Ruby trunk - Feature #9816

文字列内の数字を数値として比較するメソッド

05/08/2014 09:37 AM - naruse (Yui NARUSE)

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

Description

文字列内の数字を数値として比較するメソッドを追加しませんか

そのような比較は一般的な用途としてはGUIシェルのファイラーが比較に用いており、
Windows では StrCmpLogicalW が、OS X では NSString:compare:options:へのNSNumericSearch定数が提供されています。
http://msdn.microsoft.com/en-us/library/windows/desktop/bb759947(v=vs.85).aspx
https://developer.apple.com/library/mac/documentation/Cocoa/Reference/Foundation/Classes/NSString_Class/Reference/NSString.html#//apple_ref/c/econst/NSNumericSearch

上記のような処理自体はさほど難しいものではありませんが、Rubyレベルで実装すると大量のオブジェクトを作ってしまいます。
例えば Gem::Version.new("2.1.10".freeze)<=>Gem::Version.new("2.1.9".freeze) は47個、
"2.1.10".freeze.split('.').map(&:to_i)<=>"2.1.9".freeze.split('.').map(&:to_i) だと16個のオブジェクトを作ります。
"2.1.10".freeze.numericcmp"2.1.9".freeze ならば、もちろんオブジェクトは一つも作りません。

なお、上記の例でも示唆していますが、本メソッドは Ruby のバージョン表記の TEENY が2桁になった場合の比較に用いることができます。

パッチは以下の通りです。
なお、メソッド名は String#numericcmp としています。
（String#casecmpを念頭に置いた）

```
diff --git a/string.c b/string.c
index c589c80..66f667f 100644
--- a/string.c
+++ b/string.c
@@ -2569,6 +2569,131 @@ rb_str_casecmp(VALUE str1, VALUE str2)
     return INT2FIX(-1);
 }

+VALUE
+numerical_compare(const char **pp1, const char *p1end, const char **pp2, const char *p2end)
+{
+    const char *s1 = *pp1, *p1, *s2 = *pp2, *p2;
+    ptrdiff_t len1, len2;
+    int r;
+
+    while (s1 < p1end && *s1 == '0') s1++;
+    p1 = s1;
+    while (p1 < p1end && ISDIGIT(*p1)) p1++;
+    len1 = p1 - s1;
+
+    while (s2 < p2end && *s2 == '0') s2++;
+    p2 = s2;
+    while (p2 < p2end && ISDIGIT(*p2)) p2++;
+    len2 = p2 - s2;
+
+    if (len1 != len2) {
+   return INT2FIX(len1 < len2 ? -1 : 1);
+    }
+
+    r = memcmp(s1, s2, len1);
+    if (r) return r < 0 ? INT2FIX(-1) : INT2FIX(1);
+
+    len1 = s1 - *pp1;
+    len2 = s2 - *pp2;
+    if (len1 != len2) {
```

```
+   return INT2FIX(len1 < len2 ? -1 : 1);
+    }
+
+    *pp1 = p1;
+    *pp2 = p2;
+    return Qnil;
+}
+
+/*
+ *  call-seq:
+ *     str.numericcmp(other_str)   -> -1, 0, +1 or nil
+ *
+ *  Variant of <code>String#<=></code>, which considers digits in strings
+ *  are numeric value..
+ *
+ *     "a1".numericcmp("a1")             #=> 0
+ *     "aa".numericcmp("a1")             #=> 1
+ *     "a1".numericcmp("aa")             #=> -1
+ *     "a1".numericcmp("a01")            #=> -1
+ *     "2.1.2".numericcmp("2.1.10")      #=> 1
+ */
+
+static VALUE
+rb_str_numericcmp(VALUE str1, VALUE str2)
+{
+    long len;
+    rb_encoding *enc;
+    const char *p1, *p1end, *p2, *p2end;
+
+    StringValue(str2);
+    enc = rb_enc_compatible(str1, str2);
+    if (!enc) {
+   return Qnil;
+    }
+
+    p1 = RSTRING_PTR(str1); p1end = RSTRING_END(str1);
+    p2 = RSTRING_PTR(str2); p2end = RSTRING_END(str2);
+    if (single_byte_optimizable(str1) && single_byte_optimizable(str2)) {
+   while (p1 < p1end && p2 < p2end) {
+       if (ISDIGIT(*p1)) {
+       if (ISDIGIT(*p2)) {
+           VALUE r = numerical_compare(&p1, p1end, &p2, p2end);
+           if (!NIL_P(r)) return r;
+       }
+       else {
+           return INT2FIX(-1);
+       }
+       }
+       else if (ISDIGIT(*p2)) {
+       return INT2FIX(1);
+       }
+       if (*p1 != *p2) return INT2FIX(*p1 < *p2 ? -1 : 1);
+       p1++;
+       p2++;
+   }
+    }
+    else {
+   while (p1 < p1end && p2 < p2end) {
+            int l1, c1 = rb_enc_ascget(p1, p1end, &l1, enc);
+            int l2, c2 = rb_enc_ascget(p2, p2end, &l2, enc);
+
+            if (0 <= c1 && 0 <= c2) {
+       if (ISDIGIT(*p1)) {
+           if (ISDIGIT(*p2)) {
