binwrite(path, string, mode='wb', opt={})
=end
```

#10 - 02/28/2010 06:17 PM - mame (Yusuke Endoh)

```
=begin
Hi,
```

Please add a File::write() convenience method to the core Ruby API.

I have written a patch. The API is similar to File.read except string:

```
File.write(path, string[, offset])      #=> length written
File.write(path, string[, offset], open_args) #=> length written
```

If offset is given, the file is not truncated. Otherwise, truncated.

```
$ ./ruby -e 'File.write("foo", "foo\nbar\nbaz\n")' && cat foo
foo
bar
baz
```

```
$ ./ruby -e 'File.write("foo", "BAR", 4)' && cat foo
foo
BAR
baz
```

```
$ ./ruby -e 'File.write("foo", "FOO\n")' && cat foo
FOO
```

```
$ ./ruby -e 'File.write("foo", "ひひひひ", mode: "w", encoding: "EUC-JP")' && nkf -Ew foo
ひひひひ
```

I'll commit the patch if anyone says objection.

```
diff --git a/io.c b/io.c
index 8026ea3..08d4988 100644
--- a/io.c
+++ b/io.c
@@ -7701,7 +7701,7 @@ struct foreach_arg {
};
```

```
static void
+open_key_args(int argc, VALUE *argv, struct foreach_arg *arg)
+open_key_args_with_opt(int argc, VALUE *argv, struct foreach_arg *arg, int mandatory_argc, int default_mode, int default_perm)
{
VALUE opt, v;
```

```
@@ -7709,9 +7709,9 @@ open_key_args(int argc, VALUE *argv, struct foreach_arg *arg)
arg->io = 0;
arg->argc = argc - 1;
arg->argv = argv + 1;
```

- if (argc == 1) {
- if (argc == mandatory_argc) {
no_key:
- arg->io = rb_io_open(argv[0], INT2NUM(O_RDONLY), INT2FIX(0666), Qnil);
- arg->io = rb_io_open(argv[0], INT2NUM(default_mode), INT2FIX(default_perm), Qnil);
return;
}
opt = pop_last_hash(&arg->argv);
@@ -7736,9 +7736,23 @@ open_key_args(int argc, VALUE *argv, struct foreach_arg *arg)
rb_ary_clear(args); /prevent from GC */
return;
}
• if (default_mode != O_RDONLY && NIL_P(rb_hash_aref(opt, sym_mode))) {
• opt = rb_hash_dup(opt);

```

    • rb_hash_aset(opt, sym_mode, INT2NUM(default_mode));
    • }
    • if (default_perm != 0666 && NIL_P(rb_hash_aref(opt, sym_perm))) {
    • opt = rb_hash_dup(opt);
    • rb_hash_aset(opt, sym_perm, INT2FIX(default_perm));
    • }
    arg->io = rb_io_open(argv[0], Qnil, Qnil, opt);
    }

+static void
+open_key_args(int argc, VALUE *argv, struct foreach_arg *arg)
+{

    • open_key_args_with_opt(argc, argv, arg, 1, O_RDONLY, 0666);
    +}

static VALUE
io_s_foreach(struct foreach_arg *arg)
{
@@ -7826,6 +7840,16 @@ io_s_read(struct foreach_arg *arg)
return io_read(arg->argc, arg->argv, arg->io);
}

+struct write_arg {

    • VALUE io, str;
    +};

+static VALUE
+io_s_write(struct write_arg *arg)
+{

    • return io_write(arg->io, arg->str, 0);
    +}

struct seek_arg {
VALUE io;
VALUE offset;
@@ -7833,7 +7857,7 @@ struct seek_arg {
};

static VALUE
-seek_before_read(struct seek_arg *arg)
+seek_before_access(struct seek_arg *arg)
{
rb_io_binmode(arg->io);
return rb_io_seek(arg->io, arg->offset, arg->mode);
@@ -7844,7 +7868,7 @@ seek_before_read(struct seek_arg *arg)

    • [REDACTED]

    • [REDACTED]

    ◦ Opens the file, optionally seeks to the given offset, then returns

    ◦ Opens the file, optionally seeks to the given offset, then returns
    ◦ length bytes (defaulting to the rest of the file).
    ◦ read ensures the file is closed before returning.

@@ -7886,7 +7910,7 @@ rb_io_s_read(int argc, VALUE *argv, VALUE io)
sarg.io = arg.io;
sarg.offset = offset;
sarg.mode = SEEK_SET;

    • rb_protect((VALUE (*)(VALUE))seek_before_read, (VALUE)&sarg, &state);

    • rb_protect((VALUE (*)(VALUE))seek_before_access, (VALUE)&sarg, &state);
    if (state) {
    rb_io_close(arg.io);
    rb_jump_tag(state);

```

@@ -7900,9 +7924,9 @@ rb_io_s_read(int argc, VALUE *argv, VALUE io)

- call-seq:

- 

- Opens the file, optionally seeks to the given offset, then returns
- Opens the file, optionally seeks to the given *offset*, then returns
- *length* bytes (defaulting to the rest of the file).
- read ensures the file is closed before returning.
- binread ensures the file is closed before returning.
- The open mode would be "rb:ASCII-8BIT".

