Subject: Will UPP support full UNICODE (21bits long codepoint)?
Posted by chivstyle on Sun, 09 Aug 2020 08:40:17 GMT
View Forum Message <> Reply to Message

Hi,

?

Subject: Re: Will UPP support full UNICODE (21bits long codepoint)?
Posted by mirek on Tue, 11 Aug 2020 15:22:11 GMT
View Forum Message <> Reply to Message

chivstyle wrote on Sun, 09 August 2020 10:40Hi,

?

I am sorry, but we currently support just 16-bit unicode at the moment. Extended support was considered, but as there was not enough userbase pressure up until now, it was postponed.

But I guess it is time to fix this.

One problem is that going 32-bit codepoints is just the part of the problem, real solution should deal with all composing issues. But I guess it is a good start...

Mirek

Subject: Re: Will UPP support full UNICODE (21bits long codepoint)?
Posted by Oblivion on Fri, 14 Aug 2020 23:02:03 GMT
View Forum Message <> Reply to Message

I would love to see in U++ too.

One problem is that going 32-bit codepoints is just the part of the problem, real solution should deal with all composing issues. But I guess it is a good start...

As a first step, I can send in the updated tables for Upp::UnicodeCombine() (full set of canonical compositions, including codepoints > 16 bit). Currently this function is missing a lot of compsitions anyway...

They can replace the existing tables (and maybe we can cast down the dword table results (ignore cp > 65535) to word for the time being? Laer we can switch to dword)

But beware, it is a long list of tables, that would deserve another file (unicode.i maybe?)

comb300
comb301
comb302
comb303
comb304
comb306
comb307
comb308
comb309
comb30a
comb30b
comb30c
comb30f
comb311
comb313
comb314
comb31b
comb323
comb324
comb325
comb326
comb327
comb328
comb32d
comb32e
comb330
comb331
comb338
comb342
comb345
comb5b4
comb5b7
comb5b8
comb5b9
comb5bc
comb5bf
comb5c1
comb5c2
comb653
comb654
comb655
comb93c
comb9bc
comb9be

comb9d7
comba3c
combb3c
combb3e
combb56
combb57
combbbe
combbd7
combc56
combcc2
combcd5
combcd6
combd3e
combd57
combdca
combdcf
combddf
combf72
combf74
combf80
combfb5
combfb7
comb102e
comb1b35
comb3099
comb309a
comb110ba
comb11127
comb1133e
comb11357
comb114b0
comb114ba
comb114bd
comb115af
comb11930
comb1d165
comb1d16e
comb1d16f
comb1d170
comb1d171
comb1d172

What do you think?

Best regards,
Oblivion

Subject: Re: Will UPP support full UNICODE (21bits long codepoint)?
Posted by Oblivion on Fri, 14 Aug 2020 23:02:22 GMT
View Forum Message <> Reply to Message

I would love to see in U++ too.

One problem is that going 32-bit codepoints is just the part of the problem, real solution should deal with all composing issues. But I guess it is a good start...

As a first step, I can send in the updated tables for Upp::UnicodeCombine() (full set of canonical compositions, including codepoints > 16 bit). Currently this function is missing a lot of compsitions anyway...

They can replace the existing tables (and maybe we can cast down the dword table results to word for the time being? Laer we can switch to dword)

But beware, it is a long list of tables, that would deserbe another file (unicode.i maybe?)

comb300
comb301
comb302
comb303
comb304
comb306
comb307
comb308
comb309
comb30a
comb30b
comb30c
comb30f
comb311
comb313
comb314
comb31b
comb323
comb324
comb325
comb326
comb327
comb328
comb32d
comb32e
comb330
comb331
comb338

comb342
comb345
comb5b4
comb5b7
comb5b8
comb5b9
comb5bc
comb5bf
comb5c1
comb5c2
comb653
comb654
comb655
comb93c
comb9bc
comb9be
comb9d7
comba3c
combb3c
combb3e
combb56
combb57
combbbe
combbd7
combc56
combcc2
combcd5
combcd6
combd3e
combd57
combdca
combdcf
combddf
combf72
combf74
combf80
combfb5
combfb7
comb102e
comb1b35
comb3099
comb309a
comb110ba
comb11127
comb1133e
comb11357
comb114b0
comb114ba

comb114bd
comb115af
comb11930
comb1d165
comb1d16e
comb1d16f
comb1d170
comb1d171
comb1d172

What do you think?

Best regards,
Oblivion

Subject: Re: Will UPP support full UNICODE (21bits long codepoint)?
Posted by mirek on Fri, 14 Aug 2020 23:39:21 GMT
View Forum Message <> Reply to Message

Oblivion wrote on Sat, 15 August 2020 01:02I would love to see in U++ too.

