L2/16-301

Proposal for consideration by UTC

Newline characters should map to a completely ignorable collation element in the DUCET

Marc Lodewijck (Brussels, Belgium)
October 28, 2016

Assume the following records, which compare tertiary-equal:

```
"xy#", "xy", "x#y"
```

With the UCA “Shifted” option for handling variable collation elements, the sorting result is as follows:

```
[3]   x#y    [1EFF 1F0B | 0020 0020 | 0002 0002 | FFFF 038F FFFF |]
[2]   xy     [1EFF 1F0B | 0020 0020 | 0002 0002 | FFFF FFFF |]
[1]   xy#    [1EFF 1F0B | 0020 0020 | 0002 0002 | FFFF FFFF 038F |]
```

Inserting a variable character makes a string sort *before* the string without it; and appending a variable character makes a string sort *after* the string without it.

With the “Shift-trimmed” option, this is the outcome:

```
[2]   xy     [1EFF 1F0B | 0020 0020 | 0002 0002 | |]
[3]   x#y    [1EFF 1F0B | 0020 0020 | 0002 0002 | FFFF 038F |]
[1]   xy#    [1EFF 1F0B | 0020 0020 | 0002 0002 | FFFF FFFF 038F |]
```

Inserting a variable character anywhere makes a string sort *after* the string without it; and the string having a variable character in the lowest position makes it sort *before* the other string.

The “Shift-trimmed” option is the same as “Shifted”, except that *trailing* high-quaternary weights (from regular characters) are removed from the sort key. This means that, compared with “Shifted”, the “Shift-trimmed” option sorts strings without variable characters before ones with variable characters added—the string without variable characters has an empty fourth level.

Now suppose we wish to sort the following text strings stored in a file:

```
1 air@
2 air-
3 air
4 ab-
5 ab
6 a-b
7 @air
8 *ab
9
```

It is obvious that we won't strip off the newline character(s) at the end of each line, before it is processed further. Assume that any newline character(s) are converted to just the single character 0x0a (denoted hereinafter by the escape sequence "\n"), and thus this is exactly what is in the file, byte by byte:

```
1 air@      0x61 0x69 0x72 0x40 0x0A      # air@\n
2 air-      0x61 0x69 0x72 0x2D 0x0A      # air-\n
3 air       0x61 0x69 0x72 0x0A           # air\n
4 ab-       0x61 0x62 0x2D 0x0A           # ab-\n
5 ab        0x61 0x62 0x0A                # ab\n
6 a-b       0x61 0x2D 0x62 0x0A           # a-b\n
7 @air      0x40 0x61 0x69 0x72 0x0A      # @air\n
8 *ab       0x2A 0x61 0x62 0x0A           # *ab\n
```

Taking into account the various variable-weighting settings described in UTS #10, the following correct orderings are *expected* after collation:

	Non-ignorable	Blanked	Shifted	Shift-trimmed
[1] **air@\n**	[7] @air\n	[4] ab-\n	[8] *ab\n	[5] ab\n
[2] **air-\n**	[8] *ab\n	[5] ab\n	[6] a-b\n	[8] *ab\n
[3] **air\n**	[6] a-b\n	[6] a-b\n	[5] ab\n	[6] a-b\n
[4] **ab-\n**	[5] ab\n	[8] *ab\n	[4] ab-\n	[4] ab-\n
[5] **ab\n**	[4] ab-\n	[1] air@\n	[7] @air\n	[3] air\n
[6] **a-b\n**	[3] air\n	[2] air-\n	[3] air\n	[7] @air\n
[7] **@air\n**	[2] air-\n	[3] air\n	[2] air-\n	[2] air-\n
[8] ***ab\n**	[1] air@\n	[7] @air\n	[1] air@\n	[1] air@\n

However, what we get is as follows:

	Non-ignorable	Blanked	Shifted	Shift-trimmed
[1] **air@\n**	[7] @air\n	[4] ab-\n	[8] *ab\n	[8] *ab\n
[2] **air-\n**	[8] *ab\n	[5] ab\n	[6] a-b\n	[6] a-b\n
[3] **air\n**	[6] a-b\n	[6] a-b\n	[5] ab\n	[5] ab\n
[4] **ab-\n**	[5] ab\n	[8] *ab\n	[4] ab-\n	[4] ab-\n
[5] **ab\n**	[4] ab-\n	[1] air@\n	[7] @air\n	[7] @air\n
[6] **a-b\n**	[3] air\n	[2] air-\n	[3] air\n	[3] air\n
[7] **@air\n**	[2] air-\n	[3] air\n	[2] air-\n	[2] air-\n
[8] ***ab\n**	[1] air@\n	[7] @air\n	[1] air@\n	[1] air@\n

The outcome fails to meet our expectations with the "Shift-trimmed" option. Why doesn't it make a difference whether our records are sorted with the "Shifted" option or the "Shift-trimmed" option? Well, in the default collation element table (DUCET), U+000A maps to a *variable collation element*, and this is correctly reflected at level 4 in the sort keys.

With the parameter values "Shifted", "Shift-trimmed" and "Blanked", variable collation elements are reset to zero at levels one through three, and a new fourth-level weight is appended. That is, the primary weight `L1` of the collation element `[*L1.L2.L3]` of each variable character is shifted down to the fourth level. For example:

```
              *L1  .L2  .L3          .0   .0   .0   .L1
CE{000A} = [*0202.0020.0002]  →  [.0000.0000.0000.0202]
```

Regular characters with primary collation elements (CE not ignorable at level 1) or secondary collation elements (CE ignorable at level 1, but not at level 2) get a high quaternary weight with the "Shifted", "Shift-trimmed" and "Blanked" parameters, higher than that of any variable character—by convention, it is set to FFFF, which is the maximum possible primary weight in the DUCET.

Hence, the following collation elements will be used to produce a sort key for the string "ab\n":

```
CE{0061} = [.1C47.0020.0002.FFFF]  ← [.1C47.0020.0002]
CE{0062} = [.1C60.0020.0002.FFFF]  ← [.1C60.0020.0002]
CE{000A} = [.0000.0000.0000.0202]  ← [*0202.0020.0002]
```

Non-zero weights cannot be omitted while forming sort keys, and this is what we get, consequently:

```
[1C47 1C60 | 0020 0020 | 0002 0002 | FFFF FFFF 0202 |]
```

There are no *trailing* FFFFs (high-quaternary weights) to trim, and hence, with the "Shift-trimmed" variable-weighting setting, the string "ab\n" cannot get an empty quaternary level, which would make it sort before the other strings (having other variable characters in lower positions).

For good measure, following is our detailed data collection:

Non-ignorable

[UTS #10] *"Variable collation elements are not reset to be quaternary collation elements. All mappings defined in the table are unchanged."*

```
[7]   @air\n    [038E 1C47 1D32 1E33 0202 | 0020 0020 0020 0020 0020 | 0002 0002 0002 0002 0002 |]
[8]   *ab\n     [038F 1C47 1C60 0202 | 0020 0020 0020 0020 | 0002 0002 0002 0002 |]
[6]   a-b\n     [1C47 020D 1C60 0202 | 0020 0020 0020 0020 | 0002 0002 0002 0002 |]
[5]   ab\n      [1C47 1C60 0202 | 0020 0020 0020 | 0002 0002 0002 |]
[4]   ab-\n     [1C47 1C60 020D 0202 | 0020 0020 0020 0020 | 0002 0002 0002 0002 |]
[3]   air\n     [1C47 1D32 1E33 0202 | 0020 0020 0020 0020 | 0002 0002 0002 0002 |]
[2]   air-\n    [1C47 1D32 1E33 020D 0202 | 0020 0020 0020 0020 0020 | 0002 0002 0002 0002 0002 |]
[1]   air@\n    [1C47 1D32 1E33 038E 0202 | 0020 0020 0020 0020 0020 | 0002 0002 0002 0002 0002 |]
```