```
IO.binread("testfile")      #=> "This is line one\nThis is line two\nThis is line three\nAnd so on\n...\n"
```

```
@@ -7928,6 +7952,117 @@ rb_io_s_binread(int argc, VALUE *argv, VALUE io)
return rb_ensure(io_s_read, (VALUE)&arg, rb_io_close, arg.io);
}
```

+/

- call-seq:

```
IO.write(name, string, [offset] )    => fixnum
IO.write(name, string, [offset], open_args )  => fixnum
```

- Opens the file, optionally seeks to the given *offset*, writes
- *string*, then returns the length written.
- write ensures the file is closed before returning.
- If *offset* is not given, the file is truncated. Otherwise,
- it is not truncated.
- If the last argument is a hash, it specifies option for internal
- open(). The key would be the following. open_args: is exclusive
- to others.
- encoding: string or encoding
- specifies encoding of the read string. encoding will be ignored
- if length is specified.
- mode: string
- specifies mode argument for open(). it should start with "w" or "a" or "r+"
- otherwise it would cause error.

- perm: fixnum
- specifies perm argument for open().
- open_args: array of strings
- specifies arguments for open() as an array.

```

◦ IO.write("testfile", "0123456789")      #=> "0123456789"

◦ IO.write("testfile", "0123456789", 20)  #=> "This is line one\nThi0123456789two\nThis is line three\nAn
  d so on...\n"

• */

```

```

+static VALUE
+rb_io_s_write(int argc, VALUE *argv, VALUE io)
+{
  • VALUE offset;
  • struct foreach_arg arg;
  • struct write_arg warg;
  • int mode = O_WRONLY | O_CREAT, mandatory_argc;

  • rb_scan_args(argc, argv, "22", NULL, &warg.str, &offset, NULL);
  • if (!NIL_P(offset) && FIXNUM_P(offset)) {
  •   mandatory_argc = 3;
  • }
  • else {
  •   mode |= O_TRUNC;
  •   mandatory_argc = 2;
  • }
  • open_key_args_with_opt(argc, argv, &arg, mandatory_argc, mode, 0666);
  • if (NIL_P(arg.io)) return Qnil;
  • if (!NIL_P(offset) && FIXNUM_P(offset)) {
  •   struct seek_arg sarg;
  •   int state = 0;
  •   sarg.io = arg.io;
  •   sarg.offset = offset;
  •   sarg.mode = SEEK_SET;
  •   rb_protect((VALUE (*)(VALUE))seek_before_access, (VALUE)&sarg, &state);
  •   if (state) {

  •   [REDACTED]

  •   [REDACTED]

  • }
  • if (arg.argc == 2) arg.argc = 1;
  • }
  • warg.io = arg.io;
  • return rb_ensure(io_s_write, (VALUE)&warg, rb_io_close, arg.io);
+}

```

```

+/*
  • call-seq:

```

```
◦ IO.binwrite(name, string, [offset] ) => fixnum
```

◦ Opens the file, optionally seeks to the given *offset*, write

◦ *string* then returns the length written.

◦ binwrite ensures the file is closed before returning.

◦ The open mode would be "wb:ASCII-8BIT".

◦ If *offset* is not given, the file is truncated. Otherwise,

◦ it is not truncated.

```
◦ IO.binwrite("testfile", "0123456789") #=> "0123456789"
```

```
◦ IO.binwrite("testfile", "0123456789", 20) #=> "This is line one\nThi0123456789two\nThis is line three\nAnd so on...\n"
```

• */

+static VALUE

+rb_io_s_binwrite(int argc, VALUE *argv, VALUE io)

+{

- VALUE offset;
 - const char *mode;
 - struct write_arg warg;
 - rb_scan_args(argc, argv, "21", NULL, &warg.str, &offset);
 - if (!NIL_P(offset)) {
 - NUM2OFFT(offset);
 - mode = "ab:ASCII-8BIT";
 - }
 - else {
 - mode = "wb:ASCII-8BIT";
 - }
 - FilePathValue(argv[0]);
 - warg.io = rb_io_open(argv[0], rb_str_new_cstr(mode), Qnil, Qnil);
 - if (NIL_P(warg.io)) return Qnil;
 - if (!NIL_P(offset)) {
 - rb_io_seek(warg.io, offset, SEEK_SET);
 - }
 - return rb_ensure(io_s_write, (VALUE)&warg, rb_io_close, warg.io);
- +}

struct copy_stream_struct {

VALUE src;

VALUE dst;

@@ -9731,6 +9866,8 @@ Init_IO(void)

rb_define_singleton_method(rb_cIO, "readlines", rb_io_s_readlines, -1);

rb_define_singleton_method(rb_cIO, "read", rb_io_s_read, -1);

rb_define_singleton_method(rb_cIO, "binread", rb_io_s_binread, -1);

- rb_define_singleton_method(rb_cIO, "write", rb_io_s_write, -1);
 - rb_define_singleton_method(rb_cIO, "binwrite", rb_io_s_binwrite, -1);
 - rb_define_singleton_method(rb_cIO, "select", rb_f_select, -1);
 - rb_define_singleton_method(rb_cIO, "pipe", rb_io_s_pipe, -1);
 - rb_define_singleton_method(rb_cIO, "try_convert", rb_io_s_try_convert, 1);
- ```
diff --git a/test/ruby/test_io.rb b/test/ruby/test_io.rb
index 5fc3b13..6ea128a 100644
--- a/test/ruby/test_io.rb
+++ b/test/ruby/test_io.rb
@@ -1512,4 +1512,34 @@ End
t.close
assert_raise(IOError) {t.binmode}
end
```

- def test\_s\_write

- t = Tempfile.new("foo")
- path = t.path
- t.close(false)
- File.write(path, "foo\nbar\nbaz")
- assert("foo\nbar\nbaz", File.read(path))
- File.write(path, "FOO", 0)
- assert("FOO\nbar\nbaz", File.read(path))
- File.write(path, "BAR")
- assert("BAR", File.read(path))
- File.write(path, "\u{3042}", mode: "w", encoding: "EUC-JP")
- assert("\u{3042}".encode("EUC-JP"), File.read(path, encoding: "EUC-JP"))
- t.unlink
- end

- def test\_s\_binwrite
- t = Tempfile.new("foo")
- path = t.path
- t.close(false)
- File.binwrite(path, "foo\nbar\nbaz")
- assert("foo\nbar\nbaz", File.read(path))
- File.binwrite(path, "FOO", 0)
- assert("FOO\nbar\nbaz", File.read(path))
- File.binwrite(path, "BAR")
- assert("BAR", File.read(path))
- File.binwrite(path, "\u{3042}")
- assert("\u{3042}", File.read(path, encoding: "EUC-JP"))
- t.unlink
- end
- end

--

Yusuke Endoh [mame@tsg.ne.jp](mailto:mame@tsg.ne.jp)

=end

#### #11 - 03/04/2010 02:48 AM - mame (Yusuke Endoh)

=begin

Hi,

Yusuke Endoh wrote:

I have written a patch. The API is similar to File.read except string:

```
File.write(path, string[, offset]) #=> length written
File.write(path, string[, offset], open_args) #=> length written
```

*snip*

I'll commit the patch if anyone says objection.

Sorry, I have misunderstood matz had already approved the feature in [\[ruby-core:21732\]](#). But I realized he just discussed API design.

I have not commit it yet. Matz, may I commit my patch?

--

Yusuke Endoh [mame@tsg.ne.jp](mailto:mame@tsg.ne.jp)

=end

#### #12 - 03/04/2010 10:25 AM - matz (Yukihiro Matsumoto)

=begin

Hi,

In message "Re: [\[ruby-core:28450\]](#) [Feature [#1081](#)] add File::write() convenience method" on Thu, 4 Mar 2010 02:48:14 +0900, Yusuke Endoh [redmine@ruby-lang.org](mailto:redmine@ruby-lang.org) writes:

|I have not commit it yet. Matz, may I commit my patch?

Go ahead.

matz.

=end

**#13 - 03/04/2010 10:50 PM - mame (Yusuke Endoh)**

=begin  
Hi,

2010/3/4 Yukihiro Matsumoto [matz@ruby-lang.org](mailto:matz@ruby-lang.org):

In message "Re: [\[ruby-core:28450\]](#) [Feature [#1081](#)] add File::write() convenience method"  
on Thu, 4 Mar 2010 02:48:14 +0900, Yusuke Endoh [redmine@ruby-lang.org](mailto:redmine@ruby-lang.org) writes:

|I have not commit it yet. Matz, may I commit my patch?

Go ahead.

Thank you! Done.

--  
Yusuke ENDOH [mame@tsg.ne.jp](mailto:mame@tsg.ne.jp)

=end

**#14 - 03/04/2010 10:57 PM - mame (Yusuke Endoh)**

- Status changed from Open to Closed

- % Done changed from 0 to 100

=begin  
This issue was solved with changeset r26816.  
Suraj, thank you for reporting this issue.  
Your contribution to Ruby is greatly appreciated.  
May Ruby be with you.

=end

**#15 - 03/05/2010 11:24 PM - runpaint (Run Paint Run Run)**

=begin  
Is it intended that the length of the leading nulls are not included  
in the return value? I assumed the second line in the example below  
would return 3.

```
File.delete('/tmp/glark') #=> 1
File.write('/tmp/glark','s',2) #=> 1
File.read('/tmp/glark') #=> "\u0000\u0000s"
```

=end

**#16 - 03/06/2010 12:03 AM - mame (Yusuke Endoh)**

=begin  
Hi,

2010/3/5 Run Paint Run Run [runrun@runpaint.org](mailto:runrun@runpaint.org):

Is it intended that the length of the leading nulls are not included  
in the return value? I assumed the second line in the example below  
would return 3.

```
File.delete('/tmp/glark') #=> 1
File.write('/tmp/glark','s',2) #=> 1
File.read('/tmp/glark') #=> "\u0000\u0000s"
```

Intended. It currently returns the length actually written. The  
\u0000s are padded by seek, not write.

Consider the situation where there is /tmp/glark whose size is more

than 2:

```
File.read('/tmp/glark') #=> "foo"
File.write('/tmp/glark', "s", 2) #=> 1
File.read('/tmp/glark') #=> "fos"
```

In this case, it is natural (for me) to return 1.

But I'm happy to change the spec if you guys want.

--

Yusuke ENDOH [mame@tsq.ne.jp](mailto:mame@tsq.ne.jp)

=end

#### #17 - 03/06/2010 05:32 AM - runpaint (Run Paint Run Run)

=begin

Is it intended that the length of the leading nulls are not included in the return value? I assumed the second line in the example below would return 3.

