fontwrap

Michiel Kamermans
www.nihongoresources.com

June 7, 2008

1 What is fontwrap?

fontwrap is a PerlTEX package for automatically adding font tags in multilingual documents. More specifically, it adds font tags between unicode block changes in documents that are encoded in UTF8 unicode (which is, thankfully these days, pretty much any new multilingual document).

The whole reason most of us use TEX or LATEX or the newer LATEX 2_ε or whichever flavour of TEX you like to use, is because it lets you write your document with a minimal amount of placing control codes inside the actual text you're writing. Most of the time, your text will just be text, and you'll be damned if you have to add all kinds of special codes because that will make the source file less readable. However, when you're working in a multilingual TEX document, you might find you're wrapping bits of "foreign" text in with macros that ensure the right font or other visual styling makes its way to the final document. fontwrap was designed to remove the need for that practice, so that your document stays readable.

If you look closely at the example paragraph in block 1, you will see that all the different languages use a different font. The English font is *Palatino Linotype*, the font for Japanese is *Ume Mincho*, the font for Chinese is *SimHei*, for Korean *BatangChe*, for Arabic *Traditional Arabic*, Cyrillic uses *Dotum*,

Even though I am writing this on an English operating system in an English text editor, I can input quite a lot of different language. I can do this, because of the power of unicode: English, 日本語, 中國話, 한글, 조선글, اللُّغَةُ العَرَبِيَّة , Русский язык, *Ελληνικά, Tiếng Việt,* ภาษาไทย , *עִבְרִית and a whole scala of other languages all use different scripts, which all have their own place in the unicode 5.0 world.*

block 1: A paragraph using many different unicode blocks

Greek uses *Arno Pro*, Thai uses *Cordia New*, and for Hebrew I used *Times New Roman*. For those wondering, Vietnamese actually uses the Latin and Latin Extended Additional blocks, so it uses the same *Palatino Linotype* font as the English text.

In normal TEX , getting all the languages marked with the right fonts, with all the commas and spaces using the same font as the English text, require a mad amount of font markup, but with fontwrap this requires no markup beyond the 'fontwrap' command: all I had to do to get the text to use all these different fonts is tell fontwrap which fonts to use for which blocks in my frontmatter, and wrap write the paragraph exactly as you see it in this pdf file into my .tex — wrapping it in the \fontwrap{} macro then takes care of all my fonty needs.

2 Getting fontwrap working for your document

The basic procedure for getting fontwrap to work in your document is really quite straightforward. First, we must make sure to actually use it:

```
\usepackage{autfont}
```

The rest of the code comes in the document body itself. Before we do anything with fontwrap, it is usually a good idea to tell it which fonts to use for which unicode blocks. There is one *catch-all* command to do this, which sets the same font for every block, and several \set commands for both single blocks, and informal multi-block groups. In this document, for instance, I use this:

```
% set up fontwrap's default font.
\setfontwrapdefaultfont{Bitstream Cyberbit}
```

```
% set specific unicode groups
\setunicodegroupfont{Arabic}{Traditional Arabic}
\setunicodegroupfont{Latin}{Palatino Linotype}
\setunicodegroupfont{Japanese}{Ume Mincho}
\setunicodegroupfont{Chinese}{SimHei}
\setunicodegroupfont{Korean}{BatangChe}
\setunicodegroupfont{Cyrillic}{Dotum}
\setunicodegroupfont{Greek}{Arno Pro}

% thai and hebrew have no group, just a block
\setunicodeblockfont{Thai}{Cordia New}
\setunicodeblockfont{Hebrew}{Times New Roman}
```

Of course, you can set as few or as many as you like, or more importantly as is appropriate. If you're using a bilingual document, setting the catch-all binding and an extra font for the "foreign" bits is all you have to do. After having set up the font bindings in this way, all that's left is to type in whichever mix of languages you please, and surround your text with the \fontwrap macro:

```
\fontwrap{
  the verbatim environment used to make this
  block of text only supports Latin, but you
  would be free to type whatever you like in
  this macro.
}
```

The only downside to this is that I cannot show the actual text from example paragraph 1, because the { verbatim} environment cannot handle more than just Latin, and is one of the few blocks where fontwrap should not be used - adding font tags inside a verbatim block means you're going to get the TEX commands in your final output, instead of having them processed, because that's what verbatim does!

