# Ruby master - Bug #14413

## `-n` and `-p` flags break when stdout is closed

01/28/2018 01:35 AM - josh.cheek (Josh Cheek)

| | | | |
|---|---|---|---|
| **Status:** | Closed | | |
| **Priority:** | Normal | | |
| **Assignee:** | | | |
| **Target version:** | | | |
| **ruby -v:** | ruby 2.5.0preview1 (2017-10-10 trunk 60153) [x86_64-darwin16] | **Backport:** | 2.3: UNKNOWN, 2.4: UNKNOWN, 2.5: UNKNOWN |

**Description**

Ruby generally works well within a pipeline. The -n and -p flags are incredibly useful. However, it is common practice to use programs like head and sed, which will close the pipe after completing their job. This is convenient, because often it limits an expensive amount of output to a workable subset. I can figure out the pipeline for a subset of input, and then remove the limiting function.

However, Ruby explodes with -e:1:inwrite': Broken pipe @ io_write - (Errno::EPIPE), when it writes the current line to stdout. When in a line oriented mode, and stdout closes, I think it should exit successfully (so it doesn't break bash scripts withpipe fail`set) and silently (no error message), hence marking this a bug report rather than a feature request.

I've attached a screenshot to show that this is how every other program works, that I tried.

```
git clone https://github.com/jquery/esprima
cd esprima/test/fixtures/
ls                                                                    | head -1 # ls
find . -type f -name '*json'                                          | head -1 # find
find . -type f -name '*json' | head -1 | xargs cat | jq .             | head -1 # jq
find . -type f -name '*json' | grep -v JSX | grep -v tokenize         | head -1 # grep
find . -type f -name '*json' | sed -E '/JSX|tokenize/d'               | head -1 # sed
find . -type f -name '*json' | awk '/JSX|tokenize/ { next }; { print }' | head -1 # awk
find . -type f -name '*json' | perl -e 'while(<>) { /JSX|tokenize/ || print }' | head -1 # perl
find . -type f -name '*json' | ruby -ne 'print unless /JSX|tokenize/'  | head -1 # ruby :(
```

### Associated revisions

**Revision 6f28ebd5 - 04/11/2020 12:30 AM - nobu (Nobuyoshi Nakada)**

Silence broken pipe error messages on STDOUT [Feature #14413]

Raise SignalException for SIGPIPE to abort when EPIPE occurs.

**Revision 155f64e3 - 04/15/2020 12:05 PM - nobu (Nobuyoshi Nakada)**

Raise EPIPE at broken pipe for the backward compatibility

Instead of SignalException for SIGPIPE, raise Errno::EPIPE with
instance variable signo and re-send that signal at exit.
[Feature #14413]

### History

**#1 - 01/28/2018 01:46 AM - josh.cheek (Josh Cheek)**

Actually, maybe it should choose this behavior when -e is passed, as well. I often use -e, standalone. Here's a video from just a week ago where I used it to filter output (though, as I watch it now, I realized I could have done the same thing with the -n flag). Looking at the output, it's easy to imagine myself passing that into further pipeline segments.

https://vimeo.com/251434577

**#2 - 01/28/2018 01:08 PM - nobu (Nobuyoshi Nakada)**

*- Status changed from Open to Feedback*

In common, SIGPIPE will terminate the process printing to closed pipe.
And few programs "exit successfully" in such case.

```
$ yes | cat | head -1; echo ${PIPESTATUS[@]}
y
141 141 0

$ yes | grep ^ | head -1; echo ${PIPESTATUS[@]}
y
141 141 0

$ yes | sed '' | head -1; echo ${PIPESTATUS[@]}
y
141 141 0
```

You can set SIGPIPE signal handler to system default to behave as above.

```
$ yes | ruby -pe 'BEGIN{trap(:PIPE,:SYSTEM_DEFAULT)}' | head -1; echo ${PIPESTATUS[@]}
y
141 141 0
```

Or require a small library to do it.

```
# sigpipe.rb
BEGIN{trap(:PIPE,:SYSTEM_DEFAULT)}

$ yes | ruby -r./sigpipe -pe '' | head -1; echo ${PIPESTATUS[@]}
y
141 141 0
```

