Hyphenation patterns in ConTEXt

Hans Hagen

Abstract

A brief discussion of hyphenation patterns, exceptions, and TEX languages, especially in ConTEXt.

1 Pattern files

TEX has two mysterious commands that the average user will never or seldom meet:

```
\hyphenation{as-so-ciates}
\patterns   {.ach4}
```

Both commands can take multiple strings, so in fact both commands should be plural. The first command can be given any time and can be used to tell TEX that a word should be hyphenated in a certain way. The second command can only be issued when TEX is in virgin mode, i.e. starting with a clean slate. Normally this only happens when a format is generated.

The second command is more mysterious than the first one and its entries are a compact way to tell TEX between what character sequences it may hyphenate words. The numbers represent weights and the (often long) lists of such entries are generated with a special program called `patgen`. Since making patterns is work for specialists, we will not go into the nasty details here.

In the early stage of ConTEXt development it came with its own pattern files. Their names started with `lang-` and their suffixes were `pat` and `hyp`.

However, when ConTEXt went public, I was convinced to drop those files and use the files already available in distributions. This was achieved by using the ConTEXt filename remapping mechanism. Although those files are supposed to be generic, this is not always the case, and it remains a gamble if they work with ConTEXt. Even worse, their names are not consistent, and the names of some files as well as locations in the tree keep changing. The price ConTEXt users pay for this is lack of hyphenation until such changes are noticed and taken care of. Because constructing the files is an uncoordinated effort, all pattern files have their own characteristics, most noticably their encoding.

After the need to adapt the name mapping once again, I decided to get back to providing ConTEXt specific pattern files. Pattern cooking is a special craft and TEX users may count themselves lucky that it's taken care of. So, let's start with thanking all those TEX experts who dedicate their time and effort to get their languages hyphenated. It's their work we will build (and keep building) upon.

In the process of specific ConTEXt support, we will take care of:

- consistent naming, i.e. using language codes when possible as a prelude to a more sophisticated naming scheme, taking versions into account
- consistent splitting of patterns and hyphenation exceptions in files that can be recognized by their suffix
- making the files encoding independent using named glyphs
- providing a way to use those patterns in plain TEX as well

Instead of using a control sequence for the named glyphs, we use a different notation:

```
[ssharp] [zcaron] [idiaeresis]
```

The advantage of this notation is that we don't have to mess with spacing and parsing, and cleanup with scripts becomes more robust. The names conform to the ConTEXt way of naming glyphs and the names and reverse mappings are taken from the encoding files in the ConTEXt distribution, so you need to have ConTEXt installed.

The ConTEXt pattern files are generated by a Ruby script. Although the conversion is rather straightforward, some languages need special treatment, but a script is easily adapted. If you want a whole bunch of pattern files, just say:

```
ctxtools --patterns all
```

Or, if you want one language:

```
ctxtools --patterns nl
```

If for some reason this program does not start, try:

```
texmfstart ctxtools --patterns nl
```

When things run well, this will give you four files:

`lang-nl.pat`	the patterns in an encoding independent format
`lang-nl.hyp`	the hyphenation exceptions

`lang-nl.log`	the conversion log (can be deleted afterwards)
`lang-nl.rme`	the preambles of the files used (copyright notices and such)

If you redistribute the files, it makes sense to bundle the `rme` files as well, unless the originals are already in the distribution. It makes no sense to keep the log files on your system. When the file `lang-all.xml` is present, the info from that file will be used and added to the pattern and hyphenation files. In that case no `rme` and `log` file will be generated, unless `--log` is specified.

In the Dutch pattern file you will notice entries like the following:

```
e[ediaeresis]n3
```

So, instead of those funny (encoding specific) `^^fc` or (format specific) `\"e` we use names. Although this looks ConTEXt dependent it is rather easy to map those names back to characters, especially when one takes into account that most languages only have a few of those special characters and we only have to deal with lowercase instances.

The ConTEXt support module `supp-pat.tex` is quite generic and contains only a few lines of code. Actually, most of the code consists of a dedicated XML handler. Loading a pattern meant for EC encoded fonts in a system other than ConTEXt is done as follows:

```
\bgroup
  \input supp-pat

  \lccode"FC="FC
    \definepatterntoken ediaeresis ^^fc
  \lccode"FF="FF
    \definepatterntoken ssharp     ^^ff
  ...

  \enablepatterntokens
  \enablepatternxml

  \input lang-de.pat
  \input lang-de.hyp
\egroup
```

In addition to this one may want to set additional lower and uppercase codes. In ε-TEX these are stored with the language.