+           VALUE r = numerical_compare(&p1, p1end, &p2, p2end);
+           if (!NIL_P(r)) return r;
+           }
```

```
+           else {
+           return INT2FIX(-1);
+           }
+       }
+       else if (ISDIGIT(*p2)) {
+           return INT2FIX(1);
+       }
+       if (*p1 != *p2) return INT2FIX(*p1 < *p2 ? -1 : 1);
+       p1++;
+       p2++;
+            }
+            else {
+                int r;
+                l1 = rb_enc_mbclen(p1, p1end, enc);
+                l2 = rb_enc_mbclen(p2, p2end, enc);
+                len = l1 < l2 ? l1 : l2;
+                r = memcmp(p1, p2, len);
+                if (r != 0)
+                    return INT2FIX(r < 0 ? -1 : 1);
+                if (l1 != l2)
+                    return INT2FIX(l1 < l2 ? -1 : 1);
+            }
+       p1 += l1;
+       p2 += l2;
+   }
+    }
+    if (RSTRING_LEN(str1) == RSTRING_LEN(str2)) return INT2FIX(0);
+    if (RSTRING_LEN(str1) > RSTRING_LEN(str2)) return INT2FIX(1);
+    return INT2FIX(-1);
+}
+
 static long
 rb_str_index(VALUE str, VALUE sub, long offset)
 {
@@ -8721,6 +8846,7 @@ Init_String(void)
     rb_define_method(rb_cString, "eql?", rb_str_eql, 1);
     rb_define_method(rb_cString, "hash", rb_str_hash_m, 0);
     rb_define_method(rb_cString, "casecmp", rb_str_casecmp, 1);
+    rb_define_method(rb_cString, "numericcmp", rb_str_numericcmp, 1);
     rb_define_method(rb_cString, "+", rb_str_plus, 1);
     rb_define_method(rb_cString, "*", rb_str_times, 1);
     rb_define_method(rb_cString, "%", rb_str_format_m, 1);
diff --git a/test/ruby/test_string.rb b/test/ruby/test_string.rb
index 8366424..f9c788b 100644
--- a/test/ruby/test_string.rb
+++ b/test/ruby/test_string.rb
@@ -2104,6 +2104,29 @@ class TestString < Test::Unit::TestCase
     assert_equal(1, "\u3042B".casecmp("\u3042a"))
   end

+  def test_numericcmp
+    assert_equal(-1, "2.1.0".numericcmp("2.1.1"))
+    assert_equal(-1, "2.1.9".numericcmp("2.1.10"))
+    assert_equal( 0, "a1".numericcmp("a1"))
+    assert_equal( 1, "aa".numericcmp("a1"))
+    assert_equal(-1, "a1".numericcmp("aa"))
+    assert_equal(-1, "a1".numericcmp("a01"))
+    assert_equal(-1, "a0001".numericcmp("a00001"))
+    assert_equal( 0, "a1a".numericcmp("a1a"))
+    assert_equal( 1, "a1b".numericcmp("a1a"))
+    assert_equal(-1, "a9a".numericcmp("a10a"))
+    assert_equal( 1, "b".numericcmp("a"))
+    assert_equal( 0, "\u30421".numericcmp("\u30421"))
+    assert_equal( 1, "\u3042\u3042".numericcmp("\u30421"))
+    assert_equal(-1, "\u30421".numericcmp("\u3042\u3042"))
+    assert_equal(-1, "\u30421".numericcmp("\u304201"))
+    assert_equal(-1, "\u30420001".numericcmp("\u304200001"))
```

```
+    assert_equal( 0, "\u30421\u3042".numericcmp("\u30421\u3042"))
+    assert_equal( 1, "\u30421\u3044".numericcmp("\u30421\u3042"))
+    assert_equal(-1, "\u30429\u3042".numericcmp("\u304210\u3042"))
+    assert_equal( 1, "\u3044".numericcmp("\u3042"))
+  end
+
   def test_upcase2
     assert_equal("\u3042AB", "\u3042aB".upcase)
   end
```

History

#1 - 05/08/2014 09:50 AM - usa (Usaku NAKAMURA)

+1

自分でいちいち書くのも面倒だし、Gem::Versionをこのために持ってくるのも......という類のメソッドですよね。

#2 - 05/08/2014 10:15 AM - znz (Kazuhiro NISHIYAMA)

numericcmp だと複数の数値が入っている文字列を比較するものというのがわかりにくいと思いました。

最初に思いついたのは versioncmp という名前でしたが、
puppet などで使われているようです。
https://github.com/puppetlabs/puppet/blob/master/lib/puppet/util/package.rb

coreutils だと filevercmp という名前のようです。
https://www.gnu.org/software/coreutils/manual/html_node/Details-about-version-sort.html

rpm だと rpmvercmp という名前のようです。
http://rpm.org/api/4.4.2.2/rpmlib_8h.html

#3 - 05/08/2014 10:58 AM - naruse (Yui NARUSE)