One problem is that going 32-bit codepoints is just the part of the problem, real solution should deal with all composing issues. But I guess it is a good start...

As a first step, I can send in the updated tables for Upp::UnicodeCombine() (full set of canonical compositions, including codepoints > 16 bit). Currently this function is missing a lot of compsitions anyway...

They can replace the existing tables (and maybe we can cast down the dword table results to word for the time being? Laer we can switch to dword)

But beware, it is a long list of tables, that would deserbe another file (unicode.i maybe?)

comb300
comb301
comb302
comb303
comb304
comb306
comb307
comb308
comb309

comb30a
comb30b
comb30c
comb30f
comb311
comb313
comb314
comb31b
comb323
comb324
comb325
comb326
comb327
comb328
comb32d
comb32e
comb330
comb331
comb338
comb342
comb345
comb5b4
comb5b7
comb5b8
comb5b9
comb5bc
comb5bf
comb5c1
comb5c2
comb653
comb654
comb655
comb93c
comb9bc
comb9be
comb9d7
comba3c
combb3c
combb3e
combb56
combb57
combbbe
combbd7
combc56
combcc2
combcd5
combcd6
combd3e

combd57
combdca
combdcf
combddf
combf72
combf74
combf80
combfb5
combfb7
comb102e
comb1b35
comb3099
comb309a
comb110ba
comb11127
comb1133e
comb11357
comb114b0
comb114ba
comb114bd
comb115af
comb11930
comb1d165
comb1d16e
comb1d16f
comb1d170
comb1d171
comb1d172

What do you think?

Best regards,
Oblivion

Combine is fine, but..

After a lot of thinking, I believe that one step forward is to have

int GraphemeLength(const char *s);
int GraphemeLength(const wchar *s);

functions...

Mirek

Subject: Re: Will UPP support full UNICODE (21bits long codepoint)?
Posted by Oblivion on Sat, 15 Aug 2020 09:12:03 GMT
View Forum Message <> Reply to Message

Quote:int GraphemeLength(const char *s);
int GraphemeLength(const wchar *s);

That requires width tables.

How about generating the width tables with uniset scripts (used by xterm et al.)?

They can generate width tables for doublewidth, ambiguous width, unknown width and combining chars from the unicode database (latest version).
They are quite comprehensive.

I already started using them here (For double width and combining chars, ATM):
https://github.com/ismail-yilmaz/upp-components/blob/master/ CtrlLib/Terminal/Cell.cpp

We can put these into a generic GetCharWidth(int c) function, then utilize them in GetGraphemeLength(const wchar *s)?

This might not be the definitve solution, but it is the battle-tested one out in the wild.

Only real downside is that the tables might need updating albeit ifrequently.

Best regards,
Oblivion

Subject: Re: Will UPP support full UNICODE (21bits long codepoint)?
Posted by mirek on Sat, 15 Aug 2020 09:33:01 GMT
View Forum Message <> Reply to Message

Oblivion wrote on Sat, 15 August 2020 11:12Quote:int GraphemeLength(const char *s);
int GraphemeLength(const wchar *s);

That requires width tables.

How about generating the width tables with uniset scripts (used by xterm et al.)?

They can generate width tables for doublewidth, ambiguous width, unknown width and combining chars from the UnicodeData.txt (latest version).
They are quite comprehensive.

I already started using them here (For double width and combining chars, ATM):
https://github.com/ismail-yilmaz/upp-components/blob/master/ CtrlLib/Terminal/Cell.cpp

We can put these into a generic GetCharWidth(int c) function, then utilize them in GetGraphemeLength(const wchar *s)?

This might not be the definitve solution, but it is the battle-tested one out in the wild.

Only real downside is that the tables might need updating albeit ifrequently.

Best regards,
Oblivion

Just to make sure we are at the same page: I am not speaking about graphics width here, but a number of bytes (or words) that form a single grapheme ("combined character").

EDIT: Still not clear enough: Number of bytes of the first grapheme at s.

Mirek

Subject: Re: Will UPP support full UNICODE (21bits long codepoint)?
Posted by Oblivion on Sat, 15 Aug 2020 10:15:10 GMT
View Forum Message <> Reply to Message

Ah yes, I was thinking about the graphical width (in units),sorry.

But don't we still need to detect base char + combining char(s), which will eventually require tables to be a fast operation? (also for decomposition?).