#### #3 - 01/28/2018 02:48 PM - nobu (Nobuyoshi Nakada)

A patch to exit with SIGPIPE when EPIPE if -n or -p option is given.

```
diff --git i/error.c w/error.c
index 7cbbf101e0..2f454d01da 100644
--- i/error.c
+++ w/error.c
@@ -53,6 +53,7 @@ int rb_str_end_with_asciichar(VALUE str, int c);
 VALUE rb_eEAGAIN;
 VALUE rb_eEWOULDBLOCK;
 VALUE rb_eEINPROGRESS;
+VALUE rb_eEPIPE;
 static VALUE rb_mWarning;
 static VALUE rb_cWarningBuffer;

@@ -1816,6 +1817,9 @@ set_syserr(int n, const char *name)
       case EINPROGRESS:
         rb_eEINPROGRESS = error;
         break;
+      case EPIPE:
+        rb_eEPIPE = error;
+        break;
     }

     rb_define_const(error, "Errno", INT2NUM(n));
diff --git i/parse.y w/parse.y
index c8acac5d2c..57dbc7f7db 100644
--- i/parse.y
+++ w/parse.y
@@ -10571,11 +10571,13 @@ rb_parser_warn_location(VALUE vparser, int warn)
     p->warn_location = warn;
 }

+extern VALUE rb_eEPIPE;
 static NODE *
 parser_append_options(struct parser_params *p, NODE *node)
 {
     static const YYLTYPE default_location = {{1, 0}, {1, 0}};
     const YYLTYPE *const LOC = &default_location;
+    int ignore_epipe = p->do_print || p->do_loop;

     if (p->do_print) {
     NODE *print = NEW_FCALL(rb_intern("print"),
@@ -10584,6 +10586,19 @@ parser_append_options(struct parser_params *p, NODE *node)
     node = block_append(p, node, print);
     }
```

```
+    if (ignore_epipe) {
+   /* rescue Errno::EPIPE then raise SignalException.new(SIGPIPE) */
+   VALUE signo = INT2FIX(SIGPIPE);
+   VALUE exc = rb_class_new_instance(1, &signo, rb_eSignal);
+   NODE *res = NEW_RESBODY(NEW_LIST(NEW_LIT(rb_eEPIPE, LOC), LOC),
+               NEW_FCALL(rb_intern("raise"),
+                     NEW_LIST(NEW_LIT(exc, LOC), LOC),
+                     LOC),
+               0,
+               LOC);
+   node = NEW_RESCUE(node, res, 0, LOC);
+    }
+
     if (p->do_loop) {
    if (p->do_split) {
        NODE *split = NEW_GASGN(rb_intern("$F"),
```

#### #4 - 01/28/2018 03:57 PM - josh.cheek (Josh Cheek)

nobu (Nobuyoshi Nakada) wrote:

> A patch to exit with SIGPIPE when EPIPE if -n or -p option is given.

□□ ♥□ □□

#### #5 - 01/29/2018 01:05 AM - nobu (Nobuyoshi Nakada)

It didn't work with END {}.

```
# silent_epipe.rb
# -*- frozen-string-literal :true -*-
BEGIN {
  if Errno.const_defined?("EPIPE") and Signal.list["PIPE"]
    class SystemCallError
      prepend Module.new {
        sigpipe = SignalException.new("PIPE").freeze
        define_method(:exception) {Errno::EPIPE === self ? sigpipe : super}
      }
    end
  end
}
```

```
$ yes | ruby -w -r./silent_epipe -pe 'END{STDERR.puts :END}' | head -1; echo ${PIPESTATUS[@]}
y
END
141 141 0
```

#### #6 - 01/29/2018 02:17 AM - nobu (Nobuyoshi Nakada)

It may be better to translate Errno::EPIPE to SIGPIPE only in STDOUT.write.

```
# silent_epipe.rb
# -*- frozen-string-literal :true -*-
BEGIN {
  if Errno.const_defined?("EPIPE") and Signal.list["PIPE"]
    class << STDOUT
      prepend Module.new {
        def write(*)
          super
        rescue Errno::EPIPE
          raise SignalException.new("PIPE")
        end
      }
    end
  end
}
```

#### #7 - 12/11/2019 11:07 PM - shevegen (Robert A. Heiler)

My comment here is not primarily related to the threadstarter's comment, but indirectly. I am struggling a little bit in a cgi/www environment where debug is a bit difficult. The log files (from lighttpd) show broken pipes related to puts. I have not yet identified the cause of it, but I wanted to add that it is all a bit confusing to me as a ruby user right now. The first exposure I

had was the "broken pipe" situation; even after googling for this (and also having found the issue here), it's still a bit frustrating to deal with.

Any improvement in this regard in the future would be very useful, because I believe it is quite common for people to use ruby for different applications, including the www, GUI wrappers (such as ruby-gtk in my case), and of course my main use of ruby, in a commandline-linux situation.

