The hyphsubst package

Heiko Oberdiek
<oberdiek@uni-freiburg.de>

2008/06/09 v0.2

Abstract

A TEX format file may include alternative hyphenation patterns for a language with a different name. If the naming convention follows babel's rules, then the hyphenation patterns for a language can be replaced by the alternative hyphenation patterns, provided in the format file.

Contents

1 Documentation

1.1 In short

The package is an experimental package that allows the substitution of hyphenation patterns, example:

```
\RequirePackage[ngerman=ngerman-x-20080601]{hyphsubst}
\documentclass{article}
\usepackage[ngerman]{babel}
```

The patterns `ngerman` are replaced by the patterns `ngerman-x-20080601`. The format must contain these patterns and should use the naming scheme of either babel's `language.dat` or `etex.src`'s `language.def`.

1.2 Longer version

Assume the format may contain the following hyphenation patterns (excerpt from `language.dat`):

```
...
ngerman dehyphn.tex
ngerman-x-20071231 dehyphn-x-20071231
ngerman-x-20080601 dehyphn-x-20080601
=ngerman-x-latest % alias for ngerman-x-20080601
...
```

The patterns that contain `-x-` are experimental new patterns for `ngerman`. However, package babel does not provide the use of patterns that do not have the same name as the used language (dialect). The babel system remembers patterns in macros: `\l@`⟨*name*⟩. ε-TeX's `etex.src` uses `\lang@`⟨*name*⟩ instead. In the following we use `babel`'s naming scheme, but `etex.src`'s naming scheme is supported, too.

This package hyphsubst solves the problem by redefining the macro `\l@`⟨*name*⟩ to use other patterns.

`\HyphSubstLet` {⟨*nameA*⟩} {⟨*nameB*⟩}

`\l@`⟨*nameA*⟩ now has the same meaning as `\l@`⟨*nameB*⟩. The patterns for `nameB` must exist. If the patterns for `nameA` exist, then they will be overwritten to use the patterns for `nameB`. Example:

```
\documentclass{article}
\usepackage{hyphsubst}
\HyphSubstLet{ngerman}{ngerman-x-20080601}
\usepackage[ngerman]{babel}
```

Now the patterns `ngerman-x-20080601` are be used.

Or if you want to compare hyphenations:

```
\documentclass{article}
\usepackage{hyphsubst}
  % save original patterns for ngerman in ngerman-saved
\HyphSubstLet{ngerman-saved}{ngerman}
\usepackage[ngerman]{babel}
\begin{document}
  We start with the original patterns for ngerman.
  \HyphSubstLet{ngerman}{ngerman-x-latest}%
  Now we are using ngerman-x-latest.
  \HyphSubstLet{ngerman}{ngerman-saved}%
  Again we are using the original patterns.
\end{document}
```

`\HyphSubstIfExists` {⟨*name*⟩} {⟨*then*⟩} {⟨*else*⟩}

Tests if patterns with name ⟨*name*⟩ exist and execute ⟨*then*⟩ in case of success and ⟨*else*⟩ otherwise.

1.3 LaTeX

The package can also be loaded before `\documentclass`:

```
\RequirePackage[ngerman=ngerman-x-20080601]{hyphsubst}
\documentclass{article}
...
```

This allows to put the package in a format file.

Package options are interpreted as 'let' assignments and passed to macro `\HyphSubstLet`:

```
\usepackage[ngerman=ngerman-x-20080601]{hyphsubst}
```

The part before the equal sign is the first argument for `\HyphSubstLet` and the part after the equal sign forms the second argument:

```
\HyphSubstLet{ngerman}{ngerman-x-20080601}
```

Note, this only works for direct package options. Global options are ignored.

1.4 plain-TEX

The package can be loaded and used with plain-TEX, e.g.:

```
\input hyphsubst.sty
\HyphSubstLet{ngerman}{ngerman-x-latest}
```