Best regards,
Oblivion

Subject: Re: Will UPP support full UNICODE (21bits long codepoint)?
Posted by mirek on Sat, 15 Aug 2020 10:44:46 GMT
View Forum Message <> Reply to Message

Oblivion wrote on Sat, 15 August 2020 12:15Ah yes, I was thinking about the graphical width (in units),sorry.

But don't we still need to detect base char + combining char(s), which will eventually require tables to be a fast operation? (also for decomposition?).

Very likely yes, to implement GetGraphemeLength. But I am not at the moment sure whether combining characters are the only source of multi-codepoint graphemes.

Mirek

Subject: Re: Will UPP support full UNICODE (21bits long codepoint)?
Posted by Oblivion on Sat, 15 Aug 2020 11:15:23 GMT
View Forum Message <> Reply to Message

Quote: But I am not at the moment sure whether combining characters are the only source of multi-codepoint graphemes.

Yeah, there are at least surrogate pairs (Since U++ use 16-bit wchar), ligatures and IIRC some hangul graphemes. Anything else that I miss?

Surrogate pairs are rather well formed, but ligatures and multi-codepoint CJK/Devanagari stuff may pose problems...

Edit: Ah yes, what I miss is explained under the grapheme clusters section: http://www.unicode.org/reports/tr29/#Grapheme_Cluster_Bounda ries

---

Subject: Re: Will UPP support full UNICODE (21bits long codepoint)?
Posted by mirek on Sat, 15 Aug 2020 12:07:08 GMT
View Forum Message <> Reply to Message

Oblivion wrote on Sat, 15 August 2020 13:15Quote: But I am not at the moment sure whether combining characters are the only source of multi-codepoint graphemes.

Yeah, there are at least surrogate pairs (Since U++ use 16-bit wchar), ligatures and IIRC some hangul graphemes. Anything else that I miss?

Correct me when I am wrong:

I do not think surrogate pairs are the thing - those are just a way to encode codepoints.

Also ligatures are probably not a concern too, these can be considered characters of its own.

But hangul graphems is what worries me... Not that I have searched too much, but so far I have failed to find simple algo how to combine hangul characters...

Anyway, to explain my way of thinking: So far, in GUI, we have wchar == grapheme equivalency. That e.g. means that inside e.g. EditString, when user enters a character, it is simply inserted at cursor position in the text.

For full unicode, we will need to change 16bit wchars to graphemes. That means that "length" of text will now be number of graphemes, position of text the position of grapheme etc...

Of course, in later phase, we will need to deal with graphics too, but I think that might be relatively easy to changing all text edit routines. (To deal with graphics, Font::operator[](int chr) should probably change to something like Font::operator[](const char *grapheme)...)

Mirek

Subject: Re: Will UPP support full UNICODE (21bits long codepoint)?
Posted by Oblivion on Sat, 15 Aug 2020 13:11:14 GMT
View Forum Message <> Reply to Message

You are correct.

Quote:But hangul graphems is what worries me... Not that I have searched too much, but so far I have failed to find simple algo how to combine hangul characters...

I'm no expert on Eastern-Asian languages either, but I've been studying unicode database for some time and related existing codes that deal with UC The one source code that can give an idea is in perl):

Maybe this could help (as a starter)?: https://metacpan.org/pod/Lingua::KO::Hangul::Util

Ps. The code does not support Hangul letters (jamo) assigned after Unicode 5.2, but it can serve as a starting point. (See also the references below the page).

Also: Hangul syllable types databese for composition/decomposition:
http://www.unicode.org/Public/UNIDATA/HangulSyllableType.txt

Best regards,
Oblivion

Subject: Re: Will UPP support full UNICODE (21bits long codepoint)?
Posted by chivstyle on Mon, 17 Aug 2020 05:22:15 GMT
View Forum Message <> Reply to Message

You are right. The renderer dose not support UNICODE, it treats WCHAR as a character, it not always right. Some other routines such as GetWidth [Font] and subroutines in Font.cpp should be changed to support UNICODE.

Now, I have got a TTF font file that support almost all UNICODE codepoints. I rendered some CJK characters by FreeType, got it.

So, I think it's not a complicated job to support UNICODE.

Subject: Re: Will UPP support full UNICODE (21bits long codepoint)?
Posted by mirek on Mon, 17 Aug 2020 08:17:27 GMT
View Forum Message <> Reply to Message

chivstyle wrote on Mon, 17 August 2020 07:22
So, I think it's not a complicated job to support UNICODE.

Actually, this is not true :)

It is relatively easy to support 32-bit codepoints. But that is only a tiny part of the problem.

