Ruby - Bug #7859

Readline: Incorrect arrow key behavior in vi_editing_mode insert mode with Readline 6.2

02/16/2013 04:20 AM - davidbalbert (David Albert)

Status:	Closed		
Priority:	Normal		
Assignee:	kouji (Kouji Takao)		
Target version:			
ruby -v:	2.0.0-rc2	**Backport:**	

Description

=begin
I've discovered what I think is a bug in the (({Readline})) module in the standard library. When I am using (({vi_editing_mode})) in insert mode (rather than command mode), I am unable to use the up arrow to go up through history. It seems that I can only go up through history when in command mode. Additionally, pressing the down arrow while in insert mode changes to command mode, which seems odd.

Perhaps this is intended behavior for the (({Readline})) module, but if it is, I would propose changing it. I would expect the up and down arrows to scroll up and down through history in both command mode and insert mode when (({Readline.vi_editing_mode?})) is true. You can find examples of the expected behavior in bash (((%set -o vi%)) to get into vi mode), the Python REPL, and all other that I can remember using.

I've reproduced this with (({Readline})) 6.2 on Mac OS X 10.8.2 and Ubuntu precise64 with kernel version 3.2.0-37. It is worth noting that on Mac OS X with the EditLine wrapper, the (({Readline})) module works correctly although you must have the proper settings in your .editrc file because (({Readline.vi_editing_mode})) is not implemented.

Here is the code I used to test:

readlinetest.rb

```
require 'readline'

trap(:INT) {
exit 0
}

Readline.vi_editing_mode
puts "Readline::VERSION => #{Readline::VERSION}"

loop do
puts Readline.readline(">> ", true)
end
```

Example usage:

```
$ ruby readlinetest.rb
Readline::VERSION => 6.2
```

```
        1234
        1234
```

At this point, I would expect that the up arrow would put 1234 after the prompt, but instead nothing happens. Pressing the down arrow is the same as pressing escape and changes (({readline})) into command mode.

Let me know if there's anything else I can provide to help fix this. I tried jumping into the (({Readline})) module myself, but I'm not particularly familiar with how (({readline})) works and wasn't able to make much headway.
=end

History

#1 - 02/16/2013 01:22 PM - nobu (Nobuyoshi Nakada)

- Description updated

- Category set to ext

- Status changed from Open to Assigned

- Assignee set to kouji (Kouji Takao)

- Target version set to 2.6

#2 - 04/25/2013 05:55 PM - Anonymous

I have different, but perhaps related, problems with Readline.vi_editing_mode in Ruby 1.9.3p392 with MacPorts' readline 6.2 on MacOS 10.6.8. Please let me know if I should open a new 1.9.3 bug report instead of commenting here.

Using the readlinetest.rb program in the above report, any arrow key jumps back to the first line of history and prints it's escape sequence, and ESC itself jumps back to the first line of history. (It seems the initial escape sequence, either generated by an arrow-key or ESC itself, is printing the first line.)

```
$ ruby tmp/readlinetest.rb
Readline::VERSION => 6.2

        one
        one
        two
        two
        three
        three
        one[A     # Up-arrow was pressed here, on the empty line; down-arrow would print ">> one[B", right-arrow ">> one[B", etc.



$ ruby tmp/readlinetest.rb
Readline::VERSION => 6.2

        one
        one
        two
        two
        three
        three
        one     # ESC was pressed here, on the empty line.



$ find $MY_RUBY_HOME -name readline.bundle -exec otool -L {} ;
/Users/testuser/.rvm/rubies/ruby-1.9.3-p392/lib/ruby/1.9.1/x86_64-darwin10.8.0/readline.bundle:
/Users/testuser/.rvm/rubies/ruby-1.9.3-p392/lib/libruby.1.9.1.dylib (compatibility version 1.9.1, current version 1.9.1)
/opt/local/lib/libreadline.6.2.dylib (compatibility version 6.0.0, current version 6.2.0)
/opt/local/lib/libncurses.5.dylib (compatibility version 5.0.0, current version 5.0.0)
/usr/lib/libSystem.B.dylib (compatibility version 1.0.0, current version 125.2.11)
/usr/lib/libobjc.A.dylib (compatibility version 1.0.0, current version 227.0.0)
```

#3 - 06/01/2014 03:20 AM - cjheath (Clifford Heath)

David Albert wrote:

> =begin
> I've discovered what I think is a bug in the (({Readline})) module in the standard library.
> =end

vi mode uses an escape character to exit insert mode, enter command mode. ANSI terminals send escape sequences (Escape [ ) for arrow keys. vi and vim resolve this by using short timeouts; if an escape is received and there is no following [within 500msec or so, the escape is processed, otherwise it proceeds to resolve an arrow key. The GNU readline library has the same feature: the keyseq-timeout configuration variable, usually set in your ~/.inputrc

Check whether setting this configuration variable fixes your problems. Add a line like this to ~/.inputrc:

```
set keyseq-timeout 500
```

If you want to do further research, the GNU readline code is here: http://ftp.gnu.org/gnu/readline/
Documentation of the inputrc file: http://cnswww.cns.cwru.edu/php/chet/readline/readline.html#SEC12

Ruby's Readline wrapper for the readline library is pretty rudimentary; for example it does not correctly handle escape sequences, nor does it restore

single-character input mode after a job-control suspend (^Z followed by fg). This should be entered as a separate bug.

#4 - 12/25/2017 06:15 PM - naruse (Yui NARUSE)

- Target version deleted (2.6)

#5 - 08/22/2023 08:44 PM - jeremyevans0 (Jeremy Evans)

- Status changed from Assigned to Closed

Readline was removed from Ruby in 59fd67fc3d405e529e038172e769ff20a8fb5535. If this is still an issue, please file it upstream: https://github.com/ruby/readline-ext/issues