Just for completeness we provide the magic command to generate the XML variants:

```
ctxtools --patterns --xml all
```

This will give you files like:

```
<?xml version='1.0' standalone='yes'?>

<!-- some comment -->

<patterns>
... e&ediaeresis;n3 ...
</patterns>
```

This is also accepted as input but for our purpose it's probably best to stick to the normal method. The pattern language is a TEX specific one anyway.

2 Installing languages

Installing a language in ConTEXt should not take too much effort given that the language is supported. Language specific labels are grouped in `lang-*` files, like `lang-ger.tex` for the germanic languages.

Patterns will be loaded from the files in the general TEX distribution unless `lang-nl.pat` is found, in which case ConTEXt assumes that you prefer the ConTEXt patterns. In that case, run

```
ctxtools --patterns all
```

You need to move the files to the ConTEXt base path that you can locate with:

```
textools --find context.tex
```

You can also use `kpsewhich`, but the above method does an extensive search. Of course you can also generate the files on a temporary location. Now it's time to generate the formats:

```
texexec --make --all
```

Since XƎTEX needs patterns in UTF-8 encoding, we provide a switch for achieving that:

```
texexec --make --all --utf8
```

Beware: you need to load patterns for each language and encoding combination you are going to use. You can configure your local `cont-usr` file to take care of this. When an encoding does not have the characters that are needed, you will get an error. When using the non-ConTEXt versions this may go unnoticed because the encoding is hard coded in the file. Of course it will eventually get noticed when the hyphenations come out wrong.

The ConTEXt distribution has a file that holds the copyright and other notes about patterns, named `lang-all.xml`. An example description:

```
<description language='nl'>
  <sourcefile>nehyph96.tex</sourcefile>
  <title>TeX hyphenation patterns for the
         Dutch language</title>
  <copyright>
    <year>1996</year>
    <owner>Piet Tutelaers
       (P.T.H.Tutelaers at tue.nl)</owner>
    <comment>8-bit hyphenation patterns
      for TeX based upon the new Dutch
      spelling, officially since
      1 August 1996. These patterns follow
      the new hyphenation rules in the
      Woordenlijst Nederlandse Taal [...]
    </comment>
  </copyright>
</description>
```

This file is 'work in progress': more details will be added and comments will be enriched.

3 Commands

You can at any moment add additional hyphenation exceptions to the language specific dictionaries. For instance:

```
\language[nl] \hyphenation{pa-tiÃn-ten}
```

Switching to another language is done with the `\language` command. The document language is set with `\mainlanguage`.

If you want to let TEX know that a word should be hyphenated in a special way, you can use the `\-` command, for instance:

```
Con\-TeXt
```

Compound words are not recognized by the hyphenation engine, so there you need to add directives, like:

```
the ConTeXt|-|system
```

If you are using XML as the input format, you need to load the hyphenation filter module. Here we assume that UTF encoding is used:

```
\useXMLfilter[utf,hyp]
```

In your XML file you can now add:

```
<hyphenations language='nl' regime='utf'>
  <hyphenation>pa-tiÃn-ten
    </hyphenation>
  <hyphenation>pa-tiÃn-ten-or-ga-ni-sa-tie
    </hyphenation>
  <hyphenation>pa-tiÃn-ten-plat-form
    </hyphenation>
</hyphenations>
```

This filter also defines some auxiliary elements. Explicit hyphenation points can be inserted as follows:

```
Zullen we hier
af<hyphenate/>bre<hyphenate/>ken of niet?
```

The compound token can be anything, but keep in mind that some tokens are treated specially (see other manuals).

```
Wat is eigenlijk een
patiÃńnten<compound token="-"/>platform?
```

A language is chosen with:

```
nederlands
<language code="en">english</language>
nederlands
```

If you set attribute `scope` to `global`, labels (as used for figure captions and such) adapt to the language switch. This option actually invokes `\mainlanguage`.

4 Languages

When users in a specific language area use more than one font encoding, patterns need to be loaded multiple times. In theory this means that one can end up with more instances than TEX can host. However, the number of sensible font encodings is limited, as is the number of languages that need hyphenation. Now that memory is cheap and machines are fast, preloading a lot of pattern files is no problem.

◇ Hans Hagen
Pragma ADE
`pragma (at) wxs.nl`