2 Implementation

⟨*package⟩

2.1 Reload check and package identification

Reload check, especially if the package is not used with LATEX.

```
\begingroup
  \catcode44 12 % ,
  \catcode45 12 % -
  \catcode46 12 % .
  \catcode58 12 % :
  \catcode64 11 % @
  \catcode123 1 % {
  \catcode125 2 % }
  \expandafter\let\expandafter\x\csname ver@hyphsubst.sty\endcsname
  \ifx\x\relax % plain-TeX, first loading
  \else
    \def\empty{}%
    \ifx\x\empty % LaTeX, first loading,
      % variable is initialized, but \ProvidesPackage not yet seen
    \else
      \catcode35 6 % #
      \expandafter\ifx\csname PackageInfo\endcsname\relax
        \def\x#1#2{%
          \immediate\write-1{Package #1 Info: #2.}%
        }%
      \else
        \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
      \fi
      \x{hyphsubst}{The package is already loaded}%
      \aftergroup\endinput
    \fi
  \fi
\endgroup
```

Package identification:

```
\begingroup
  \catcode35 6 % #
  \catcode40 12 % (
  \catcode41 12 % )
  \catcode44 12 % ,
  \catcode45 12 % -
  \catcode46 12 % .
  \catcode47 12 % /
  \catcode58 12 % :
  \catcode64 11 % @
  \catcode91 12 % [
  \catcode93 12 % ]
  \catcode123 1 % {
  \catcode125 2 % }
  \expandafter\ifx\csname ProvidesPackage\endcsname\relax
    \def\x#1#2#3[#4]{\endgroup
      \immediate\write-1{Package: #3 #4}%
      \xdef#1{#4}%
    }%
  \else
    \def\x#1#2[#3]{\endgroup
      #2[{#3}]%
      \ifx#1\@undefined
        \xdef#1{#3}%
      \fi
      \ifx#1\relax
        \xdef#1{#3}%
      \fi
    }%
  \fi
\expandafter\x\csname ver@hyphsubst.sty\endcsname
\ProvidesPackage{hyphsubst}%
  [2008/06/09 v0.2 Substitute hyphenation patterns (HO)]
```

```
\begingroup
  \catcode123 1 % {
  \catcode125 2 % }
  \def\x{\endgroup
    \expandafter\edef\csname HyphSubst@AtEnd\endcsname{%
      \catcode35 \the\catcode35\relax
      \catcode64 \the\catcode64\relax
      \catcode123 \the\catcode123\relax
      \catcode125 \the\catcode125\relax
    }%
  }%
\x
\catcode35 6 % #
\catcode64 11 % @
\catcode123 1 % {
\catcode125 2 % }
\def\TMP@EnsureCode#1#2{%
  \edef\HyphSubst@AtEnd{%
    \HyphSubst@AtEnd
    \catcode#1 \the\catcode#1\relax
  }%
  \catcode#1 #2\relax
}
\TMP@EnsureCode{39}{12}% '
\TMP@EnsureCode{46}{12}% .
\TMP@EnsureCode{47}{12}% /
\TMP@EnsureCode{58}{12}% :
\TMP@EnsureCode{61}{12}% =
\TMP@EnsureCode{96}{12}% `
```

2.2 Package

```
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname RequirePackage\endcsname\relax
 \input infwarerr.sty\relax
\else
 \RequirePackage{infwarerr}[2007/09/09]%
\fi
```

\HyphSubst@l

```
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname et@xlang\endcsname\relax
 \def\HyphSubst@l{l@}%
\else
 \def\HyphSubst@l{lang@}%
\fi
```

\HyphSubstLet

```
\def\HyphSubstLet#1#2{%
 \begingroup
  \def\x{}%
  \expandafter\ifx\csname\HyphSubst@l#2\endcsname\relax
   \@PackageError{hyphsubst}{Unknown pattern `#2'}\@ehc
  \else
   \def\lmsg{}%
   \expandafter\ifx\csname\HyphSubst@l#1\endcsname\relax
    \edef\msg{%
     New: \expandafter\string\csname\HyphSubst@l#1\endcsname
     \noexpand\MessageBreak
    }%
   \else
    \edef\msg{%
     Redefined: \expandafter\string\csname\HyphSubst@l#1\endcsname
     \noexpand\MessageBreak
     old value: \number\csname\HyphSubst@l#1\endcsname
     \noexpand\MessageBreak
    }%
    \ifnum\csname\HyphSubst@l#1\endcsname=\language
     \edef\x{%
      \noexpand\language=%
        \number\csname\HyphSubst@l#2\endcsname\relax
     }%
     \edef\lmsg{%
      \noexpand\MessageBreak
      \string\language\noexpand\space updated%
     }%
    \fi
   \fi
   \expandafter\global\expandafter\let
     \csname\HyphSubst@l#1\expandafter\endcsname
     \csname\HyphSubst@l#2\endcsname
   \@PackageInfo{hyphsubst}{%
    \msg
    new value: \number\csname\HyphSubst@l#1\endcsname
    \lmsg
   }%
  \fi
 \expandafter\endgroup\x
}
```

