L2/18-118

Redefining kDefaultSortKey in UAX #38

John H. Jenkins

12 April 2018

Inasmuch as we're going to start using Plane 3 in the near future, we're going to have to revisit the definition of kDefaultSortKey as defined in UAX #38. Right now, it's defined to be a 32-bit integer which actually uses only 31 bits, so that it may be treated as either signed or unsigned and still sort properly.

Bits 0-16 are a normalized form of the code point, set up so that the URO comes first, then Extension A, Extension B, and so on. All seventeen bits are used, and the normalized form of U+2FFFF is 0x1FFFF, so there's no way of including U+30000 in the current scheme without needing one more bit.

Bits 17-22 are used for the residual stroke count (that is, the strokes not included in the radical). Values of 0 through 63 are possible; glyphs with a negative residual stroke count are treated as having value 0.

(Note: We don't document this aspect of the kDefaultSortKey in UAX #38. We should, or else stop doing it.)

Bits 23-30 are the KangXi radical. We don't distinguish simplified and traditional forms of the radical.

As I see it, there are two possible approaches.

1) Use bits 0-17 for the normalized code point, and shift the residual stroke counts and KangXi radical representations up by one bit. This would specifically make the kDefaultSortKey an unsigned 32-bit integer.

2) Redefine the field entirely to make it a 64-bit integer. Support for 64-bit integers is becoming increasingly common, so this is likely not to be a problem. We can therefore be a bit more expansive:

Bits 0-19 can be used for the code point. This would allow us to encode characters beyond from Plane 3 upward having to do further redefinition.

It's also overkill because it lets us move into plane 15, but the advantage is that it leaves us on a nibble boundary. (Five nibbles)

Bits 20-23 can be used for the Unicode block, with 14 and 15 used for the two compatibility blocks, 0 for the URO, 1 for Extension A and so on through 13 for Extension M. Doing it this way obviates the need for a normalized form for the code point. (One nibble)

(Alternatively, if we think Extension N will ever be a thing—and that seems altogether likely—we can use two nibbles.)

Bits 24-27 can be used to flag a character as simplified (0x1) or traditional (0x0). I know that using a whole nibble for this is wasteful when we just need a single bit, but it keeps us on nibble boundaries. As an alternative, we can move or clone this until just after the KangXi radical so that we can distinguish simplified and traditional radical forms, as we do in the Unihan database proper. (One nibble)

Bits 28-31 can be used for the first residual stroke type, 1 through 5. We don't have this data for all of Unihan, but we can allocate space for if and when we acquire the data. (One nibble)

Bits 32-39 can be used for the residual stroke count, theoretically 0 through 255. In theory, we could also make this signed, allowing values -128 to 127. (Two nibbles)

Bits 40-47 can be used for the KangXi radical, 1 through 214. (Two nibbles)

Bits 48 through 63 are left unused. (Four nibbles)

Under the current scheme, U+4E95 gets a kDefaultSortKey value of 0x03840095. (Assuming I did my math correctly.)

Under scheme 1), that would become 0x07080095.

Under scheme 2), we instead would have 0x0000070200004E95. If we have first-stroke data, this becomes 0x0000007230004E95.

Definition scheme 2) makes it easier for a human being to parse: the radical number is 7, it is traditional, the residual stroke count is 2, and the first residual stroke is 3. (The parsing of the default sort key 0x0000A23640730F1D is left as an exercise for the reader.)

My vote would be for 2).

Because the name kDefaultSortKey has been confusing, we could take advantage of this opportunity to rename it (kUnicodeRSChartAndNothingElseSortKey) or drop the name altogether, simply document in UAX #38 that this value is used by Unicode's RS charts.

I should note that the kDefaultSortKey is used only in the radical-stroke charts included in the Unicode Standard, so it should continue to be defined in UAX #38 with the current caveats even if its name is changed or it becomes anonymous.