Blanked

[UTS #10] *"Variable collation elements and any subsequent ignorable collation elements are reset so that all weights (except for the identical level) are zero. It is the same as the Shifted Option, except that there is no fourth level."*

```
[4]   ab-\n     [1C47 1C60 | 0020 0020 | 0002 0002 |]
[5]   ab\n      [1C47 1C60 | 0020 0020 | 0002 0002 |]
[6]   a-b\n     [1C47 1C60 | 0020 0020 | 0002 0002 |]
[8]   *ab\n     [1C47 1C60 | 0020 0020 | 0002 0002 |]
[1]   air@\n    [1C47 1D32 1E33 | 0020 0020 0020 | 0002 0002 0002 |]
[2]   air-\n    [1C47 1D32 1E33 | 0020 0020 0020 | 0002 0002 0002 |]
[3]   air\n     [1C47 1D32 1E33 | 0020 0020 0020 | 0002 0002 0002 |]
[7]   @air\n    [1C47 1D32 1E33 | 0020 0020 0020 | 0002 0002 0002 |]
```

Shifted

[UTS #10] *"Variable collation elements are reset to zero at levels one through three. In addition, a new fourth-level weight is appended, whose value depends on the type (. . .). Any subsequent primary or secondary ignorables following a variable are reset so that their weights at levels one through four are zero. (. . .)"*

```
[8]   *ab\n     [1C47 1C60 | 0020 0020 | 0002 0002 | 038F FFFF FFFF 0202 |]
[6]   a-b\n     [1C47 1C60 | 0020 0020 | 0002 0002 | FFFF 020D FFFF 0202 |]
[5]   ab\n      [1C47 1C60 | 0020 0020 | 0002 0002 | FFFF FFFF 0202 |]
[4]   ab-\n     [1C47 1C60 | 0020 0020 | 0002 0002 | FFFF FFFF 020D 0202 |]
[7]   @air\n    [1C47 1D32 1E33 | 0020 0020 0020 | 0002 0002 0002 | 038E FFFF FFFF FFFF 0202 |]
[3]   air\n     [1C47 1D32 1E33 | 0020 0020 0020 | 0002 0002 0002 | FFFF FFFF FFFF 0202 |]
[2]   air-\n    [1C47 1D32 1E33 | 0020 0020 0020 | 0002 0002 0002 | FFFF FFFF FFFF 020D 0202 |]
[1]   air@\n    [1C47 1D32 1E33 | 0020 0020 0020 | 0002 0002 0002 | FFFF FFFF FFFF 038E 0202 |]
```

Shift-trimmed

[UTS #10] *“This option is the same as Shifted, except that all trailing FFFFs are trimmed from the sort key. (. . .)”*

```
[8]   *ab\n     [1C47 1C60 | 0020 0020 | 0002 0002 | 038F FFFF FFFF 0202 |]
[6]   a-b\n     [1C47 1C60 | 0020 0020 | 0002 0002 | FFFF 020D FFFF 0202 |]
[5]   ab\n      [1C47 1C60 | 0020 0020 | 0002 0002 | FFFF FFFF 0202 |]
[4]   ab-\n     [1C47 1C60 | 0020 0020 | 0002 0002 | FFFF FFFF 020D 0202 |]
[7]   @air\n    [1C47 1D32 1E33 | 0020 0020 0020 | 0002 0002 0002 | 038E FFFF FFFF FFFF 0202 |]
[3]   air\n     [1C47 1D32 1E33 | 0020 0020 0020 | 0002 0002 0002 | FFFF FFFF FFFF 0202 |]
[2]   air-\n    [1C47 1D32 1E33 | 0020 0020 0020 | 0002 0002 0002 | FFFF FFFF FFFF 020D 0202 |]
[1]   air@\n    [1C47 1D32 1E33 | 0020 0020 0020 | 0002 0002 0002 | FFFF FFFF FFFF 038E 0202 |]
```

The FFFFs that are *expected* to be trimmed are in blue.