```
>> File.delete('/tmp/glark') #=> 1
>> File.write('/tmp/glark','s',2) #=> 1
>> File.read('/tmp/glark') #=> "\u0000\u0000s"
```

Intended. It currently returns the length actually written. The \u0000s are padded by seek, not write.

OK, just wanted to check. As for #binwrite, I'm also assuming that, at least on Linux, it's intentionally ignoring the offset, treating it instead as merely a cue to append.

```
File.delete('/tmp/offset') #=> 1
File.write('/tmp/offset','ruby') #=> 4
File.write('/tmp/offset','ruby',2) #=> 4
File.size('/tmp/offset') #=> 6
```

```
File.delete('/tmp/offset') #=> 1
File.binwrite('/tmp/offset','ruby') #=> 4
File.binwrite('/tmp/offset','ruby',2) #=> 4
File.size('/tmp/offset') #=> 8
```

=end

#### #18 - 03/06/2010 10:15 AM - mame (Yusuke Endoh)

=begin

Hi,

2010/3/6 Run Paint Run Run [runrun@runpaint.org](mailto:runrun@runpaint.org):

As for #binwrite, I'm also assuming that, at least on Linux, it's intentionally ignoring the offset, treating it instead as merely a cue to append.

```
File.delete('/tmp/offset') #=> 1
File.write('/tmp/offset','ruby') #=> 4
File.write('/tmp/offset','ruby',2) #=> 4
File.size('/tmp/offset') #=> 6
```

```
File.delete('/tmp/offset') #=> 1
File.binwrite('/tmp/offset','ruby') #=> 4
File.binwrite('/tmp/offset','ruby',2) #=> 4
```

```
File.size('/tmp/offset') #=> 8
```

No, it is never intended :-(  
Fixed. Thanks.

## I had used assert instead of assert\_equal in the test...

--  
Yusuke ENDOH [mame@tsg.ne.jp](mailto:mame@tsg.ne.jp)

=end

**#19 - 03/06/2010 12:01 PM - runpaint (Run Paint Run Run)**

=begin  
Teamwork. :-)

The only other inconsistency I found is how offsets are handled for non-existent paths.

```
>> File.delete('/tmp/offset') #=> 1
>> File.binwrite('/tmp/offset', 'string', 2)
Errno::ENOENT: No such file or directory - /tmp/offset
from (irb):6:in `binwrite'
from (irb):6
from /usr/local/bin/irb:12:in `<main>'
>> File.write('/tmp/offset', 'string', 2) #=> 6
>> File.read('/tmp/offset') #=> "\u0000\u0000string"
```

```
>> File.delete('/tmp/offset') #=> 1
>> File.binwrite('/tmp/offset', 'string', -2)
Errno::ENOENT: No such file or directory - /tmp/offset
from (irb):14:in `binwrite'
from (irb):14
from /usr/local/bin/irb:12:in `<main>'
>> File.write('/tmp/offset', 'string', -2)
Errno::EINVAL: Invalid argument - /tmp/offset
from (irb):16:in `write'
from (irb):16
from /usr/local/bin/irb:12:in `<main>'
```

=end

**#20 - 03/06/2010 12:49 PM - mame (Yusuke Endoh)**

=begin  
2010/3/6 Run Paint Run Run [runrun@runpaint.org](mailto:runrun@runpaint.org):

The only other inconsistency I found is how offsets are handled for non-existent paths.

Grr, I hate IO.

In addition, I noticed we cannot specify permission with the current File#binwrite.

I'll consider for some time.

--  
Yusuke ENDOH [mame@tsg.ne.jp](mailto:mame@tsg.ne.jp)

=end

**#21 - 03/06/2010 02:03 PM - mame (Yusuke Endoh)**

=begin  
Hi,

2010/3/6 Yusuke ENDOH [mame@tsg.ne.jp](mailto:mame@tsg.ne.jp):

2010/3/6 Run Paint Run Run [runrun@runpaint.org](mailto:runrun@runpaint.org):

The only other inconsistency I found is how offsets are handled for non-existent paths.

Grr, I hate IO.

In addition, I noticed we cannot specify permission with the current File#binwrite.

I'll consider for some time.

I give up. I have already reverted and deleted the two methods.

I wrote the patch because I thought closing the ticket encourages 1.9.2 release.

But it seems to take some time to make the feature stable, and I became afraid that it may delay the release. It is never the effect I expected.

I think refactoring io.c is needed to implement the feature.  
open argument is easy to use, but difficult for me to implement :-(

--

Yusuke ENDOH [mame@tsg.ne.jp](mailto:mame@tsg.ne.jp)

=end

**#22 - 03/06/2010 02:06 PM - mame (Yusuke Endoh)**

- Status changed from Closed to Open

- Target version changed from 1.9.2 to 2.0.0

=begin

=end

**#23 - 03/06/2010 11:58 PM - runpaint (Run Paint Run Run)**

=begin

I give up. I have already reverted and deleted the two methods.

I wrote the patch because I thought closing the ticket encourages 1.9.2 release.

But it seems to take some time to make the feature stable, and I became afraid that it may delay the release. It is never the effect I expected.

I think refactoring io.c is needed to implement the feature.  
open argument is easy to use, but difficult for me to implement :-(

Why don't we just start simple? Add IO.write and IO.binwrite that take exactly two arguments--a path and a string--and truncate the file and write to it the string. That covers the common use case which inspired this ticket, and allows the optional arguments to be added later without breaking compatibility.

=end

**#24 - 03/07/2010 02:32 AM - runpaint (Run Paint Run Run)**

- File 0001-io.c-io\_s\_write-io\_s\_binwrite-Re-add-IO.write-binwri.patch added

=begin

For example, with the usual caveat that I don't speak C, the attached patch passes the following RubySpec: <http://goo.gl/RcAW> (the two methods are treated as aliases because I haven't looked at testing binmode under Linux). I suspect that if it was rewritten by somebody who knew what they were doing it would satisfy the original reporter and simplify the common idiom of "File.open('/tmp/glark','w'){|f| f << 'string'}". If more functionality is desired, IO.open can be used or another ticket can be opened.

=end

**#25 - 03/07/2010 03:36 AM - nobu (Nobuyoshi Nakada)**