Moving on, \fontwrap does not look into other macros and environments by default. If you want it to process text in macros such as \emph

or \caption then you need to explicitly tell it that it is allowed to do this. This command, and the equivalent command for environments, goes in the preamble:

```
% allow processing of content for the following macros:
\setfontwrapallowedmacros{section,subsection,
                          subsubsection,paragraph,
                          subparagraph,emph, caption,
                          ... }
% allow processing of content for the following environments:
\setfontwrapallowedenvironments{tabular, ... }
```

whenever \fontwrap is now used, it will process text in general document structure macros, as well as the tabular environment, which is useful if we use "foreign" text in any tables we're bound to end up using.

And with that the basic use is pretty much covered.

3 Available commands

First off, \fontwrap of course:

```
\fontwrap{ ... }
```

and wrapped in the fontwrap verbatim environment in case whitespace really, really matters:

```
\begin{fontwrapverbatim}
  \fontwrap{ ... }
\end{fontwrapverbatim}
```

Secondly, the allowances:

```
\setfontwrapallowedmacros{comma delimited list}
\setfontwrapallowedenvironments{comma delimited list}
```

Thirdly, the font setup commands:

```
\setunicodegroupfont{block name}{font name}
\setunicodeblockfont{block name}{font name}
```

Arabic, Chinese, CJK (which combines all Chinese, Japanese and Korean blocks), Cyrillic, Diacritics, Greek (including some Coptic), Korean, Japanese, Latin, Mathematics, Phonetics, Punctuation, Symbols, Yi and finally, Other, which is just a lump category for everything else, really…

block 2: All available informal group names

There are several informal groups available, which are listed in block 2. Also not unimportant to note: these are all case sensitive. The "other" group is a bit of an eyesore, but for now it will have to do. Of course, Linear B and Ethiopian form informal groups too, but I just don't use them, so they will be given their own group when I'm done refining fontwrap, really.

In addition to these groups, there are also the individual blocks, in case there is no group for what you want to set a font for, such as Hebrew, Thai, or really exotic things like Cuneiform or Byzantine musical symbols! There are a total of 158 blocks available for font binding, listed in block 3.

These, too, are case sensitive.

4 Running PerlTEX and possible errors

Running TEX files that use fontwrap means you have to use PerlTEX to get it all to work. Luckily, PerlTEX is just a TEX wrapper, so you can tell it which TEX engine to use and it will. Because fontwrap relies on the fontspec package, we have to use XeTEX:

```
perltex --latex=xelatex myfile.tex
```

This should run fine, but there are three problems you might run into.