Kazuhiro NISHIYAMA wrote:

> numericcmp だと複数の数値が入っている文字列を比較するものというのがわかりにくいと思いました。
>
> 最初に思いついたのは versioncmp という名前でしたが、
> puppet などで使われているようです。
> https://github.com/puppetlabs/puppet/blob/master/lib/puppet/util/package.rb

名前の候補としてはいいんじゃないですかね。
String に生やしているわけではないので、puppetが使っているのは無視していいと思います。

> coreutils だと filevercmp という名前のようです。
> https://www.gnu.org/software/coreutils/manual/html_node/Details-about-version-sort.html
>
> rpm だと rpmvercmp という名前のようです。
> http://rpm.org/api/4.4.2.2/rpmlib_8h.html

rpmvercmpは論外として、filevercmpは挙動がだいぶ違いますね。
https://github.com/gagern/gnulib/blob/master/lib/filevercmp.c

#4 - 05/08/2014 11:58 AM - tadf (tadayoshi funaba)

名前はともかく俺が欲しいのは filevercmp のほうかもしれない。

```
x #=> ["2.1.10", "2.1.2", "8 layers", "8 layers 2", "8 layers 2.nki", "8 layers.nki", "a16", "a17"]
puts x.sort{|a,b| a.numericcmp(b)}
2.1.2
2.1.10
8 layers
8 layers 2
8 layers 2.nki
8 layers.nki
a16
a17
#=> nil

$ ls -1v
2.1.2
```

```
2.1.10
8 layers
8 layers.nki
8 layers 2
8 layers 2.nki
a16
a17
```

#5 - 05/08/2014 02:25 PM - nobu (Nobuyoshi Nakada)

versioncmpは'-'と'.'だけ特別扱いって感じですかね。
あと、片方だけが数字で終わっているときの終端チェックが抜けているような。

```
diff --git i/string.c w/string.c
index 66f667f..855d74f 100644
--- i/string.c
+++ w/string.c
@@ -2639,6 +2639,11 @@ rb_str_numericcmp(VALUE str1, VALUE str2)
        if (ISDIGIT(*p2)) {
            VALUE r = numerical_compare(&p1, p1end, &p2, p2end);
            if (!NIL_P(r)) return r;
+           if (p1 >= p1end) {
+           if (p2 < p2end) return INT2FIX(-1);
+           break;
+           }
+           else if (p2 >= p2end) return INT2FIX(+1);
        }
        else {
            return INT2FIX(-1);
@@ -2647,7 +2652,13 @@ rb_str_numericcmp(VALUE str1, VALUE str2)
        else if (ISDIGIT(*p2)) {
        return INT2FIX(1);
        }
-       if (*p1 != *p2) return INT2FIX(*p1 < *p2 ? -1 : 1);
+       if (*p1 != *p2) {
+       if (*p1 == '-') return INT2FIX(-1);
+       if (*p2 == '-') return INT2FIX(+1);
+       if (*p1 == '.') return INT2FIX(-1);
+       if (*p2 == '.') return INT2FIX(+1);
+       return INT2FIX(*p1 < *p2 ? -1 : 1);
+       }
        p1++;
        p2++;
    }
@@ -2662,6 +2673,11 @@ rb_str_numericcmp(VALUE str1, VALUE str2)
            if (ISDIGIT(*p2)) {
            VALUE r = numerical_compare(&p1, p1end, &p2, p2end);
            if (!NIL_P(r)) return r;
+           if (p1 >= p1end) {
+               if (p2 < p2end) return INT2FIX(-1);
+               break;
+           }
+           else if (p2 >= p2end) return INT2FIX(+1);
            }
            else {
            return INT2FIX(-1);
@@ -2670,7 +2686,13 @@ rb_str_numericcmp(VALUE str1, VALUE str2)
        else if (ISDIGIT(*p2)) {
            return INT2FIX(1);
        }
-       if (*p1 != *p2) return INT2FIX(*p1 < *p2 ? -1 : 1);
+       if (*p1 != *p2) {
+           if (*p1 == '-') return INT2FIX(-1);
+           if (*p2 == '-') return INT2FIX(+1);
+           if (*p1 == '.') return INT2FIX(-1);
+           if (*p2 == '.') return INT2FIX(+1);
+           return INT2FIX(*p1 < *p2 ? -1 : 1);
+       }
        p1++;
        p2++;
             }
diff --git i/test/ruby/test_string.rb w/test/ruby/test_string.rb
index f9c788b..313e9cf 100644
--- i/test/ruby/test_string.rb
+++ w/test/ruby/test_string.rb
```

```
@@ -2116,6 +2116,9 @@ class TestString < Test::Unit::TestCase
     assert_equal( 1, "a1b".numericcmp("a1a"))
     assert_equal(-1, "a9a".numericcmp("a10a"))
     assert_equal( 1, "b".numericcmp("a"))
+    assert_equal( 1, "a.1".numericcmp("a-1"))
+    assert_equal(-1, "a.1".numericcmp("a.1.a"))
+    assert_equal( 1, "a 1".numericcmp("a.x"))
     assert_equal( 0, "\u30421".numericcmp("\u30421"))
     assert_equal( 1, "\u3042\u3042".numericcmp("\u30421"))
     assert_equal(-1, "\u30421".numericcmp("\u3042\u3042"))
@@ -2125,6 +2128,9 @@ class TestString < Test::Unit::TestCase
     assert_equal( 1, "\u30421\u3044".numericcmp("\u30421\u3042"))
     assert_equal(-1, "\u30429\u3042".numericcmp("\u304210\u3042"))
     assert_equal( 1, "\u3044".numericcmp("\u3042"))
+    assert_equal( 1, "\u3042.1".numericcmp("\u3042-1"))
+    assert_equal(-1, "\u3042.1".numericcmp("\u3042.1.\u3042"))
+    assert_equal( 1, "\u3042 1".numericcmp("\u3042.x"))
   end

   def test_upcase2
```

