Ruby master - Feature #15771

Add `String#split` option to set `split_type string` with a single space separator

04/16/2019 04:42 AM - 284km (kazuma furuhashi)

Status:	Feedback
Priority:	Normal
Assignee:	
Target version:	

Description

When String#split's separator is a single space character, it executes under split_type: awk.

When you want to split literally by a single space " ", and not a sequence of space characters, you need to take special care. For example, the CSV library detours this behavior like this:

```
if @column_separator == " ".encode(@encoding)
  @split_column_separator = Regexp.new(@escaped_column_separator)
else
  @split_column_separator = @column_separator
end
```

Unfortunately, using a regexp here makes it slower than using a string. The following result shows it is about nine times slower.

```
$ be benchmark-driver string_split_string-regexp.yml --rbenv '2.6.2'
Comparison:
              string:   3161117.6 i/s
              regexp:    344448.0 i/s - 9.18x  slower
```

I want to add a :literal option to execute the method under split_type: string as follows:

```
" a  b   c    ".split(" ")                    # => ["a", "b", "c"]
" a  b   c    ".split(" ", literal: true)     # => ["", "a", "", "b", "", "", "c"]
" a  b   c    ".split(" ", -1)                # => ["a", "b", "c", ""]
" a  b   c    ".split(" ", -1, literal: true) # => ["", "a", "", "b", "", "", "c", "", "", "", ""]
```

Implementation

- https://github.com/284km/ruby/tree/split_space
  - test code: https://github.com/284km/ruby/blob/split_space/test/ruby/test_string.rb#L1708-L1713

History

#1 - 04/18/2019 11:15 PM - 284km (kazuma furuhashi)

pull request: https://github.com/ruby/ruby/pull/2132

#2 - 05/02/2020 07:50 AM - sawa (Tsuyoshi Sawada)

- Description updated

- Subject changed from Add `String#split` option to set split_type string when a single space separator to Add `String#split` option to set `split_type string` with a single space separator

#3 - 05/03/2020 06:22 AM - sawa (Tsuyoshi Sawada)

- Description updated

#4 - 05/03/2020 06:25 AM - sawa (Tsuyoshi Sawada)

- Description updated

#5 - 05/07/2020 04:58 PM - Eregon (Benoit Daloze)

Since splitting on whitespace is the default (ignoring $; which is deprecated in 2.7), maybe we could make split(" ") not special longer-term?

#6 - 05/08/2020 03:29 PM - Dan0042 (Daniel DeLorme)

I've often thought that the default behavior should be tied to nil rather than " ", but in terms of compatibility I don't really think it's worth the change. The proposed option makes it easy to avoid special-casing the " " separator; imho str.split(sep, literal: true) feels cleaner than str.split(sep==" " ? / / : sep).
Not too sure about literal: true though, maybe awk: false would be more meaningful?

#7 - 05/12/2020 12:23 AM - sawa (Tsuyoshi Sawada)

My guess is that, perhaps, even now, it is very rare to use a single space string argument with the expectation to match single or multiple spaces. In such use cases, normally, split is used without an argument, or with a regex argument such as /\s+/ or / +/. If it turns out that it is rare enough, then the behavior of split could be changed to split_type: string altogether.

#8 - 05/12/2020 03:21 AM - znz (Kazuhiro NISHIYAMA)

How about optimization split(/ /) (when regexp is single space only) instead of changing split(" ")?

#9 - 05/12/2020 07:35 AM - nobu (Nobuyoshi Nakada)

znz (Kazuhiro NISHIYAMA) wrote in #note-8:

> How about optimization split(/ /) (when regexp is single space only) instead of changing split(" ")?

Sounds nice. https://github.com/ruby/ruby/pull/3103

#10 - 05/13/2020 03:06 PM - Dan0042 (Daniel DeLorme)

That optimization is nice to have, but I think the point of this ticket is that it's currently not possible to have an arbitrary string separator.

```
str = "aaababababbbabbabaabaaabbbabab"
sep = "x" #or anything except " "
str.gsub("a",sep).split(sep).size #=> 15
sep = " "
str.gsub("a",sep).split(sep).size #=> 9
```

sawa (Tsuyoshi Sawada) wrote in #note-7:

> My guess is that, perhaps, even now, it is very rare to use a single space string argument with the expectation to match single or multiple spaces.

It looks like using a single space string argument is not so rare:
https://pastebin.com/pPyEf2GA

#11 - 05/26/2020 11:56 PM - matz (Yukihiro Matsumoto)

- Status changed from Open to Feedback

I get the point, but we still need a concrete use-case. (Unlike tabs and commas) Space-separated CSV is not common, and consequent spaces are considered as one space usually. I still feel like it's a theoretical concern.

Besides that, literal does not seem to be a right name for the option. In programming languages, the term literal means literal constants (e.g. "literal" or 3), so it can be confusing.

Matz.

#12 - 05/27/2020 04:52 AM - sawa (Tsuyoshi Sawada)

Dan0042 (Daniel DeLorme) wrote in #note-10:

> That optimization is nice to have, but I think the point of this ticket is that it's currently not possible to have an arbitrary string separator.

I agree.

Dan0042 (Daniel DeLorme) wrote in #note-10:

> sawa (Tsuyoshi Sawada) wrote in #note-7:
>
> > My guess is that, perhaps, even now, it is very rare to use a single space string argument with the expectation to match single or multiple spaces.
>
> It looks like using a single space string argument is not so rare:
> https://pastebin.com/pPyEf2GA

I didn't write that using a single space string argument is rare, I wrote that using a single space string argument **with the expectation to match single or multiple spaces** is rare.

In fact, my guess is that using a single space string argument is not rare, and that most of them **expect to match only single space**. I have not confirmed this. If it turns out to be correct, then that would constitute the use cases that are asked for.

#13 - 05/27/2020 04:59 AM - sawa (Tsuyoshi Sawada)

I have a particular use case. I was creating a file that describes UTF-8 characters, which included lines like this:

```
002  !"#$%&'()*+,-./
003 0123456789:;<=>?
004 @ABCDEFGHIJKLMNO
```

The first three characters of each line describes the significant hex-digits of the UTF-8 code, which are followed by a space character that separates the following sixteen characters that belong to that line.

Notice that the first space character after 002 is used as a separator, and the space character right after it is intended to express a literal space character.

I tried to parse each line with a code like this:

```
significant_digits, characters = line.split(" ", 2)
```

but it did not work as I expected, which is:

```
significant_digits # => "002"
characters # => ' !"#$%&'()*+,-./'
```

and I realized the issue mentioned on this ticket.

#14 - 05/27/2020 01:34 PM - matz (Yukihiro Matsumoto)

sawa (Tsuyoshi Sawada), for your "use-case", line.split(/ /, 2) is far better than line.split(" ", 2, literal: true), I think, no matter what keyword we'd choose.

- shorter
- clearer
- need to rewrite anyway

Matz.

#15 - 05/27/2020 09:19 PM - Dan0042 (Daniel DeLorme)

I think it's worth mentioning nobu's comment from the dev meeting:

> have wanted to deprecate that behavior for years, and made non-nil $/ warned.

So what about finally deprecating this behavior? If the incompatibility is too bad it's always possible to go back. sawa's use-case, rather than being about the literal option, is more about the benefit of treating " " as split_type string. And I remember being very surprised about the behavior of str.split(" ") when I started out in ruby.

Maybe:
if $VERBOSE show a warning "use nil or / /" if separator is " "
if !$VERBOSE show a warning "use nil" if separator is " " and matches differently from / /