\HyphSubstIfExists

```
\def\HyphSubstIfExists#1{%
 \begingroup\expandafter\expandafter\expandafter\endgroup
 \expandafter\ifx\csname\HyphSubst@l#1\endcsname\relax
```

```
    \expandafter\@secondoftwo
  \else
    \expandafter\@firstoftwo
  \fi
}
```

\@firstoftwo

```
\expandafter\ifx\csname @firstoftwo\endcsname\relax
  \long\def\@firstoftwo#1#2{#1}%
\fi
```

\@secondoftwo

```
\expandafter\ifx\csname @secondoftwo\endcsname\relax
  \long\def\@secondoftwo#1#2{#2}%
\fi
```

```
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname documentclass\endcsname\relax
  \HyphSubst@AtEnd
  \expandafter\endinput
\fi
```

```
\DeclareOption*{%
  \expandafter\HyphSubst@Option\CurrentOption==\relax
}
\def\HyphSubst@Option#1=#2=#3\relax{%
  \HyphSubstLet{#1}{#2}%
}
\ProcessOptions*\relax
```

```
\HyphSubst@AtEnd
⟨/package⟩
```

3 Test

3.1 Catcode checks for loading

```
⟨*test1⟩
```

```
\catcode`\{=1 %
\catcode`\}=2 %
\catcode`\#=6 %
\catcode`\@=11 %
\expandafter\ifx\csname count@\endcsname\relax
  \countdef\count@=255 %
\fi
\expandafter\ifx\csname @gobble\endcsname\relax
  \long\def\@gobble#1{}%
\fi
\expandafter\ifx\csname @firstofone\endcsname\relax
  \long\def\@firstofone#1{#1}%
\fi
\expandafter\ifx\csname loop\endcsname\relax
  \expandafter\@firstofone
\else
  \expandafter\@gobble
\fi
{%
  \def\loop#1\repeat{%
    \def\body{#1}%
    \iterate
  }%
  \def\iterate{%
```

```
    \body
      \let\next\iterate
    \else
      \let\next\relax
    \fi
    \next
  }%
  \let\repeat=\fi
}%
\def\RestoreCatcodes{}
\count@=0 %
\loop
  \edef\RestoreCatcodes{%
    \RestoreCatcodes
    \catcode\the\count@=\the\catcode\count@\relax
  }%
\ifnum\count@<255 %
  \advance\count@ 1 %
\repeat

\def\RangeCatcodeInvalid#1#2{%
  \count@=#1\relax
  \loop
    \catcode\count@=15 %
  \ifnum\count@<#2\relax
    \advance\count@ 1 %
  \repeat
}
\expandafter\ifx\csname LoadCommand\endcsname\relax
  \def\LoadCommand{\input hyphsubst.sty\relax}%
\fi
\def\Test{%
  \RangeCatcodeInvalid{0}{47}%
  \RangeCatcodeInvalid{58}{64}%
  \RangeCatcodeInvalid{91}{96}%
  \RangeCatcodeInvalid{123}{255}%
  \catcode`\@=12 %
  \catcode`\\=0 %
  \catcode`\{=1 %
  \catcode`\}=2 %
  \catcode`\#=6 %
  \catcode`\[=12 %
  \catcode`\]=12 %
  \catcode`\%=14 %
  \catcode`\ =10 %
  \catcode13=5 %
  \LoadCommand
  \RestoreCatcodes
}
\Test
\csname @@end\endcsname
\end
```

⟨/test1⟩

3.2 Main tests

⟨*test2⟩

```
\input hyphsubst.sty\relax

\catcode`\@=11\relax
\ifx\et@xlang\@undefined
  \def\l#1{\csname l@#1\endcsname}%
\else
```

```
  \def\l#1{\csname lang@#1\endcsname}%
\fi
\def\Check#1#2{%
  \ifnum#1=#2\relax
  \else
    \@PackageError{test}{Wrong number: #1 <> #2}\@ehc
  \fi
}

\language=0\relax
\HyphSubstLet{ZeroSaved}{ngerman}
\Check{\l{USenglish}}{0}%
\HyphSubstLet{USenglish}{ngerman}
\Check{\l{USenglish}}{\l{ngerman}}
\ifnum\l{USenglish}>0 %
\else
  \@PackageError{test}{\string\language\space is not updated}\@ehc
\fi
\HyphSubstLet{german}{ngerman}
\Check{\l{german}}{\l{ngerman}}
\Check{\l{USenglish}}{\l{ngerman}}
\csname @@end\endcsname\end
⟨/test2⟩
```

4 Installation

4.1 Download

Package. This package is available on CTAN[1]:

CTAN:macros/latex/contrib/oberdiek/hyphsubst.dtx The source file.