Mirek

Subject: Re: Will UPP support full UNICODE (21bits long codepoint)?
Posted by mirek on Mon, 17 Aug 2020 09:01:35 GMT
View Forum Message <> Reply to Message

Oblivion wrote on Sat, 15 August 2020 13:15Quote: But I am not at the moment sure whether combining characters are the only source of multi-codepoint graphemes.

Yeah, there are at least surrogate pairs (Since U++ use 16-bit wchar), ligatures and IIRC some hangul graphemes. Anything else that I miss?

Surrogate pairs are rather well formed, but ligatures and multi-codepoint CJK/Devanagari stuff may pose problems...

Edit: Ah yes, what I miss is explained under the grapheme clusters section:
http://www.unicode.org/reports/tr29/#Grapheme_Cluster_Bounda ries

I guess this might be the path forward:

https://en.wikipedia.org/wiki/HarfBuzz

It looks like most toolkits simply use HarfBuzz anyway... :)

Mirek

Subject: Re: Will UPP support full UNICODE (21bits long codepoint)?
Posted by Oblivion on Mon, 17 Aug 2020 09:50:50 GMT
View Forum Message <> Reply to Message

Quote:It looks like most toolkits simply use HarfBuzz anyway...

Well, this seems to be the best option but I was even afraid of suggesting it, as it means another dependency (and possibly a lot of work) :)

By the way, If you think it's ok, In the meantime we can have better precomposition support. I've attached Charset.cpp with the patched UnicodeCombine for full precompositions support (for 16-bit UCS canonicals only).

(I can also send the extractor code for uppbox if needed)

Best regards,
Oblivion

File Attachments
1) CharSet.cpp, downloaded 133 times

Subject: Re: Will UPP support full UNICODE (21bits long codepoint)?
Posted by mirek on Mon, 17 Aug 2020 12:04:20 GMT
View Forum Message <> Reply to Message

Oblivion wrote on Mon, 17 August 2020 11:50Quote:It looks like most toolkits simply use HarfBuzz anyway...

Well, this seems to be the best option but I was even afraid of suggesting it, as it means another dependency (and possibly a lot of work) :)

By the way, If you think it's ok, In the meantime we can have better precomposition support. I've attached Charset.cpp with the patched UnicodeCombine for full precompositions support (for 16-bit UCS canonicals only).

(I can also send the extractor code for uppbox if needed)

Best regards,
Oblivion

I am sorry to say that because it is mostly due to lack of docs, but I think all this is already better covered with

int UnicodeDecompose(dword codepoint, dword t[MAX_DECOMPOSED], bool only_canonical);
Vector<dword> UnicodeDecompose(dword codepoint, bool only_canonical);

- the reason why this was not quite documented is that above functions are sort of abandoned effort in previous attempt at better Unicode support. Anyway, they are using quite effective z-compressed table (as not to increase .exe size too much). This (and other) tables are producced directly from Unicode tables by uppbox/unicode. And they should also support more than 1 combining marks...

Subject: Re: Will UPP support full UNICODE (21bits long codepoint)?
Posted by Oblivion on Mon, 17 Aug 2020 13:17:28 GMT
View Forum Message <> Reply to Message

 8o

I see. Until we have a better unicode support I think I can use this "undocumented" feature. Thanks!

Best regards,
Oblivion

Subject: Re: Will UPP support full UNICODE (21bits long codepoint)?
Posted by mirek on Tue, 18 Aug 2020 14:55:35 GMT
View Forum Message <> Reply to Message

I have optimised some UnicodeInfo routines and obsoleted UnicodeCombine, while changing its implementation it use UnicodeCompose.

Also I have checked the hangul issue and it rather seems like it is not an issue at all. First of all, algorithm is really trivial, second, maybe it is not even needed as korean texts probably typically contain composed characters anyway, so perhaps it is even better to treat jamo as individual graphemes.

Mirek

Subject: Re: Will UPP support full UNICODE (21bits long codepoint)?
Posted by deep on Thu, 20 Aug 2020 05:10:00 GMT
View Forum Message <> Reply to Message

Hi,

As part of this can we have Indian char sets also working.
There is some similarity between Korean and Indian scripts.

I am willing to work on this with some guidance.

Subject: Re: Will UPP support full UNICODE (21bits long codepoint)?
Posted by mirek on Sat, 22 Aug 2020 16:01:03 GMT
View Forum Message <> Reply to Message

I am now investigating the whole thing in Win32. Let us say we would want just to support 32-bit codepoints as the first step.

Now current model needs following info: a) that the font has glyph for required, b) at least advance width for the glyph.

