Subject: Re: Choosing the best way to go full UNICODE
Posted by mirek on Wed, 31 May 2017 09:00:44 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Wed, 31 May 2017 10:30It looks like there are many possible ways to go forward. We can try several things and probably a lot of things will work.

As long as we understand that there is no universal way to make Unicode indexable, but on a case by case basis, you can. The only thing you can universally do is to iterate linearly over Unicode.

If you can iterate linearly, Unicode is indexable...

Quote:
But I still think I gave you a partial solution so many years ago.

To reiterate:
1. Utf8 to Utf16 and vice-versa must be fixed under all scenarios. We also need to add Utf8 to Utf32, but that is trivial compared to Utf16. So proper error recovery must be implemented and 4 byte long sequences must be converted to surrogate pairs.

Agreed.

Quote:
2. The Unicode table must be expanded to more than 2048 characters. Maybe not full range, but Unicode is based on blocks. We can move a bit closer to the CJK block, because for CJK, nobody expects more than the basics. Probably 8k at least.

Not so sure about this - not that important IMO at this point. So I will not get correct ToUpper for many characters - that has little impact on most applications.

Quote:
4. Implement DString? I don't know yet. On the other hand, implementing DSting is easy and it can be dropped in its own header and available for use.

I am now leaning against it. Vector<int> is good enough for utf32 - what we eventually need to do with it.

I am now really thinking that "multibyte" String is the solution. The one that returns a variable sequence of bytes for each position. I am now even thinking this does not need to be bound to graphemes only.

The longterm point with that is to replace WString as processing facility in editors.