Subject: Re: 16 bits wchar
Posted by cbpporter on Mon, 04 Aug 2008 13:53:58 GMT
View Forum Message <> Reply to Message

luzr wrote on Mon, 04 August 2008 16:07Well, is my understanding correct that your method leads to non-BMP support in UTF-8/String and non-BMP support in WString via surrogate pairs?

First is fine and very good achievement (except that we still have font issue in X11).

Yes, full Unicode code range for conversions and experimental support for size based calculation, fonts and output is what I'm trying to achieve.

Under Linux we will either go with some determined at start font for some ranges of Unicode, or we need to do full font pooling on display operation, and somehow cache the results. I don't know how slow the operation of font enquery is, but with my 1GB of fonts it is pretty slow with full ppoling (i.e. Opera or Character Map).

Quote:WString should be the means of manipulating unicode texts on per-character basis (e.g. in editor).

Using multiple code units per character doesn't disable the use of a text editor or any means of manipulating Unicode texts. It just needs a little bit smarter methods for some operations. I know that using only one word is convenient, but Unicode says that there are up to two words per codepoint and there is no other work around than using 32 bits, which is not a lot better, because not even with UTF32 there isn't a 1:1 relationship between character and display operation of that character. Nonwhitespaces, separators, control characters, combining characters and others must be filtered out, and the end result is the same as if you would use 16 bit chars (where the same operations must be done and I don't think they are done right now). Have you ever tried using combining characters in Upp? And even worse, using combining characters with zero width placeholders, where

I think than one by one all methods that take a string must and traverse it must be reviewed and altered to use a new style of traversing. This only applies to codepoint based addressing, like in GetTextSize. This could be done in an unified way, with iterators, or even "fake index" iterators (which will be a little bit slower than iterators, who should have the same performance as index based traversing).
Anyway performance shouldn't be a problem, because I've been experimenting with a faster method of conversion, which uses local caches short strings up to a static length, bypassing the general algorithm of traverse data, compute code points, determine if escaping is necessary, return length and then recompute data and using a faster method in which the second computation is done only for long strings. It should be faster, but I'm not done benchmarking yet because Linux console apps refuse to print anything since Today (and to connect to mysql, but that is unrelated).

This way there is no need for WString actually, except the fact that it helps as an optimization because Win32 uses it. In the end, we will probably need a full text layout engine, breaking text in multiple segments, and drawing them one by one to support composition, multichar composition,

RTL.