#6 - 05/08/2014 11:46 PM - akr (Akira Tanaka)

Yui NARUSE wrote:

> なお、メソッド名は String#numericcmp としています。

"2.9" と "2.10" の比較を考えると、
整数の並びの辞書順の比較として 2.9 が大きいとするか、
数 (実数) の比較として 2.10 が大きいとするか、
という考え方の違いがあります。

numericcmp という名前は後者を想起させる気がします。

今回欲しいのは前者だと思うので、それを想起させる名前がいいんじゃないでしょうか。

ではどういう名前にするかというと、あまりいい名前が思いつくわけでもないのですが、intcmp とか？
(intarycmp はあまりにナニ)

versioncmp は機能じゃなくて (大きな) 用法からの命名といえるのかも。

(あと numericcmp は cc という同じ文字の並びの間で単語が分かれるというのが、
ぱっとみたときに単語分割しにくいように思います。)

#7 - 05/09/2014 02:28 AM - kosaki (Motohiro KOSAKI)

Feature#5861 の時と違って、色々とバックグランドの話が提示されたので私は説得されました。

+1

```
+  def test_numericcmp
+    assert_equal(-1, "2.1.0".numericcmp("2.1.1"))
+    assert_equal(-1, "2.1.9".numericcmp("2.1.10"))
+    assert_equal( 0, "a1".numericcmp("a1"))
+    assert_equal( 1, "aa".numericcmp("a1"))
+    assert_equal(-1, "a1".numericcmp("aa"))
+    assert_equal(-1, "a1".numericcmp("a01"))
+    assert_equal(-1, "a0001".numericcmp("a00001"))
```

ここちょっと不思議なんですけど、辞書順だと後者が小さくなって、数値だと同値だと思うのですが、
なんで前者が小さくなるべきなんでしょうか。どこかのファイラーだとそうなってるということ?

あと、ふなばさんの指摘が気になっているので、バージョン文字列以外に想定しているユースーケースが
あれば教えて下さい。

```
+    assert_equal( 0, "a1a".numericcmp("a1a"))
+    assert_equal( 1, "a1b".numericcmp("a1a"))
+    assert_equal(-1, "a9a".numericcmp("a10a"))
+    assert_equal( 1, "b".numericcmp("a"))
+    assert_equal( 0, "\u30421".numericcmp("\u30421"))
+    assert_equal( 1, "\u3042\u3042".numericcmp("\u30421"))
+    assert_equal(-1, "\u30421".numericcmp("\u3042\u3042"))
+    assert_equal(-1, "\u30421".numericcmp("\u304201"))
```

```
+    assert_equal(-1, "\u30420001".numericcmp("\u304200001"))
+    assert_equal( 0, "\u30421\u3042".numericcmp("\u30421\u3042"))
+    assert_equal( 1, "\u30421\u3044".numericcmp("\u30421\u3042"))
+    assert_equal(-1, "\u30429\u3042".numericcmp("\u304210\u3042"))
+    assert_equal( 1, "\u3044".numericcmp("\u3042"))
+  end
+
   def test_upcase2
     assert_equal("\u3042AB", "\u3042aB".upcase)
   end
```

#8 - 05/09/2014 02:48 AM - kosaki (Motohiro KOSAKI)

2014-05-08 7:54 GMT-04:00 Tadayoshi Funaba tadf@dotrb.org:

> 名前はともかく俺が欲しいのは filevercmp のほうかもしれない。
>
> ```
> x #=> ["2.1.10", "2.1.2", "8 layers", "8 layers 2", "8 layers 2.nki", "8 layers.nki", "a16", "a17"]
> puts x.sort{|a,b| a.numericcmp(b)}
> 2.1.2
> 2.1.10
> 8 layers
> 8 layers 2
> 8 layers 2.nki
> 8 layers.nki
> a16
> a17
> #=> nil
>
> $ ls -1v
> 2.1.2
> 2.1.10
> 8 layers
> 8 layers.nki
> 8 layers 2
> 8 layers 2.nki
> a16
> a17
> ```