AegeanNumbers, AlphabeticPresentationForms, AncientGreekMusicalNotation, AncientGreekNumbers, Arabic, ArabicPresentationFormsA, ArabicPresentationFormsB, ArabicSupplement, Armenian, Arrows, Balinese, BasicLatin, Bengali, BlockElements, Bopomofo, BopomofoExtended, BoxDrawing, BraillePatterns, Buginese, Buhid, ByzantineMusicalSymbols, Cherokee, CJKCompatibility, CJKCompatibilityForms, CJKCompatibilityIdeographs, CJKCompatibilityIdeographsSupplement, CJKRadicalsSupplement, CJKStrokes, CJKSymbolsandPunctuation, CJKUnifiedIdeographs, CJKUnifiedIdeographsExtensionA, CJKUnifiedIdeographsExtensionB, CombiningDiacriticalMarks, CombiningDiacriticalMarksforSymbols, CombiningDiacriticalMarksSupplement, CombiningHalfMarks, ControlPictures, Coptic, CountingRodNumerals, Cuneiform, CuneiformNumbersandPunctuation, CurrencySymbols, CypriotSyllabary, Cyrillic, CyrillicExtendedA, CyrillicExtendedB, CyrillicSupplement, Deseret, Devanagari, Dingbats, DominoTiles, EnclosedAlphanumerics, EnclosedCJKLettersandMonths, Ethiopic, EthiopicExtended, EthiopicSupplement, GeneralPunctuation, GeometricShapes, Georgian, GeorgianSupplement, Glagolitic, Gothic, GreekandCoptic, GreekExtended, Gujarati, Gurmukhi, HalfwidthandFullwidthForms, HangulCompatibilityJamo, HangulJamo, HangulSyllables, Hanunoo, Hebrew, HighPrivateUseSurrogates, HighSurrogates, Hiragana, IdeographicDescriptionCharacters, IPAExtensions, Kanbun, KangxiRadicals, Kannada, Katakana, KatakanaPhoneticExtensions, Kharoshthi, Khmer, KhmerSymbols, Lao, LatinExtendedAdditional, LatinExtendedA, LatinExtendedB, LatinExtendedC, LatinExtendedD, LatinSupplement, LetterlikeSymbols, Limbu, LinearBIdeograms, LinearBSyllabary, LowSurrogates, MahjongTiles, Malayalam, MathematicalAlphanumericSymbols, MathematicalOperators, MiscellaneousMathematicalSymbolsA, MiscellaneousMathematicalSymbolsB, MiscellaneousSymbols, MiscellaneousSymbolsandArrows, MiscellaneousTechnical, ModifierToneLetters, Mongolian, MusicalSymbols, Myanmar, NewTaiLue, NKo, NumberForms, Ogham, OldItalic, OldPersian, OpticalCharacterRecognition, Oriya, Osmanya, PhagsPa, Phoenician, PhoneticExtensions, PhoneticExtensionsSupplement, PrivateUseArea, Runic, Shavian, Sinhala, SmallFormVariants, SpacingModifierLetters, Specials, SuperscriptsandSubscripts, SupplementalArrowsA, SupplementalArrowsB, SupplementalMathematicalOperators, SupplementalPunctuation, SupplementaryPrivateUseAreaA, SupplementaryPrivateUseAreaB, SylotiNagri, Syriac, Tagalog, Tagbanwa, Tags, TaiLe, TaiXuanJingSymbols, Tamil, Telugu, Thaana, Thai, Tibetan, Tifinagh, Ugaritic, UnifiedCanadianAboriginalSyllabics, VariationSelectors, VariationSelectorsSupplement, VerticalForms, YiRadicals, YiSyllables, and finally YijingHexagramSymbols.

block 3: All 158 blocks available in unicode 5.0

No unicode mapping available You get this error when TEX uses a font that cannot represent the unicode glyphs you have written. For instance, using something other than Latin text in a verbatim block will cause this error. It's not fatal in any way, it just means that you will see empty blocks in your final document.

Free to wrong pool PerlTEX uses Perl *(fairly obviously)* but it does so sort of multithreaded. It also uses the Perl "safe" module, and that's where things go funky. The combination of multithread perl and "safe" can lead to perl trying to free the memory it used, but failing at this because it tries to do so in entirely the wrong thread. This is completely inconsequential, other than that it can lead to memory leaks. Now, I made sure to unset all the perl variables I use once fontwrap is done, so you shouldn't run into any problems *(unless maybe you were counting the bytes by hand)* .

Overfull/underfull hbox The boon of TEX, this means that a particular sentence is made up of letters and spaces in such a way that TEX cannot really get the glue stretched properly for it to look nice in your final document. You're going to have to go in, and fix the problem yourself by rephrasing the sentence... either that or leave it in and turn off whatever visual notification for problematic hboxes you use during draft generation.

5 The end…

And… I think that's it. I can't think of anything more to tell you with respects to using fontwrap. If you have any questions you can always check out the .sty file, or contact me through the contact page on my website, http://www.nihongoresources.com.

Enjoy!

- Mike Kamermans