#### #8 - 04/03/2020 08:05 PM - josh.cheek (Josh Cheek)

*- File Screen Shot 2020-04-03 at 2.58.41 PM.png added*

I saw flipflop was readded and was wanting to explain to people how to use it. I feel like if people knew how to use it, they wouldn't have wanted to remove it. So, I made this example (shell is fish, not sure if this example would work in bash):

```
# a csv with a header
> cat ~/Downloads/Casos1' (3)'.csv | head -2
"ID de caso","Fecha de diagnóstico","Ciudad de ubicación","Departamento o Distrito","Atención**","Edad","Sexo"
,"Tipo*","País de procedencia"
"1","06/03/2020","Bogotá","Bogotá D.C.","Recuperado","19","F","Importado","Italia"


> cat ~/Downloads/Casos1' (3)'.csv |

# `-n` iterates over each line of input
# `-e` means the argument is the program, not a filename
ruby -ne '
  # `print` will print the current line.
  # `2..` is a flipflop beginning at 2, with no upper bound.
  # It will flip to true on line 2, and never flip back to false.
  # Hence, this prints all lines except for the first.
  print if 2..
' |

head -1
"1","06/03/2020","Bogotá","Bogotá D.C.","Recuperado","19","F","Importado","Italia"
Traceback (most recent call last):
    2: from -e:6:in `<main>'
    1: from -e:6:in `print'
-e:6:in `write': Broken pipe @ io_write - <STDOUT> (Errno::EPIPE)
```

It does the correct behaviour (essentially sed 1d), but this example isn't compelling, because the error message means you would have to redirect stderr to /dev/null, which is hacky, and would also swallow real errors.

Perhaps another way to show this:

```
# desired
> seq 1000000 | sed -n '12,$p' | head -2
12
13

# actual
> seq 1000000 | ruby -ne 'print if 12..' | head -2
12
13
Traceback (most recent call last):
    2: from -e:1:in `<main>'
    1: from -e:1:in `print'
-e:1:in `write': Broken pipe @ io_write - <STDOUT> (Errno::EPIPE)
```

#### #9 - 04/03/2020 08:44 PM - josh.cheek (Josh Cheek)

For anyone curious, these are the examples I went with instead: https://twitter.com/josh_cheek/status/1246175226213404672

#### #10 - 04/10/2020 04:01 PM - ioquatix (Samuel Williams)

I think we need to be careful about changing the exceptions that IO operations can raise.

We already have:

SocketError, IOError, EOFError, Errno::EPIPE, Errno::ECONNRESET, and several others which need to be handled correctly.

Sorry, I may misunderstand this issue or PR, but it would be good to get clarity on the current PR.

#### #11 - 04/10/2020 04:45 PM - akr (Akira Tanaka)

I like silent exit at EPIPE on STDOUT.

The error message of ruby ... | head doesn't help us.
The user, who types the command line, knows the ruby process cannot continue to output to STDOUT after EPIPE.

Alto, I think it's Unix-ish.

These reasons doesn't depend on -p/-n option.
So I advocated (at a developer meeting) to change
the behavior at EPIPE on STDOUT regardless of -p/-n option.

However, it would be dangerous for other IOs, such as sockets.
So, the change should not affect to all IOs.
(STDERR would be changeable, though.)

#### #12 - 04/11/2020 12:30 AM - nobu (Nobuyoshi Nakada)

*- Status changed from Feedback to Closed*

Applied in changeset git|6f28ebd585fba1aff1c9591ced08ed11b68ba9e3.

---

Silence broken pipe error messages on STDOUT [Feature #14413]

Raise SignalException for SIGPIPE to abort when EPIPE occurs.

#### #13 - 04/11/2020 02:39 AM - ioquatix (Samuel Williams)

I understand change. It makes sense. My only concern is:

- Is exit status changed when SignalException is raised?
- It's reasonable for users to want non-blocking output to stdout. The semantics are changed for this one file descriptor, may cause confusion/bugs. User expect IO#write to fail with Errno::EPIPE in such a situation, but now it's different.

I don't have strong opinion about it, but the consistency of error handling becomes even more tricky with this change. If I have time I will try it out and report back.

Maybe an alternative solution is not to change what error is raised, but change how that error is handled when ruby is part of a shell pipeline. I don't know where Kernel#write comes from, but maybe top level puts, write and so on, if those are commonly used for pipe, maybe changing those semantics is more straight forward?

e.g. https://github.com/ruby/ruby/blob/a0c7c23c9cec0d0ffcba012279cd652d28ad5bf3/io.c#L7788 could handle EPIPE.

### Files

| | | | |
|---|---|---|---|
| Screen Shot 2018-01-27 at 7.27.49 PM.png | 458 KB | 01/28/2018 | josh.cheek (Josh Cheek) |
| Screen Shot 2020-04-03 at 2.58.41 PM.png | 158 KB | 04/03/2020 | josh.cheek (Josh Cheek) |