Putting the Cork back in the bottle

Improving Unicode support in TEX

Mojca Miklavec & Arthur Reutenauer

Genesis

As a part of its *Summer of Code* programs, Google sponsors us to work on projects related to TEX.

I chose to work on "making TEX more Unicode-compliant".

The Name of the Game

Close

The Name of the Game

What is Unicode?

Close

The Name of the Game

What is Unicode?

A universal character set, suitable for representing any writing system.

Close

The Name of the Game

What is Unicode?

A universal character set, suitable for representing any writing system.

What is TEX?

The Name of the Game

What is Unicode?

A universal character set, suitable for representing any writing system.

What is TEX?

'You kidding me?

Close

The Name of the Game

What is Unicode?

A universal character set, suitable for representing any writing system.

What is TEX?

'You kidding me?

What does it mean for a TEX-based system to be Unicode-compliant?

The Name of the Game

What is Unicode?

A universal character set, suitable for representing any writing system.

What is TEX?

'You kidding me?

What does it mean for a TEX-based system to be Unicode-compliant?

Can you repeat the question?

A First Step

Supporting Unicode implies to support – at least – UTF-8 encoding. All of the TEX macro packages can accomodate it, but until very recent times, some parts of the core support files ignored it completely, in particular, hyphenation patterns. They used various "legacy" encodings known to TEX.

This was a problem when XETEX was integrated in TEX Live in 2007.

Jonathan Kew then devised a way to convert the patterns to UTF-8 on the fly, if needed. He wrapped them in files called `xu-<hyphen>.tex` (pronounced "zoo hyphen"). These detect if they are running XETEX or some other TEX engine, and convert the patterns to UTF-8 in the former case: you make characters active, and define them to yield the corresponding UTF-8 byte sequence.

Patterns

For TEX Live 2008 we wanted to address the problem the other way round: the input files should be in UTF-8, and we should convert them to TEX's font encodings when using 8-bit engines.

We also wanted to make a clear distinction between the patterns and the TEX support code: `\catcode`'s, `\lccode`'s (and other things you don't want to hear about).

Finally, we wished to adopt a clean naming scheme for the languages at stake, and we chose IETF language tags for that, a.k.a. RFC 4646. It was the only standard we found that could name all the language variants we needed to name. ISO codes simply weren't enough.

The next two slides give an overview of this strategy.

Loading the patterns

The top-level file is called loadhyph-<language code>.tex, **like here for Slovenian:**

```
% Test whether we received one or two arguments

\def\testengine#1#2!{\def\secondarg{#2}}

% That's Tau (as in Taco or TEX, Tau-Epsilon-Chi),
% a 2-byte UTF-8 character

\testengine T!\relax

% Unicode-aware engines (such as XeTeX or LuaTeX)
% only see a single (2-byte) argument

\ifx\secondarg\empty
    \message{UTF-8 Slovenian Hyphenation Patterns}
\else
    \message{EC Slovenian Hyphenation Patterns}
    \input conv-utf8-ec.tex
\fi
\input hyph-sl.tex
```

Converting the patterns

The converter files are called conv-utf8-<font encoding>.tex.

Extract of conv-utf8-ec.tex:

```
\catcode"C4=\active
\catcode"C5=\active

\def^^c4#1{%
 \ifx#1^^8d^^a3\else % č - U+010D
\fi}

\def^^c5#1{%
 \ifx#1^^a1^^b2\else % š - U+0161
 \ifx#1^^be^^ba\else % ž - U+017E
\fi\fi}

%
% ensure all the chars above have valid lccode's
%

\lccode"A3="A3 % č - U+010D
\lccode"B2="B2 % š - U+0161
\lccode"BA="BA % ž - U+017E
```

Taming the patterns

In many cases, things don't work as smoothly as they could be expected to.

Some languages use patterns that try to accomodate T1 and OT1 in the same file. This happens for German, French, Danish, Latin.

Some files can be customized to load completely different pattern sets (Russian, Ukrainian).

Sometimes, Unicode is inherently bad at representing the language at hand (Ancient / Polytonic Greek).

Sometimes Babel isn't on our side (Serbian).

Genesis
Patterns
Intermission
More Unicode
Thanks
Close

Breeding the patterns

The "new" patterns have been available for a few weeks on CTAN under the name `hyph-utf8`, and have been imported into TEX Live for inclusion in the 2008 DVD. They are the basis for language support in plain TEX and LATEX through Babel, and have been integrated in the ConTEXt multilingual infracstructure.

Genesis
Patterns
Intermission
More Unicode
Thanks
Close

Intermission

What we have learned in the process:

Do TEX and Unicode belong together?

Close

Do TEX and Unicode belong together?

All the data is easy to represent in Unicode.

Do TEX and Unicode belong together?

All the data is easy to represent in Unicode.

The real problem was to integrate the patterns in the general TEX landscape,and to preserve Holy Backward Compatibility.

How many languages?

~~42~~. 49.

ar	Arabic
fa	Farsi
eu	Basque
bg	Bulgarian
cop	Coptic
hr	Croatian
cs	Czech
da	Danish
nl	Dutch
eo	Esperanto
et	Estonian
fi	Finnish
fr	French
de-1901	German, "old" spelling
de-1996	German, "new" spelling
el-monoton	Modern Greek, monotonic spelling
el-polyton	Modern Greek, polytonic spelling
grc	Ancient Greek
grc-x-ibycus	Ancient Greek in Ibycus encoding
hu	Hungarian
is	Icelandic
id	Indonesian
ia	Interlingua
ga	Irish
it	Italian
la	Latin
mn-cyrl	Mongolian, Cyrillic script
mn-cyrl-x-2a	Mongolian, Cyrillic script (new patterns)
no	Norwegian
nb	Norwegian Bokmål
nn	Norwegian Nynorsk
zh-latn	Chinese Pinyin
pl	Polish
pt	Portuguese
ro	Romanian
ru	Russian
sr-latn	Serbian in the Latin script
sr-cyrl	Serbian in the Cyrillic script
sh-latn	Serbo-Croatian in the Latin script
sh-cyrl	Serbo-Croatian in the Cyrillic script
sl	Slovene
es	Spanish
sv	Swedish
tr	Turkish
en-gb	British English
en-us	American English
uk	Ukrainian
hsb	Upper Sorbian
cy	Welsh

More Unicode

Actually, supporting Unicode implies much more than that: my original proposal for Google Summer of Code included, in particular, to better handle *combining characters* in XeTeX and luaTeX.

Combining characters are Unicode's diacritical marks: you put them after a *base character* to add an accent to the latter.

For example, Unicode character U+017E LATIN SMALL LETTER Z WITH CARON (ž) can also be represented as the sequences of two characters U+007A LATIN SMALL LETTER Z (z) followed by U+030C COMBINING CARON (ˇ).

Unicode specifies algorithms, known as normalization, to transform character sequences in fully decomposed or fully composed form.

Handling normalization natively

Close

Handling normalization natively

XeTeX has been recently extended to support this at the engine level.

Handling normalization natively

XeTeX has been recently extended to support this at the engine level.

In luaTeX, it can be handled in the macro package thanks to appropriate hooks,which it was ConTeXt Mark IV already does.

Handling normalization natively

XeTeX has been recently extended to support this at the engine level.

In luaTeX, it can be handled in the macro package thanks to appropriate hooks,which it was ConTeXt Mark IV already does.

Problem solved!