If we do not want to use Uniscribe, just good old GDI, so far I have only found solution to b), but unfortunately it returns just dimensions of that "box" character that is used in case glyph is missing instead of indicating anyhow that it is actually missing....

```
void Test(int ch, Font fnt)
{
 TIMING("Glyph");
 HFONT hfont = GetWin32Font(fnt, 0);
 VERIFY(hfont);
 if(hfont) {
  HDC hdc = CreateIC("DISPLAY", NULL, NULL, NULL);
  HFONT ohfont = (HFONT) ::SelectObject(hdc, hfont);
  GLYPHMETRICS gm;
  memset(&gm, 0, sizeof(gm));
  MAT2 m_matrix;
  memset8(&m_matrix, 0, sizeof(m_matrix));
  m_matrix.eM11.value = 1;
  m_matrix.eM22.value = 1;
  int gsz = GetGlyphOutlineW(hdc, ch, GGO_NATIVE|GGO_UNHINTED|GGO_METRICS, &gm,
0, NULL, &m_matrix);
  if(gsz == GDI_ERROR)
   DLOG("Failed " << ch);
  if(gm.gmCellIncX != 75)
   DLOG(ch << " " << gm.gmCellIncX << ", " << gm.gmCellIncY << ", " << gm.gmptGlyphOrigin.x);
  ::SelectObject(hdc, ohfont);
  ::DeleteDC(hdc);
 }
}

GUI_APP_MAIN
{
 for(int i = 32; i < 100000; i++)
  Test(i, Font().Height(100).FaceName("MingLiU-ExtB"));
}
```

Subject: Re: Will UPP support full UNICODE (21bits long codepoint)?
Posted by Oblivion on Sat, 22 Aug 2020 17:29:23 GMT
View Forum Message <> Reply to Message

Quote:If we do not want to use Uniscribe, just good old GDI,

Why not? Does U++ also takes care of devices that rely exclusively on GDI?

I would even suggest moving to DirectWrite (IIRC, Novo did suggest moving to Directx some time ago...)

The only GDI function that comes to my mind (to determine if the glyph is missing) is GetGlyphIndices().
It takes a string and let you mark the non-existing ones.

  https://docs.microsoft.com/tr-tr/windows/win32/api/wingdi/nf -wingdi-getglyphindicesw

Best regards,
Oblivion

Subject: Re: Will UPP support full UNICODE (21bits long codepoint)?
Posted by mirek on Sun, 23 Aug 2020 08:06:49 GMT
View Forum Message <> Reply to Message

Oblivion wrote on Sat, 22 August 2020 19:29Quote:If we do not want to use Uniscribe, just good old GDI,

Why not? Does U++ also takes care of devices that rely exclusively on GDI?

I guess it is about a) minimising changes in the code b) implementing in other platforms.

I believe we should do this in several phases:

Phase 1: Moving to code to utf8 (that is String instead of WString) and 32-bit codepoints support.
Phase 2: Basic combining characters support
Phase 3: Advanced text layout

for phase 3, we can either decide to use platform specific library (Uniscribe/pango), which would mean encapsulating it to something platform independent, or just use Harfbuzz on all platforms, which I guess will require dealing with glyph metrics and presence.

Mirek

Subject: Re: Will UPP support full UNICODE (21bits long codepoint)?
Posted by Oblivion on Sun, 23 Aug 2020 08:52:24 GMT
View Forum Message <> Reply to Message

Quote:
for phase 3, we can either decide to use platform specific library (Uniscribe/pango), which would mean encapsulating it to something platform independent, or just use Harfbuzz

To my knowledge, the latest version of pango can be compiled on Windows too (but satisfying its dependencies might not worth the effort). Also, the latest version of it seems to support harfbuzz as a backend.

Subject: Re: Will UPP support full UNICODE (21bits long codepoint)?
Posted by mirek on Sun, 23 Aug 2020 11:15:56 GMT
View Forum Message <> Reply to Message

Oblivion wrote on Sun, 23 August 2020 10:52Quote:
for phase 3, we can either decide to use platform specific library (Uniscribe/pango), which would mean encapsulating it to something platform independent, or just use Harfbuzz

To my knowledge, the latest version of pango can be compiled on Windows too (but satisfying its dependencies might not worth the effort). Also, the latest version of it seems to support harfbuzz as a backend.

Well, pango is definitely using harfbuzz, it is AFAIK even developed by the same person...

Subject: Re: Will UPP support full UNICODE (21bits long codepoint)?
Posted by mirek on Mon, 24 Aug 2020 09:31:21 GMT
View Forum Message <> Reply to Message

https://stackoverflow.com/questions/54050095/how-to-tell-if-a-surrogate-pair-unicode-character-is-supported-by-the-font

Looks like GetCharacterPlacement should do what we need...

Will test soon.

Mirek