```
=begin
Hi,
```

At Sun, 7 Mar 2010 02:32:56 +0900,  
Run Paint Run Run wrote in [\[ruby-core:28536\]](#):

File 0001-io.c-io\_s\_write-io\_s\_binwrite-Re-add-IO.write-binwri.patch added

It's broken, and will cause segfault or similar.

```
+static VALUE
+rb_io_s_write(int argc, VALUE path, VALUE str, VALUE io)

+static VALUE
+rb_io_s_binwrite(int argc, VALUE path, VALUE str, VALUE io)

 • rb_define_singleton_method(rb_cIO, "write", rb_io_s_write, 2);
 • rb_define_singleton_method(rb_cIO, "binwrite", rb_io_s_binwrite, 2);
```

```
--
Nobu Nakada
```

```
=end
```

#### #26 - 03/09/2010 12:23 AM - mame (Yusuke Endoh)

```
=begin
Hi,
```

2010/3/6 Run Paint Run Run [runrun@runpaint.org](mailto:runrun@runpaint.org):

I give up. I have already reverted and deleted the two methods.

I wrote the patch because I thought closing the ticket encourages 1.9.2 release.

But it seems to take some time to make the feature stable, and I became afraid that it may delay the release. ?It is never the effect I expected.

I think refactoring io.c is needed to implement the feature.  
open argument is easy to use, but difficult for me to implement :-(

Why don't we just start simple?

Because there are already complex File.read/binread.  
It is confusing to make File.write/binwrite inconsistent with them.

That covers the common use case which inspired  
this ticket, and allows the optional arguments to be added later  
without breaking compatibility.

There is no reason to rush into 1.9.2.

```
--
Yusuke ENDOH mame@tsg.ne.jp
```

```
=end
```

#### #27 - 03/15/2010 10:41 AM - sunaku (Suraj Kurapati)

```
=begin
Hi,
```

Is this issue solved? Or was the patch<sup>1</sup> removed from Ruby?

Sorry for asking these (perhaps self-evident) questions;  
I really don't understand the discussion succeeding <sup>1</sup>.

Thanks for your consideration.

