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

Subject: Re: WString vs Grapheme Cluster idea (with possible flaw)
Posted by Didier on Tue, 25 May 2021 17:06:00 GMT
View Forum Message <> Reply to Message

Hello Mirek,

I don't master all these caracter support problems but one thing that comes to my mind when I read you're proposition is:

With 300 000 graphemes in a table ==> you get 1.2 MB of used RAM : no problem With 4 000 000 000 graphemes in a table ==> you get 12 GB of used RAM : this will be a problem before you

run out of virtual characters

But I may misunderstand what you propose

Subject: Re: WString vs Grapheme Cluster idea (with possible flaw)
Posted by mirek on Wed, 26 May 2021 06:24:49 GMT
View Forum Message <> Reply to Message

Didier wrote on Tue, 25 May 2021 19:06Hello Mirek,

I don't master all these caracter support problems but one thing that comes to my mind when I read you're proposition is:

With 300 000 graphemes in a table ==> you get 1.2 MB of used RAM : no problem With 4 000 000 000 graphemes in a table ==> you get 12 GB of used RAM : this will be a problem before you run out of virtual characters

But I may misunderstand what you propose

Yes, I guess you are right. Bad idea after all.

Means we will need something like GString, where individual characters are (potentially) represented as String...

Mirek

Subject: Re: WString vs Grapheme Cluster idea (with possible flaw)
Posted by Didier on Wed, 26 May 2021 08:52:44 GMT
View Forum Message <> Reply to Message

Well it really depends on how this is done.

One thing is for sure, all 4 000 000 000 graphemes won't be used at the same time
If we look at how virtual memory is managed, it's just about the same: we have a huge virtual memory but use only a tiny bit of it.
I don't know if it's feasable for graphemes since this has to be performant and we still would have the indirection table to manage which could be the killer

Why not use UTF32 ? sounds like the same (from my very far and poor UTF knowledge)

Subject: Re: WString vs Grapheme Cluster idea (with possible flaw)

Posted by mirek on Wed, 26 May 2021 09:27:08 GMT
View Forum Message <> Reply to Message

Didier wrote on Wed, 26 May 2021 10:52Well it really depends on how this is done.

One thing is for sure, all 4 000 000 000 graphemes won't be used at the same time
If we look at how virtual memory is managed, it's just about the same: we have a huge virtual memory but use only a tiny bit of it.
I don't know if it's feasable for graphemes since this has to be performant and we still would have the indirection table to manage which could be the killer

I think it would indeed work in most of cases. But those remaining ones probably make it unfeasible...

Quote:
Why not use UTF32 ? sounds like the same (from my very far and poor UTF knowledge)

Single grapheme can contain multiple unicode codepoints (that is, multiple UTF32 "characters"), yet we need to treat it as single character in most of scenarios. UTF32 is incomplete solution.

Mirek