The DUCET cannot yield the expected results for the “Shift-trimmed” option for handling variable collation elements as long as one major change is not issued. To put it plainly, the newline character U+000A should be a completely ignorable character:

```
000A  ; [*0202.0020.0002]   →   000A  ; [.0000.0000.0000]
```

If this were to happen, then it would be reflected as follows in our data collection:

Non-ignorable

(Resulting order is unchanged.)

```
[7]   @air\n    [038E 1C47 1D32 1E33 | 0020 0020 0020 0020 | 0002 0002 0002 0002 |]
[8]   *ab\n     [038F 1C47 1C60 | 0020 0020 0020 | 0002 0002 0002 |]
[6]   a-b\n     [1C47 020D 1C60 | 0020 0020 0020 | 0002 0002 0002 |]
[5]   ab\n      [1C47 1C60 | 0020 0020 | 0002 0002 |]
[4]   ab-\n     [1C47 1C60 020D | 0020 0020 0020 | 0002 0002 0002 |]
[3]   air\n     [1C47 1D32 1E33 | 0020 0020 0020 | 0002 0002 0002 |]
[2]   air-\n    [1C47 1D32 1E33 020D | 0020 0020 0020 0020 | 0002 0002 0002 0002 |]
[1]   air@\n    [1C47 1D32 1E33 038E | 0020 0020 0020 0020 | 0002 0002 0002 0002 |]
```

Blanked

(Resulting order is unchanged—it’s quite obvious.)

```
[4]   ab-\n     [1C47 1C60 | 0020 0020 | 0002 0002 |]
[5]   ab\n      [1C47 1C60 | 0020 0020 | 0002 0002 |]
[6]   a-b\n     [1C47 1C60 | 0020 0020 | 0002 0002 |]
[8]   *ab\n     [1C47 1C60 | 0020 0020 | 0002 0002 |]
[1]   air@\n    [1C47 1D32 1E33 | 0020 0020 0020 | 0002 0002 0002 |]
[2]   air-\n    [1C47 1D32 1E33 | 0020 0020 0020 | 0002 0002 0002 |]
[3]   air\n     [1C47 1D32 1E33 | 0020 0020 0020 | 0002 0002 0002 |]
[7]   @air\n    [1C47 1D32 1E33 | 0020 0020 0020 | 0002 0002 0002 |]
```

Shifted

(Resulting order is unchanged.)

```
[8]   *ab\n     [1C47 1C60 | 0020 0020 | 0002 0002 | 038F FFFF FFFF |]
[6]   a-b\n     [1C47 1C60 | 0020 0020 | 0002 0002 | FFFF 020D FFFF |]
[5]   ab\n      [1C47 1C60 | 0020 0020 | 0002 0002 | FFFF FFFF |]
[4]   ab-\n     [1C47 1C60 | 0020 0020 | 0002 0002 | FFFF FFFF 020D |]
[7]   @air\n    [1C47 1D32 1E33 | 0020 0020 0020 | 0002 0002 0002 | 038E FFFF FFFF FFFF |]
[3]   air\n     [1C47 1D32 1E33 | 0020 0020 0020 | 0002 0002 0002 | FFFF FFFF FFFF |]
[2]   air-\n    [1C47 1D32 1E33 | 0020 0020 0020 | 0002 0002 0002 | FFFF FFFF FFFF 020D |]
[1]   air@\n    [1C47 1D32 1E33 | 0020 0020 0020 | 0002 0002 0002 | FFFF FFFF FFFF 038E |]
```

Shift-trimmed

The new ordering meets our expectations.

```
[5]   ab\n      [1C47 1C60 | 0020 0020 | 0002 0002 | |]
[8]   *ab\n     [1C47 1C60 | 0020 0020 | 0002 0002 | 038F |]
[6]   a-b\n     [1C47 1C60 | 0020 0020 | 0002 0002 | FFFF 020D |]
[4]   ab-\n     [1C47 1C60 | 0020 0020 | 0002 0002 | FFFF FFFF 020D |]
[3]   air\n     [1C47 1D32 1E33 | 0020 0020 0020 | 0002 0002 0002 | |]
[7]   @air\n    [1C47 1D32 1E33 | 0020 0020 0020 | 0002 0002 0002 | 038E |]
[2]   air-\n    [1C47 1D32 1E33 | 0020 0020 0020 | 0002 0002 0002 | FFFF FFFF FFFF 020D |]
[1]   air@\n    [1C47 1D32 1E33 | 0020 0020 0020 | 0002 0002 0002 | FFFF FFFF FFFF 038E |]
```

As compared to the prior results, orderings remain unchanged with the first three variable-weighting settings (“Non-ignorable”, “Blanked”, and “Shifted”) if U+000A is zeroed. Thus, an amendment to the DUCET would only have the desired impact, in line with expectations, and finally perfectly consonant with the rules laid out in UTS #10.