```
=end
```

**#28 - 03/16/2010 12:52 AM - mame (Yusuke Endoh)**

=begin  
Hi,

2010/3/15 Suraj Kurapati [redmine@ruby-lang.org](mailto:redmine@ruby-lang.org):

Is this issue solved? ?Or was the patch[1] removed from Ruby?

Removed. [\[ruby-core:28530\]](#)

I thought it is difficult (for me) to make the feature stable by the deadline.  
I will not disagree with a patch that anyone writes, if it passes Run Paint's corner cases. [\[ruby-core:28517\]](#) [\[ruby-core:28528\]](#)

--  
Yusuke ENDOH [mame@tsg.ne.jp](mailto:mame@tsg.ne.jp)

=end

**#29 - 04/29/2010 03:42 AM - rogerdpack (Roger Pack)**

- *File new.diff added*

=begin  
I was unable to reproduce the odd behavior observed previously by Run Paint Run.

Here is the original diff plus some test cases that cover the corner cases (they don't appear to fail--maybe something else has been fixed since then, allowing things to work right now?)  
Maybe somebody can point out a failing test case to me?

Thanks.

./ruby test/ruby/test\_io.rb  
Loaded suite test/ruby/test\_io  
Started

.....  
Finished in 0.687023 seconds.

91 tests, 358 assertions, 0 failures, 0 errors, 0 skips

Test run options: --seed 26858  
=end

**#30 - 04/29/2010 12:22 PM - nobu (Nobuyoshi Nakada)**

=begin  
Hi,

At Thu, 29 Apr 2010 03:42:57 +0900,  
Roger Pack wrote in [\[ruby-core:29861\]](#):

Here is the original diff plus some test cases that cover the corner cases (they don't appear to fail--maybe something else has been fixed since then, allowing things to work right now?)  
Maybe somebody can point out a failing test case to me?

It's sorry that your tests have no meanings at all, except for `assert_raise`. You should use `assert_equal` instead of mere `assert`.

--  
Nobu Nakada

=end

**#31 - 04/30/2010 02:32 AM - rogerdpack (Roger Pack)**

- *File latest.diff added*

=begin  
Sorry for the poor tests.  
Fixing the tests revealed that there were some bugs in binwrite.

The attached (new) patch fixes the tests and code. The tests pass on windows and linux (and I think are accurate).  
Feel free to refactor it as desired.

(as a note, windows currently fails some other tests in test\_io.rb <http://gist.github.com/383867>).

Thanks.  
-rp  
=end

**#32 - 09/14/2010 04:39 PM - shyouhei (Shyouhei Urabe)**

- Status changed from Open to Assigned

=begin  
  
=end

**#33 - 11/01/2010 11:37 PM - rogerdpack (Roger Pack)**

=begin  
Any chance of getting this committed at all? (File#write)?  
Thanks!  
=end

**#34 - 11/02/2010 07:25 AM - Eregon (Benoit Daloze)**

=begin  
On 1 November 2010 15:37, Roger Pack [redmine@ruby-lang.org](mailto:redmine@ruby-lang.org) wrote:

Issue [#1081](#) has been updated by Roger Pack.

Any chance of getting this committed at all? (File#write)?  
Thanks!

I want to show my wish to back up this, it is definitely worth it.

Is there any issue left ?

=end

**#35 - 03/19/2011 04:41 AM - rogerdpack (Roger Pack)**

- File *add\_file\_write\_prelude.diff* added  
- % Done changed from 100 to 0

=begin  
Ok here is a new patch (the old one no longer merged cleanly, plus had minor bugs). Feel free to reformat it or convert to C.

The format is:

File::write(path, string, offset, opt={}) or  
File::write(path, string, opt={})

Which I think is reasonable, and matches the previous commits (the ones that were rolled out because of bugs). Apparently offset is common in other  
api's so I guess is good to have in there.  
Thanks for consideration of this.

-r  
=end

**#36 - 05/11/2011 08:13 PM - kosaki (Motohiro KOSAKI)**

- Assignee changed from matz (Yukihiro Matsumoto) to kosaki (Motohiro KOSAKI)

**#37 - 05/13/2011 03:37 AM - kosaki (Motohiro KOSAKI)**

- Assignee changed from kosaki (Motohiro KOSAKI) to mame (Yusuke Endoh)

**#38 - 05/18/2011 03:37 AM - rogerdpack (Roger Pack)**

- File latest\_also\_accomodate\_lack\_of\_mode\_param.diff added

Uploading new tests and a fix that accomodate for the previously failing example.

**#39 - 05/28/2011 02:50 PM - mame (Yusuke Endoh)**

- Target version changed from 2.0.0 to 1.9.3

Hello,

The status of this feature request is considered as "accepted".

If anyone writes a patch (in C) for this feature by the implementation deadline (the end of June) and if there is no objection to the patch, it will be imported in 1.9.3.

If not, this feature will be postponed to 1.9.4 or later.

Sora seems to be interested in this feature. Good luck.

--

Yusuke Endoh [mame@tsg.ne.jp](mailto:mame@tsg.ne.jp)

**#40 - 05/29/2011 11:32 PM - sorah (Sorah Fukumori)**

Hi, I'm implementing this feature in C.

I've been completed File.write, I'll complete implementation of File.binwrite in C.

--sora\_h

**#41 - 05/30/2011 10:57 AM - sorah (Sorah Fukumori)**

- File sorah\_implementation.diff added

Hi,

My Implementation has been completed File.write, binwrite.

Patch is attached.

RDocs in patch and test is copied from previous patches.

**#42 - 05/30/2011 10:53 PM - mame (Yusuke Endoh)**

Hello,

2011/5/30 Shota Fukumori [sorah@tubusu.net](mailto:sorah@tubusu.net):

My Implementation has been completed File.write, binwrite.

Good work.

But File.binwrite does not accept an optional hash while write does.

```
$ rm -f /tmp/x && ./ruby -e 'File.write("/tmp/x", "foo", perm: 0700)'
"foo"
```

```
$ rm -f /tmp/x && ./ruby -e 'File.binwrite("/tmp/x", "foo", perm: 0700)'
-e:1:in binwrite: can't convert Hash into Integer (TypeError) from -e:1:in '
```

--

Yusuke Endoh [mame@tsg.ne.jp](mailto:mame@tsg.ne.jp)

**#43 - 05/31/2011 03:43 PM - sorah (Sorah Fukumori)**

Patch has updated, Now File.binwrite accepts Hash for specifying options:

<https://gist.github.com/69c544ec245f3a07aabd>

**#44 - 05/31/2011 10:18 PM - mame (Yusuke Endoh)**

Hello,

2011/5/31 Shota Fukumori [sorah@tubusu.net](mailto:sorah@tubusu.net):

Patch has updated, Now File.binwrite accepts Hash for specifying options:

Great! I tested your patch and noticed no problem.  
Roger and rubyspec folks, could you also check it?

In terms of maintainability, I think that it would be better to  
define a common function for `rb_io_s_write` and `rb_io_s_binwrite`:

<http://www.atdot.net/sp/view/pw92ml>

```
diff --git a/io.c b/io.c
index 4e1945c..25e0974 100644
--- a/io.c
+++ b/io.c
@@ -805,6 +805,12 @@ struct binwrite_arg {http://www.atdot.net/sp/view/pw92ml
 long length;
 };

+struct write_arg {
+
+ • VALUE io;
+ • VALUE str;
+ • int nosync;
+ +};

static VALUE
io_binwrite_string(VALUE arg)
{
@@ -836,6 +837,124 @@ rb_io_s_binread(int argc, VALUE *argv, VALUE io)
return rb_ensure(io_s_read, (VALUE)&arg, rb_io_close, arg.io);
}

+static VALUE
+io_s_write0(struct write_arg *arg)
+{
+
+ • return io_write(arg->io,arg->str,arg->nosync);
+ +}