ためしたところ、Windows Explorerはまたちょっと違う順序を示すようです（Windows8で確認）

```
05/08/2014  10:34 PM                 0 2.1.10.txt
05/08/2014  10:34 PM                 0 2.1.2.txt
05/08/2014  10:35 PM                 0 8 layers 2.nki.txt
05/08/2014  10:35 PM                 0 8 layers 2.txt
05/08/2014  10:35 PM                 0 8 layers.nki.txt
05/08/2014  10:35 PM                 0 8 layers.txt
05/08/2014  10:35 PM                 0 a16.txt
05/08/2014  10:36 PM                 0 a17.txt
```

でも "8 layers 2" が "8 layers" より前に来て嬉しいケースが思いつかないので無視していいと
思ってます。（ドキュメントに多少追記しておくと親切かもだけど）

usaさん、コメントありますか?

#9 - 05/09/2014 05:32 AM - naruse (Yui NARUSE)

tadayoshi funaba wrote:

> 名前はともかく俺が欲しいのは filevercmp のほうかもしれない。

仕様が明確に決まっているのと、バージョン比較というわたしのユースケースには影響がないので、
それでもいい感はありますが、どうなんでしょうね。

Akira Tanaka wrote:

> Yui NARUSE wrote:
>
>> なお、メソッド名は String#numericcmp としています。

> "2.9" と "2.10" の比較を考えると、
> 整数の並びの辞書順の比較として 2.9 が大きいとするか、
> 数 (実数) の比較として 2.10 が大きいとするか、
> という考え方の違いがあります。
>
> numericcmp という名前は後者を想起させる気がします。
> 今回欲しいのは前者だと思うので、それを想起させる名前がいいんじゃないでしょうか。

それぞれ後半が逆な気がしますが、MS的にはlogicalcmpらしいですね。

> ではどういう名前にするかというと、あまりいい名前が思いつくわけでもないのですが、intcmp とか?
> (intarycmp はあまりにナニ)

個人的にはnumericcmpより悪化している気がします。

> versioncmp は機能じゃなくて (大きな) 用法からの命名といえるのかも。

前提としての入力をバージョン文字列に限っていればまぁ機能かもしれません。

> (あと numericcmp は cc という同じ文字の並びの間で単語が分かれるというのが、
> ぱっとみたときに単語分割しにくいように思います。)

それは、akrさんが"cc"という文字列を単語として認識することに最適化されているからではないか
という疑義を持っています。

Motohiro KOSAKI wrote:

> > ```
> > +    assert_equal(-1, "a1".numericcmp("a01"))
> > +    assert_equal(-1, "a0001".numericcmp("a00001"))
> > ```
>
> ここちょっと不思議なんですけど、辞書順だと後者が小さくなって、数値だと同値だと思うのですが、
> なんで前者が小さくなるべきなんでしょうか。どこかのファイラーだとそうなってるということ?

Windows の Explorer や OS X ではこういう並びでした。

> あと、ふなばさんの指摘が気になっているので、バージョン文字列以外に想定しているユースーケースが
> あれば教えて下さい。

Ignore caseや正規化と組み合わせると、(ls -vではなく) OS X のファイルの並びに使える…
とかはまぁ想定はしていますがやらないでしょうな。

#10 - 05/09/2014 05:40 AM - nobu (Nobuyoshi Nakada)

Yui NARUSE wrote:

> > (あと numericcmp は cc という同じ文字の並びの間で単語が分かれるというのが、
> > ぱっとみたときに単語分割しにくいように思います。)
>
> それは、akrさんが"cc"という文字列を単語として認識することに最適化されているからではないか
> という疑義を持っています。

numcmpとか。

#11 - 05/09/2014 06:12 AM - sawa (Tsuyoshi Sawada)

メソッド名は分かりませんが、

`a.foo(b)`

という書き方は対称性を崩していて何となく気持ち悪いので、どうせなら

`String.foo(a, b)`

という書き方も見当していただけないでしょうか。あるいは、`<=>`と同様に、メソッドの前の`.`の要らないメソッドがあるといいんですけどね。

#12 - 05/16/2014 06:09 AM - akr (Akira Tanaka)

Yui NARUSE wrote:

>> (あと numericcmp は cc という同じ文字の並びの間で単語が分かれるというのが、
>> ぱっとみたときに単語分割しにくいように思います。)

> それは、akrさんが"cc"という文字列を単語として認識することに最適化されているからではないか
> という疑義を持っています。

個人的に、cc という単語に慣れているという気はしません。

(ゲシュタルト心理学によれば) 類同の要因というのがあるそうで、おなじものが並んでいると
それがまとまりを感じさせるという話があるようです。
で、c という同じものがふたつ並んでいるのがそこで分割されるように見えない原因とすると、
私個人というよりも広い範囲に成り立ってもおかしくないんじゃないかなぁ、と思っています。