CTAN:macros/latex/contrib/oberdiek/hyphsubst.pdf Documentation.

Bundle. All the packages of the bundle 'oberdiek' are also available in a TDS compliant ZIP archive. There the packages are already unpacked and the documentation files are generated. The files and directories obey the TDS standard.

CTAN:install/macros/latex/contrib/oberdiek.tds.zip

TDS refers to the standard "A Directory Structure for TEX Files" (CTAN:tds/tds.pdf). Directories with `texmf` in their name are usually organized this way.

4.2 Bundle installation

Unpacking. Unpack the `oberdiek.tds.zip` in the TDS tree (also known as `texmf` tree) of your choice. Example (linux):

```
unzip oberdiek.tds.zip -d ~/texmf
```

Script installation. Check the directory `TDS:scripts/oberdiek/` for scripts that need further installation steps. Package attachfile2 comes with the Perl script `pdfatfi.pl` that should be installed in such a way that it can be called as `pdfatfi`. Example (linux):

```
chmod +x scripts/oberdiek/pdfatfi.pl
cp scripts/oberdiek/pdfatfi.pl /usr/local/bin/
```

[1] ftp://ftp.ctan.org/tex-archive/

4.3 Package installation

Unpacking. The `.dtx` file is a self-extracting docstrip archive. The files are extracted by running the `.dtx` through plain-TEX:

```
tex hyphsubst.dtx
```

TDS. Now the different files must be moved into the different directories in your installation TDS tree (also known as `texmf` tree):

```
hyphsubst.sty             → tex/generic/oberdiek/hyphsubst.sty
hyphsubst.pdf             → doc/latex/oberdiek/hyphsubst.pdf
test/hyphsubst-test1.tex → doc/latex/oberdiek/test/hyphsubst-test1.tex
test/hyphsubst-test2.tex → doc/latex/oberdiek/test/hyphsubst-test2.tex
hyphsubst.dtx             → source/latex/oberdiek/hyphsubst.dtx
```

If you have a `docstrip.cfg` that configures and enables docstrip's TDS installing feature, then some files can already be in the right place, see the documentation of docstrip.

4.4 Refresh file name databases

If your TEX distribution (teTEX, mikTEX, . . .) relies on file name databases, you must refresh these. For example, teTEX users run `texhash` or `mktexlsr`.

4.5 Some details for the interested

Attached source. The PDF documentation on CTAN also includes the `.dtx` source file. It can be extracted by AcrobatReader 6 or higher. Another option is pdftk, e.g. unpack the file into the current directory:

```
pdftk hyphsubst.pdf unpack_files output .
```

Unpacking with LATEX. The `.dtx` chooses its action depending on the format:

plain-TEX: Run docstrip and extract the files.

LATEX: Generate the documentation.

If you insist on using LATEX for docstrip (really, docstrip does not need LATEX), then inform the autodetect routine about your intention:

```
latex \let\install=y\input{hyphsubst.dtx}
```

Do not forget to quote the argument according to the demands of your shell.

Generating the documentation. You can use both the `.dtx` or the `.drv` to generate the documentation. The process can be configured by the configuration file `ltxdoc.cfg`. For instance, put this line into this file, if you want to have A4 as paper format:

```
\PassOptionsToClass{a4paper}{article}
```

An example follows how to generate the documentation with pdfLATEX:

```
pdflatex hyphsubst.dtx
makeindex -s gind.ist hyphsubst.idx
pdflatex hyphsubst.dtx
makeindex -s gind.ist hyphsubst.idx
pdflatex hyphsubst.dtx
```

5 History

[2008/06/07 v0.1]

- First public version.

[2008/06/09 v0.2]

- Support for ε-TEX's `language.def` added.
- Fix for undefined `\lmsg`.

6 Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.