Our data also show that ignoring U+000A can result in shorter sort keys, which may lead to improved performance, as longer sort keys cause serious performance degradations.

Yet, to be precise, the Unicode Standard defines a number of characters that conforming applications should recognize as line terminators. All of them must, like U+000A, be treated as are most non-printing characters in the range U+0000—U+001F (C0 controls), U+007F (delete), and all C1 controls (U+0080—U+009F):

```
000A  ; [*0202.0020.0002] # LINE FEED (in ISO 6429)
000B  ; [*0203.0020.0002] # VERTICAL TABULATION (in ISO 6429)
000C  ; [*0204.0020.0002] # FORM FEED (in ISO 6429)
000D  ; [*0205.0020.0002] # CARRIAGE RETURN (in ISO 6429)
0085  ; [*0206.0020.0002] # NEXT LINE (in ISO 6429)
2028  ; [*0207.0020.0002] # LINE SEPARATOR
2029  ; [*0208.0020.0002] # PARAGRAPH SEPARATOR
```

I question whether there is any need to reconsider the sorting weight of U+0009 (indicated as HORIZONTAL TABULATION in the DUCET) and add it to the above list. The characters listed are used to control the interpretation or display of text, but these characters themselves have no visual or spatial representation, unlike U+0009, which would be the only variable weighted character (that is, either treated as quaternary or not) in General Category “Cc” if its sorting weight is retained in its present form.

* * *

The Unicode Collation Algorithm is maintained in synchronization with the International Standard, ISO/IEC 14651, which also supports the “Shift-trimmed” option, however known as “<forward,position> parameter”. The synchronized version of ISO/IEC 14651 has a Common Template Table (CTT) built for the same repertoire and ordering, but it stores collation elements with a quaternary weight (with the “Shifted” option for variable weighting), and thus the code points listed above are as follows:

```
<U000A> IGNORE;IGNORE;IGNORE;<U000A> % LINE FEED (in ISO 6429)
<U000B> IGNORE;IGNORE;IGNORE;<U000B> % VERTICAL TABULATION (in ISO 6429)
<U000C> IGNORE;IGNORE;IGNORE;<U000C> % FORM FEED (in ISO 6429)
<U000D> IGNORE;IGNORE;IGNORE;<U000D> % CARRIAGE RETURN (in ISO 6429)
<U0085> IGNORE;IGNORE;IGNORE;<U0085> % NEXT LINE (in ISO 6429)
<U2028> IGNORE;IGNORE;IGNORE;<U2028> % LINE SEPARATOR
<U2029> IGNORE;IGNORE;IGNORE;<U2029> % PARAGRAPH SEPARATOR
```

If the DUCET is amended as described above, each of these code points will map to the following collation element in the CTT:

```
IGNORE;IGNORE;IGNORE;IGNORE
```

NOTE: Aside from the “<forward,position>” option, ISO/IEC 14651 further permits a “relaxed” solution for handling variable collation elements, without the “position” parameter, whereby *all* high quaternary weights are backed out of the sort key. As variable characters are nil by default on levels 1 to 3 (recorded as such in the CTT), the behavior of the option “Blanked” can be obtained simply by setting collation strength at level 3, whereas the “Non-ignorable” option could not be provided.

The “Shift-trimmed” variable-weighting setting is not currently implemented in CLDR and ICU, but as regards the root collation data files, they shall of course be deemed to be amended accordingly.

In the meantime, unless given an appropriate tailoring, implementations cannot pass the Canadian benchmark (CAN/CSA-Z243.4.1-98) referred to in UTS #10.

Voilà.

mlodewijck@gmail.com