#13 - 09/04/2014 11:18 AM - naruse (Yui NARUSE)

熟考の結果、Gem::Versionと仕様をあわせました。
理由は、

- Gem::Versionでこれを使ってくれればオブジェクトの生成数が減る
- 2.2.0-preview1のようなRubyのバージョンの比較ができる からです。

順序のイメージは Prereleases sort between real releases (newest to oldest) のような感じです

1. 1.0
2. 1.0.b1
3. 1.0.a.2
4. 0.9

```
diff --git a/string.c b/string.c
index bec0bfd..e2b3c6f 100644
--- a/string.c
+++ b/string.c
@@ -2605,6 +2605,232 @@ rb_str_casecmp(VALUE str1, VALUE str2)
     return INT2FIX(-1);
 }

+static int
+version_string_p(VALUE str)
+{
+    const char *p = RSTRING_PTR(str);
+    const char *e = RSTRING_END(str);
+
+    if (!rb_enc_asciicompat(STR_ENC_GET(str))) return FALSE;
+
+    if (!ISDIGIT(*p)) return FALSE;
+    do { if (++p >= e) return TRUE; } while (ISDIGIT(*p));
+
+    while (*p == '.') {
+   if (++p >= e) return FALSE;
+   if (!ISALNUM(*p)) return FALSE;
+   do { if (++p >= e) return TRUE; } while (ISALNUM(*p));
+    }
+
+    if (*p != '-') return FALSE;
+     do {
+   if (++p >= e) return FALSE;
+   if (!ISALNUM(*p) && *p != '-') return FALSE;
+   do { if (++p >= e) return TRUE; } while (ISALNUM(*p) || *p == '-');
+    } while (*p == '.');
+
+    return FALSE;
+}
+
+/* return value: whether end of nueric part is EOS
+ * sp: first nonzero digit
+ * ep: end of digits
+ */
+static void
+search_numerical_str(const char **sp, const char **ep)
+{
+    const char *p = *sp;
```

```
+    const char *e = *ep;
+    assert(p < e);
+    for (;;) {
+   if (*p != '0') break;
+   p++;
+   if (p == e) {
+       *sp = p;
+       goto finish;
+   }
+    }
+    *sp = p;
+    assert(p < e);
+    for (;;) {
+   if (!ISDIGIT(*p)) break;
+   p++;
+   if (p == e) {
+       goto finish;
+   }
+    }
+finish:
+    *ep = p;
+    return;
+}
+
+static VALUE
+numerical_compare(const char **pp1, const char *p1end, const char **pp2, const char *p2end)
+{
+    const char *s1 = *pp1, *p1=p1end, *s2 = *pp2, *p2=p2end;
+    ptrdiff_t len1, len2;
+    int r;
+
+    search_numerical_str(&s1, &p1);
+    search_numerical_str(&s2, &p2);
+
+    /* compre digits length */
+    len1 = p1 - s1;
+    len2 = p2 - s2;
+    if (len1 != len2) return INT2FIX(len1 < len2 ? -1 : 1);
+
+    /* compre numeric value */
+    r = memcmp(s1, s2, len1);
+    if (r) return r < 0 ? INT2FIX(-1) : INT2FIX(1);
+
+    *pp1 = p1;
+    *pp2 = p2;
+    return Qnil;
+}
+
+/*
+ *  call-seq:
+ *     str.versioncmp(other_str)   -> -1, 0, +1 or nil
+ *
+ *  Compare strings as version strings.
+ *
+ *     "a1".versioncmp("a1")            #=> 0
+ *     "aa".versioncmp("a1")            #=> 1
+ *     "a1".versioncmp("aa")            #=> -1
+ *     "a1".versioncmp("a01")           #=> -1
+ *     "2.1.2".numericcmp("2.1.10")     #=> 1
+ */
+
+static VALUE
+rb_str_versioncmp(VALUE str1, VALUE str2)
+{
+    const char *p, *pe, *q, *qe;
+
+    StringValue(str2);
+    if (!version_string_p(str1)) {
+   rb_raise(rb_eArgError, "receiver is not version string '%+"PRIsVALUE"'", str1);
+    }
+    if (!version_string_p(str2)) {
+   rb_raise(rb_eArgError, "argument is not version string '%+"PRIsVALUE"'", str2);
+    }
+
+    p = RSTRING_PTR(str1); pe = RSTRING_END(str1);
```

```
+    q = RSTRING_PTR(str2); qe = RSTRING_END(str2);
+
+    for (;;) {
+   if (*p == '-') {
+hyphen_left:
+       if (*q == '-') goto next_char;
+       while (*q == '.') {
+       if (++q == qe) return INT2FIX(1);
+       }
+       if (*q != 'p') return INT2FIX(ISDIGIT(*q) || 'p' < *q ? -1 : 1);
+       if (++q == qe) return INT2FIX(1);
+       if (*q != 'r') return INT2FIX(ISDIGIT(*q) || 'r' < *q ? -1 : 1);
+       if (++q == qe) return INT2FIX(1);
+       if (*q != 'e') return INT2FIX(ISDIGIT(*q) || 'e' < *q ? -1 : 1);
+       if (++q == qe) return INT2FIX(1);
+       if (*q != '.') {
+       if (*q == '-') {
+           p++;
+           goto hyphen_right;
+       }
+       else if (ISALPHA(*q)) return INT2FIX(-1);
