Subject: Re: Choosing the best way to go full UNICODE
Posted by cbpporter on Tue, 30 May 2017 09:23:15 GMT
View Forum Message <> Reply to Message

I 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.

I prefer 1-3 string classes with an 8 bit bias and the same amount of "String walker" classes and once one implements this, I believe in practice the amount of switching from one encoding to another gets minimized.

I my library I only have String, with plans to add WString someday, and the complexity of Utf8 has never made me wish for DString. Conversion to Utf16 only happens in WinAPI context anyway. Plus on the web, Utf8 is the standard. You will probably find that across the globe, on average, Utf8 is still the smaller method of storage. And in CJK context, Utf16 makes a lot of sense, even with occasional surrogate pairs. Historic/academic CJK is where Utf32 shines, but still, in such contexts, Utf is not the most popular.

Anyway, I would go with the same approach I went so many years ago.

Conversion from Utf8 to Ucs2 must be upgraded to Unicode 9.0 compliant Utf8 to Utf16. Utf8 1-4 code units must be converted to a single code point. Correct escaping and error recovery must be implemented. The Unicode standard give you exact deterministic outcomes for any illformated sequences, and combined with the EE you already have in U++, you can get non-destructive error handling. This will leave you with the ability to read and write Unicode. The rest of U++ won't be helped by this, but it is still an important step and will solve a few problems.

Then, one by one, I would convert pieces of code over to a standardized traversal mechanic, be it a string walker class or something else you come up with.