+static VALUE
+io_s_write(int argc, VALUE *argv, int binary)
+{
+
+ • VALUE string, offset, opt;
+ • struct foreach_arg arg;
+ • struct write_arg warg;

+ • rb_scan_args(argc, argv, "21:", NULL, &string, &offset, &opt);

+ • if (NIL_P(opt)) opt = rb_hash_new();
+ • else opt = rb_hash_dup(opt);

+ • if (NIL_P(rb_hash_aref(opt,sym_mode))) {
+ • int mode = O_WRONLY|O_CREAT;
+ • #ifdef O_BINARY
+ • if (binary) mode |= O_BINARY;
+ • #endif
+ • if (NIL_P(offset)) mode |= O_TRUNC;
+ • rb_hash_aset(opt,sym_mode,INT2NUM(mode));
+ • }
+ • open_key_args(argc,argv,opt,&arg);

+ #ifndef O_BINARY

+ • if (binary) rb_io_binmode_m(arg.io);
+ • #endif

+ • if (NIL_P(arg.io)) return Qnil;
+ • if (!NIL_P(offset)) {
+ • struct seek_arg sarg;
```

- `int state = 0;`
- `sarg.io = arg.io;`
- `sarg.offset = offset;`
- `sarg.mode = SEEK_SET;`
- `rb_protect(seek_before_access, (VALUE)&sarg, &state);`
- `if (state) {`
- `[REDACTED]`
- `[REDACTED]`
- `}`
- `}`
- `warg.io = arg.io;`
- `warg.str = string;`
- `warg.nosync = 0;`
- `return rb_ensure(io_s_write0, (VALUE)&warg, rb_io_close, arg.io);`
- `+}`

`+/*`

- call-seq:
- `IO.write(name, string, [offset] ) => fixnum`
- `IO.write(name, string, [offset], open_args ) => fixnum`
- Opens the file, optionally seeks to the given *offset*, writes
- *string*, then returns the length written.
- `write` ensures the file is closed before returning.
- If *offset* is not given, the file is truncated. Otherwise,
- it is not truncated.
- If the last argument is a hash, it specifies option for internal
- `open()`. The key would be the following. `open_args`: is exclusive
- to others.
- encoding: string or encoding
- specifies encoding of the read string. encoding will be ignored
- if length is specified.
- mode: string
- specifies mode argument for `open()`. it should start with "w" or "a" or "r+"
- otherwise it would cause error.
- perm: fixnum

- specifies perm argument for open().

- open\_args: array of strings

- specifies arguments for open() as an array.

- `IO.write("testfile", "0123456789")` `#=> "0123456789"`

- `IO.write("testfile", "0123456789", 20)` `#=> "This is line one\nThi0123456789two\nThis is line three\nAnd so on...\n"`

- `*/`

```
+static VALUE
```

```
+rb_io_s_write(int argc, VALUE *argv, VALUE io)
```

```
+{
```

- `io_s_write(argc, argv, 0);`  
`+`

```
+/*
```

- call-seq:

- `IO.binwrite(name, string, [offset] )` `=> fixnum`

- Opens the file, optionally seeks to the given *offset*, write

- *string* then returns the length written.

- `binwrite` ensures the file is closed before returning.

- The open mode would be "wb:ASCII-8BIT".

- If *offset* is not given, the file is truncated. Otherwise,

- it is not truncated.

- `IO.binwrite("testfile", "0123456789")` `#=> "0123456789"`

- `IO.binwrite("testfile", "0123456789", 20)` `#=> "This is line one\nThi0123456789two\nThis is line three\nAnd so on...\n"`

- `*/`

```
+static VALUE
```

```
+rb_io_s_binwrite(int argc, VALUE *argv, VALUE io)
```

```
+{
```

- `io_s_write(argc, argv, 1);`  
`+`

```
struct copy_stream_struct {
```

```
VALUE src;
```

```
VALUE dst;
```

```

@@ -10315,6 +10439,8 @@ Init_IO(void)
rb_define_singleton_method(rb_cIO, "readlines", rb_io_s_readlines, -1);
rb_define_singleton_method(rb_cIO, "read", rb_io_s_read, -1);
rb_define_singleton_method(rb_cIO, "binread", rb_io_s_binread, -1);

 • rb_define_singleton_method(rb_cIO, "write", rb_io_s_write, -1);

 • rb_define_singleton_method(rb_cIO, "binwrite", rb_io_s_binwrite, -1);
 rb_define_singleton_method(rb_cIO, "select", rb_f_select, -1);
 rb_define_singleton_method(rb_cIO, "pipe", rb_io_s_pipe, -1);
 rb_define_singleton_method(rb_cIO, "try_convert", rb_io_s_try_convert, 1);
diff --git a/test/ruby/test_io.rb b/test/ruby/test_io.rb
index f919227..8aaecd1 100644
--- a/test/ruby/test_io.rb
+++ b/test/ruby/test_io.rb
@@ -1861,4 +1861,61 @@ End
end
end

 • def test_s_write

 • t = Tempfile.new("foo")

 • path = t.path

 • t.close(false)

 • File.write(path, "foo\nbar\nbaz")

 • assert_equal("foo\nbar\nbaz", File.read(path))

 • File.write(path, "FOO", 0)

 • assert_equal("FOO\nbar\nbaz", File.read(path))

 • File.write(path, "BAR")

 • assert_equal("BAR", File.read(path))

