The iflang package
Heiko Oberdiek*
2018/01/21 v1.7

Abstract
This package provides expandible checks for the current language based on macro \languagename or hyphenation patterns.

Contents
1 Documentation 2
2 Implementation 2
   2.1 Reload check and package identification 3
   2.2 Tools 4
       2.2.1 Provide some basic macros of \LaTeX 4
       2.2.2 Expandible existence check for macros 5
       2.2.3 Macros for messages 5
       2.2.4 Support for etex.src 5
   2.3 \IfLanguagePatterns 6
   2.4 \IfLanguageName 6
   2.5 Check plausibility of \languagename 8
3 Installation 8
   3.1 Download 8
   3.2 Bundle installation 8
   3.3 Package installation 8
   3.4 Refresh file name databases 9
   3.5 Some details for the interested 9
4 Acknowledgement 9
5 History 9
   [2007/04/10 v1.0] 9
   [2007/04/11 v1.1] 9
   [2007/04/12 v1.2] 10
   [2007/04/26 v1.3] 10
   [2007/11/11 v1.5] 10
   [2016/05/16 v1.6] 10
   [2018/01/21 v1.7] 10
6 Index 10

*Please report any issues at https://github.com/ho-tex/oberdiek/issues
1 Documentation

Package babel defines \iflanguage. As first argument it takes a language name and executes the second or third argument depending on the current language. This language test is based on hyphenation patterns. However, it is possible that different languages or dialects share the same patterns. In such cases \iflanguage fails.

However, package babel and some other packages such as german or ngerman store the language name in the macro \language if \selectlanguage is called.

\IfLanguageName{(lang)}{(then)}{(else)}

Makro \IfLanguageName compares language \langle lang \rangle with the current setting of macro \language. If both contains the same name then the \langle then \rangle part is called, otherwise the \langle else \rangle part.

The macro is expandable. Thus it can be safely used inside \edef or \csname. If case of errors like an undefined \language the \langle else \rangle part is executed.

Note: Macro \IfLanguageName relies on the fact, that \language is set correctly:

Package babel:
- Full support of \language in its language switching commands.

Format based on babel (language.dat):
- If package babel is not used (or not yet loaded), then babel's hyphen.cfg has set \language to the last language in language.dat, but \language (current patterns) is zero and points to the first language. Thus the value of \language is basically garbage. Package iflang warns if \language and \language do not fit. This can be fixed by loading package babel previously.

Format based on ε-Tex's etex.src (language.def):
- Unhappily it does not support \language. Thus this package hooks into \uselanguage to get \language defined and updated there. At package loading time the changed \uselanguage has not been called yet. Thus package iflang tries USenglish. This is the definite default language of etex.src. If the current patterns suit this default language, an undefined \language remains undefined and a warning is given.

\IfLanguagePatterns{(lang)}{(then)}{(else)}

This macro behaves similar to \IfLanguageName. But the language test is based on the current pattern in force (\language). Also this macro is expandable, in case of errors the \langle else \rangle part is called.

The following naming convention for the pattern are supported:

\texttt{babel/language.dat : \l@\langle language \rangle}
\texttt{etex/src/language.def : \lang@\langle language \rangle}

Package iflang looks for \et@xp@t@xpatterns (defined in etex.src) to find out the naming convention in use.

2 Implementation
2.1 Reload check and package identification

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

\begin{verbatim}
\begingroup\catcode61\catcode48\catcode32=10\relax%
\catcode13=5 % ^^M
\endlinechar=13 %
\catcode35=6 % #
\catcode39=12 % ,
\catcode44=12 % ,
\catcode45=12 % -
\catcode46=12 % .
\catcode58=12 % :
\catcode64=11 % @
\catcode123=1 % {
\catcode125=2 % }
\endlinechar=13 %
\catcode35=6 % #
\catcode39=12 % ,
\catcode44=12 % ,
\catcode46=12 % .
\catcode47=12 % /
\catcode58=12 % :
\catcode64=11 % @
\catcode123=1 % {
\catcode125=2 % }
\endlinechar=13 %
\endgroup
\end{verbatim}

Package identification:

\begin{verbatim}
\begingroup\catcode61\catcode48\catcode32=10\relax%
\catcode13=5 % ^^M
\endlinechar=13 %
\catcode35=6 % #
\catcode39=12 % ,
\catcode44=12 % ,
\catcode45=12 % -
\catcode46=12 % .
\catcode47=12 % /
\catcode58=12 % :
\catcode64=11 % @
\catcode123=1 % {
\catcode125=2 % }
\endlinechar=13 %
\endgroup
\end{verbatim}

\makeatletter
\begin{verbatim}
\expandafter\expandafter\expandafter\edef\expandafter\@tempa
\expandafter\expandafter\expandafter{%}
\expandafter\edef\expandafter\@tempb
\expandafter\expandafter\expandafter{%}
\end{verbatim}