+       q--; /* DIGIT */
+       }
+   }
+   else if (*q == '-') {
+hyphen_right:
+       if (*p == '-') goto next_char;
+       while (*p == '.') {
+       if (++p == pe) return INT2FIX(-1);
+       }
+       if (*p != 'p') return INT2FIX(ISDIGIT(*p) || 'p' < *p ? 1 : -1);
+       if (++p == pe) return INT2FIX(-1);
+       if (*p != 'r') return INT2FIX(ISDIGIT(*p) || 'r' < *p ? 1 : -1);
+       if (++p == pe) return INT2FIX(-1);
+       if (*p != 'e') return INT2FIX(ISDIGIT(*p) || 'e' < *p ? 1 : -1);
+       if (++p == pe) return INT2FIX(-1);
+       if (*p == '-') {
+       q++;
+       goto hyphen_left;
+       }
+       else if (ISALPHA(*p)) return INT2FIX(1);
+       else if (ISDIGIT(*p)) {
+       p--; /* DIGIT */
+       }
+   }
+   else if (ISDIGIT(*p)) {
+       if (ISDIGIT(*q)) {
+       VALUE r = numerical_compare(&p, pe, &q, qe);
+       if(!NIL_P(r)) return r;
+       goto incremented;
+       }
+       else {
+       return INT2FIX(1);
+       }
+   }
+   else if (ISDIGIT(*q)) {
+       return INT2FIX(-1);
+   }
+   else if (ISALPHA(*p)) {
+       if (ISALPHA(*q)) {
+       for (;;) {
+           if (*p != *q) return INT2FIX(*p < *q ? -1 : 1);
+           p++;
+           q++;
+           if (p == pe) {
+           if (q == qe) return INT2FIX(0);
+           if (ISALPHA(*q)) return INT2FIX(-1);
+           goto incremented;
+           }
+           else if (q == qe) {
+           if (ISALPHA(*p)) return INT2FIX(1);
+           goto incremented;
+           }
+           else if (ISALPHA(*p)) {
+           if (!ISALPHA(*q)) return INT2FIX(1);
```

```
+           }
+           else if (ISALPHA(*q)) return INT2FIX(-1);
+           else goto incremented;
+       }
+       continue;
+       }
+       else return INT2FIX(1);
+   }
+   else if (ISALPHA(*q)) {
+       return INT2FIX(-1);
+   }
+   else rb_bug("%s %s",p,q);
+
+next_char:
+   p++;
+   q++;
+
+incremented:
+   while (*p == '.' && ++p != pe);
+   while (*q == '.' && ++q != qe);
+   if (p == pe) {
+       if (q == qe) return INT2FIX(0);
+       if (ISDIGIT(*q)) {
+       return INT2FIX(-1);
+       }
+       else /*if (ISALPHA(*q) || *q == '-')*/ {
+       return INT2FIX(1);
+       }
+   }
+   else if (q == qe) {
+       if (ISDIGIT(*p)) {
+       return INT2FIX(1);
+       }
+       else /*if (ISALPHA(*p) || *p == '-')*/ {
+       return INT2FIX(-1);
+       }
+   }
+    }
+    UNREACHABLE;
+}
+
 #define rb_str_index(str, sub, offset) rb_strseq_index(str, sub, offset, 0)

 static long
@@ -8778,6 +9004,7 @@ Init_String(void)
     rb_define_method(rb_cString, "eql?", rb_str_eql, 1);
     rb_define_method(rb_cString, "hash", rb_str_hash_m, 0);
     rb_define_method(rb_cString, "casecmp", rb_str_casecmp, 1);
+    rb_define_method(rb_cString, "versioncmp", rb_str_versioncmp, 1);
     rb_define_method(rb_cString, "+", rb_str_plus, 1);
     rb_define_method(rb_cString, "*", rb_str_times, 1);
     rb_define_method(rb_cString, "%", rb_str_format_m, 1);
diff --git a/test/ruby/test_string.rb b/test/ruby/test_string.rb
index e8decc0..9e92fb7 100644
--- a/test/ruby/test_string.rb
+++ b/test/ruby/test_string.rb
@@ -2112,6 +2112,41 @@ def test_casecmp
     assert_equal(1, "\u3042B".casecmp("\u3042a"))
   end

+  def test_versioncmp
+    require "rubygems"
+    ary = %w[
+      1
+      2
+      10
+      1.a
+      1.a-a
+      1.a--
+      1.a--.-
+      1.a-1
+      1.a.q
+      1.a--a
+      1.a--1
+      1.a.pre.a
```

```
+      1.a-pre.a
+      1.a.pre-a
+      1.a1
+      1.a2
+      1.aa
+      1.b
+      1.01
+      1.1
+      1.1a
+      1.1-a
+      1.1-b
+      1.1q
+      1.2
+      1.10
+    ]
+    ary.product(ary) do |a, b|
+      assert_equal(Gem::Version.new(a)<=>Gem::Version.new(b), a.versioncmp(b), "#{a.dump}, #{b.dump}")
+    end
+  end
+
   def test_upcase2
     assert_equal("\u3042AB", "\u3042aB".upcase)
   end
```