 • File.write(path, "\u{3042}", mode: "w", encoding: "EUC-JP")

 • assert_equal("\u{3042}.encode("EUC-JP"), File.read(path, encoding: "EUC-JP"))

 • File.delete t

 • assert_equal(6, File.write(path, 'string', 2))

 • File.delete t

 • assert_raise(Errno::EINVAL) { File.write('/tmp/nonexisting', 'string', -2) }

 • assert_equal(6, File.write(path, 'string'))

 • assert_equal(3, File.write(path, 'sub', 1))

 • assert_equal("ssubng", File.read(path))

 • File.delete t

 • assert_equal(3, File.write(path, "foo", encoding: "UTF-8"))

 • File.delete t

 • assert_equal(3, File.write(path, "foo", 0, encoding: "UTF-8"))

 • assert_equal("foo", File.read(path))

 • assert_equal(1, File.write(path, "f", 1, encoding: "UTF-8"))

 • assert_equal("ffo", File.read(path))

 • File.delete t

 • assert_equal(1, File.write(path, "f", 1, encoding: "UTF-8"))

```

- `assert_equal("\00f", File.read(path))`
- `assert_equal(1, File.write(path, "f", 0, encoding: "UTF-8"))`
- `assert_equal("ff", File.read(path))`
- `t.unlink`
- `end`
- `def test_s_binwrite`
- `t = Tempfile.new("foo")`
- `path = t.path`
- `t.close(false)`
- `File.binwrite(path, "foo\nbar\nbaz")`
- `assert_equal("foo\nbar\nbaz", File.read(path))`
- `File.binwrite(path, "FOO", 0)`
- `assert_equal("FOO\nbar\nbaz", File.read(path))`
- `File.binwrite(path, "BAR")`
- `assert_equal("BAR", File.read(path))`
- `File.binwrite(path, "\u{3042}")`
- `assert_equal("\u{3042}".force_encoding("ASCII-8BIT"), File.binread(path))`
- `File.delete t`
- `assert_equal(6, File.binwrite(path, 'string', 2))`
- `File.delete t`
- `assert_equal(6, File.binwrite(path, 'string'))`
- `assert_equal(3, File.binwrite(path, 'sub', 1))`
- `assert_equal("ssubng", File.binread(path))`
- `assert_equal(6, File.size(path))`
- `assert_raise(Errno::EINVAL) { File.binwrite('/tmp/nonexisting', 'string', -2) }`
- `assert_nothing_raised(TypeError) { File.binwrite(path, "string", mode: "w", encoding: "EUC-JP") }`
- `t.unlink`
- `end`
- `end`

--

Yusuke Endoh [mame@tsg.ne.jp](mailto:mame@tsg.ne.jp)

**#45 - 06/01/2011 07:23 AM - sorah (Sorah Fukumori)**

hi,

On Tue, May 31, 2011 at 10:20 PM, Yusuke Endoh [mame@tsg.ne.jp](mailto:mame@tsg.ne.jp) wrote:

Great! Â I tested your patch and noticed no problem.  
Roger and rubyspec folks, could you also check it?

In terms of maintainability, I think that it would be better to  
define a common function for `rb_io_s_write` and `rb_io_s_binwrite`:

ok. thanks refactoring.

--

Shota Fukumori a.k.a. @sora\_h - <http://codnote.net/>

**#46 - 06/01/2011 11:20 AM - sorah (Sorah Fukumori)**

I want to commit ASAP because we're going to freeze specification,  
so if there aren't any problems, I'll commit patch at <http://redmine.ruby-lang.org/issues/1081#note-44> [ruby-core:36630].

I remembered that matz approved this new method with this specification, if not please notify me.

**#47 - 06/02/2011 04:30 PM - sorah (Sorah Fukumori)**

- Status changed from Assigned to Closed

- % Done changed from 0 to 100

This issue was solved with changeset r31902.  
Suraj, thank you for reporting this issue.  
Your contribution to Ruby is greatly appreciated.  
May Ruby be with you.

- 
- io.c: Add File.write, File.binwrite. [Feature [#1081](#)] [[ruby-core:21701](#)]
  - test/ruby/test\_io.rb: Test for File.write, File.binwrite.
  - NEWS: News for above.

**#48 - 10/16/2011 12:14 PM - hackeron (Roman Gaufrman)**

It seems the docs here are wrong <http://www.ruby-doc.org/core-1.9.3/IO.html#method-c-write> -- they mention 4 arguments when only 3 are supported and the :mode isn't explained - please update the docs.

**Files**

|                                                                 |         |            |                              |
|-----------------------------------------------------------------|---------|------------|------------------------------|
| 0001-io.c-io_s_write-io_s_binwrite-Re-add-IO.write-binwri.patch | 2.86 KB | 03/07/2010 | runpaint (Run Paint Run Run) |
| new.diff                                                        | 9.02 KB | 04/29/2010 | rogerdpack (Roger Pack)      |
| latest.diff                                                     | 7.49 KB | 04/30/2010 | rogerdpack (Roger Pack)      |
| add_file_write_prelude.diff                                     | 4.66 KB | 03/19/2011 | rogerdpack (Roger Pack)      |
| latest_also_accomodate_lack_of_mode_param.diff                  | 5.59 KB | 05/18/2011 | rogerdpack (Roger Pack)      |
| sorah_implementation.diff                                       | 7.46 KB | 05/30/2011 | sorah (Sorah Fukumori)       |