Subject: WString vs Grapheme Cluster idea (with possible flaw)
Posted by mirek on Tue, 25 May 2021 12:25:48 GMT
View Forum Message <> Reply to Message

One of things that remains to be resolved in time is full unicode support. Which basically means that instead of working with UTF-16 characters, we need to start working with (extended) grapheme clusters.

In other words, where now there is wchar, in future needs to be some type with variable number of bytes. Which of course causes all kinds of problems...

However, recently I have got an interesting idea how to solve this "on cheap". Unicode defines codepoints up to about 300 000. Now if we redefine wchar to be 32 bit long, we have about 4 billions of "unusued" bit patterns in wchar. My idea then is to use them as "virtual characters" that eventually map to extended grapheme cluster. It would work this way:

When converting UTF-8 to WString, there would be a global map "grapheme cluster" -> "virtual character". When new grapheme cluster would be encountered, it would be added into this global map and in WString there would always just be 32-bit value.

This would make everything supersimple - e.g. current Ctrl::Key could keep working as it is, every piece of code with WString would work exactly the same (as in most cases, like in EditField, grapheme cluster is exactly what we want to work with). Draw::DrawText would just detect that the character is virtual and obtained the grapheme cluster from the global map...

Now the possible flaw in the reasoning: 4 billions is a large number, but is it large enough to stop worrying about it?

I think in the regular use, it is just fine. But what about webservices and dedicated hackers? What to do if we run out of virtual characters?

Realistically, it is probably ok... But are there any ideas how to eventually handle this (what to do when we reach 4 billions of virtual characters)?

Mirek