Subject: Re: Choosing the best way to go full UNICODE
Posted by mirek on Tue, 30 May 2017 09:45:16 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Tue, 30 May 2017 11:23I meant code point.

Get is the code point.

And what I claimed is that having code point = code unit only half-way solves the problem and you get the least favorite and advantageous encoding in DString/Utf32.

The index of Utf code-point can be easily solved with a string walker that can work for any encoding, from utf8 to 32 and there is no need to tie yourself down with DString supremacy.

Because once you reach "full" Unicode support with DString, you know that most of the non-opaque string code in U++ will use the DString implementation and everything will be biased towards 32 bit.

Interestingly, this is not true at all in current U++ with WString. Most data are always in utf8 String. WString is usually only used temporary to process data. Which was the original intention going to 32 bits 'DString'.

Actually, rethinking graphem problem, maybe we could have

GraphemString

with

String operator[](int i) // returns complete grapheme at position i
int GetCount() // returns number of graphemes

etc...

It would be an 'index' over Utf8 String, grapheme addressed.

See, my problem can be demonstrated by

TextCtrl::Ln

There is the storage for lines of editors. Now 'len' is number of 'cursor characters'. When changes are done to the line, e.g. inserting something at cursor position, text is unpacked to WString, Insert at cursor position is done, then text is packed back to utf8.

Now maybe, if I decide that grapheme is the 'cursor character', going to GraphemeString would be relatively simple. And maybe even faster than using WString...

(All that said, all this will need some way how to provide GUI input / Draw output for graphemes....

but that is a story for another day).