#14 - 09/12/2014 02:21 PM - akr (Akira Tanaka)

Yui NARUSE wrote:

> 熟考の結果、Gem::Versionと仕様をあわせました。
> 理由は、
>
> - Gem::Versionでこれを使ってくれればオブジェクトの生成数が減る
> - 2.2.0-preview1のようなRubyのバージョンの比較ができる からです。
>
> 順序のイメージは Prereleases sort between real releases (newest to oldest) のような感じです
>
> 1. 1.0
> 2. 1.0.b1
> 3. 1.0.a.2
> 4. 0.9

よくわからなかったので試してみたのですが、
Ruby のバージョンの比較ができるというのは、
preview, rc, patch release の順に並ぶという意味ではないですよね？
(そうしたいという話ではありません。)

```
% cat t.rb
puts %w[
1.9.2-preview1
1.9.2-preview3
1.9.2-rc1
1.9.2-rc2
1.9.2-p0
1.9.2-p136
1.9.2-p180
1.9.2-p290
1.9.2-p320
1.9.2-p330
].sort {|a, b| a.versioncmp(b) }
% ./ruby t.rb
1.9.2-p0
1.9.2-p136
1.9.2-p180
1.9.2-p290
1.9.2-p320
1.9.2-p330
1.9.2-preview1
1.9.2-preview3
1.9.2-rc1
1.9.2-rc2
```

思うのですが、挙動の説明が足りない気がします。

#15 - 10/28/2014 07:59 AM - naruse (Yui NARUSE)

こんな感じで追記してみました

```
+ *  call-seq:
+ *     str.versioncmp(other_str)   -> -1, 0, +1 or nil
+ *
+ *  Compare strings as version strings.
+ *
+ *     "a1".versioncmp("a1")            #=> 0
+ *     "aa".versioncmp("a1")            #=> 1
+ *     "a1".versioncmp("aa")            #=> -1
+ *     "a1".versioncmp("a01")           #=> -1
+ *     "2.1.2".versioncmp("2.1.10")     #=> 1
+ *
+ * The ordering rule of this is the same as Gem::Version.
+ * It raises ArgumentError if the receiver or the argument is not
+ * a version string (/[0-9]+(\.[0-9a-zA-Z]+)*(-[0-9A-Za-z-]+(\.[0-9A-Za-z-]+)*)?/).
+ *
+ * If both of them are valid version strings, their '-' are replaced
+ * with '.pre.', then compare each numeric or character parts numerically
+ * both are numeric, or dictionary order if both are characters, or characters
+ * are prior if one is numeric and another is characters. Returns -1 if the
+ * receiver is prior, or returns 1 if the argument is prior, else returns 0.
```

#16 - 10/28/2014 08:20 AM - akr (Akira Tanaka)

Gem::Version と同じっていうのは本当なんですかね。

とりあえず追記されたという部分の例を Gem::Version でやってみると
結果が一致しないんですが。

```
% ruby -e 'p Gem::Version.new("a1") <=> Gem::Version.new("a01")'
/home/akr/ruby/o0/lib/ruby/2.2.0/rubygems/version.rb:206:in `initialize': Malformed version number string a1 (
ArgumentError)
    from /home/akr/ruby/o0/lib/ruby/2.2.0/rubygems/version.rb:198:in `new'
    from /home/akr/ruby/o0/lib/ruby/2.2.0/rubygems/version.rb:198:in `new'
    from -e:1:in `<main>'
zsh: exit 1     /home/akr/alias/ruby -e 'p Gem::Version.new("a1") <=> Gem::Version.new("a01")
% ruby -e 'p Gem::Version.new("2.1.2") <=> Gem::Version.new("2.1.10")'
-1
% ruby -v
ruby 2.2.0dev (2014-10-25 trunk 48136) [x86_64-linux]
```

a1 は例外になるし、2.1.2 と 2.1.10 は逆の結果になっています。

やはり仕様をちゃんと書くというところに向き合うべきではないのかなぁ、
と思います。

#17 - 10/28/2014 08:29 AM - akr (Akira Tanaka)

Akira Tanaka wrote:

> やはり仕様をちゃんと書くというところに向き合うべきではないのかなぁ、
> と思います。

いや、仕様はそれなりに書いてあるぽいですね。読み方が甘かったです。