\makeatother

\@firstoftwo
\@secondoftwo
2.2.2 Expandible existence check for macros

\IfLang@ifDefined

2.2.3 Macros for messages

\IfLang@prefix

2.2.4 Support for etex.src

\IfLang@prefix
The first `\uselanguage` that is executed as last line in `language.def` cannot patched this way. However, `language.def` is very strict. It forces the first added and used language to be `USenglish`. Thus, if `\languagename` is not defined, we can quite safely assume `USenglish`. As additional safety precaution the actual used patterns are checked.

```latex
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname languagename\endcsname\relax
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname lang@USenglish\endcsname\relax
\@PackageWarningNoLine@iflang}{%\string\lang@USenglish\space is missing%}
\else
\@PackageWarningNoLine@iflang}{%\string\languagename\space is not set,\MessageBreak current language is unknown%}
\fi
\fi
\fi
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname languagename\endcsname\relax
\@PackageInfoNoLine@iflang}{%\string\languagename\space is not set%}
\fi
```

### 2.3 `\IfLanguagePatterns`

```latex
\def\IfLanguagePatterns#1{%\IfLang@prefix\endcsname=\language
\def\IfLanguagePatterns#1{%\IfLang@prefix\endcsname=\language
\ifnum\IfLang@IfDefined{\IfLang@prefix\endcsname}{%\string\languagename\space is not set,\MessageBreak current language is unknown%}
\else
\@PackageWarningNoLine@iflang}{%\string\languagename\space is not set%}
\fi
\fi
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname pdf@strcmp\endcsname\relax
\@firstoftwo
\else
\@secondoftwo
\fi
\expandafter\@firstoftwo
\fi
```

### 2.4 `\IfLanguageName`

```latex
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname pdf@strcmp\endcsname\relax
\@firstoftwo
\else
\@secondoftwo
\fi
\expandafter\@firstoftwo
\fi
```
We do not have \pdfstrcmp (and \pdfstrcmpp). Thus we must define our own expandable string comparison. The following implementation is based on a \TeX pearl from David Kastrup, presented at the conference Bacho\TeX\ 2005: \url{http://www.gust.org.pl/projects/pearls/2005p/david-kastrup/bachotex2005-david-kastrup-pearl1.pdf}

The original code allows macros inside the second string. Because also \texttt{languagename} might consists of further macros, we need a variant that allows macros in the first string, too.

\begin{verbatim}
\def\IfLang@StrNil{\relax}%
\def\IfLang@StrEqual#1{%
  \number\IfLang@StrEqualStart{}{}#1\IfLang@StrNil
}
\def\IfLang@StrEqualStart#1#2#3{%
  \ifx#3\IfLang@StrNil
    \IfLang@StrEqualStop
  \fi
  \ifcat\noexpand#3\relax
    \IfLang@StrExpand{#1}{#2}#3%
  \fi
  \IfLang@StrEqualStart{\if#3#1}{#2\fi}%
}%
\def\IfLang@StrEqualStop\fi#1\IfLang@StrEqualStart#2#3#4{%
  \fi
  #2#4\relax'#313%
}%
\def\IfLang@StrExpand#1#2#3\fi\IfLang@StrEqualStart#4#5{%
  \fi
  \IfLang@@StrExpand{#1}{#2}#3%
}%
\def\IfLang@@StrExpand#1#2#3\IfLang@StrNil{%
  \expandafter\IfLang@@@StrExpand#3\IfLang@StrNil{#1}{#2}%
}%
\def\IfLang@@@StrExpand#1\IfLang@StrNil#2#3{%
  \IfLang@StrEqualStart{#2}{#3}#1\IfLang@StrNil
}%
\end{verbatim}

\texttt{\IfLanguageName}

\begin{verbatim}
\def\IfLanguageName#1{%
  \ifnum\IfLang@IfDefined{languagename}{%
    \if\expandafter\IfLang@StrEqual\expandafter%
      {\languagename}{#1}%
    0%
  \else
    1%
  \fi
  \expandafter\@firstoftwo
}%
\end{verbatim}

\texttt{\IfLanguageName}
2.5 Check plausibility of \languagename

\begingroup \expandafter \expandafter \expandafter \endgroup
\expandafter \ifx \csname \languagename \endcsname \relax
\else
\IfLanguagePatterns{\languagename}{%}
\PackageWarningNoLine{iflang}{% Mismatch between \string\languagename\space (patterns)\MessageBreak and setting of \string\languagename
}%
\fi
\IfLangAtEnd
⟨/package⟩

3 Installation

3.1 Download

Package. This package is available on CTAN:\textsuperscript{1}:  
\url{CTAN:macros/latex/contrib/oberdiek/iflang.dtx} The source file.  
\url{CTAN:macros/latex/contrib/oberdiek/iflang.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.  
\url{CTAN:install/macros/latex/contrib/oberdiek.tds.zip}  

TDS refers to the standard “A Directory Structure for \TeX\ Files” (\url{CTAN:pkg/tds}). Directories with texmf in their name are usually organized this way.

3.2 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

3.3 Package installation

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

tex iflang.dtx

\textsuperscript{1}CTAN:pkg/iflang
TDS. Now the different files must be moved into the different directories in your installation TDS tree (also known as texmf tree):

- `iflang.sty` → `tex/generic/oberdiek/iflang.sty`
- `iflang.pdf` → `doc/latex/oberdiek/iflang.pdf`
- `iflang.dtx` → `source/latex/oberdiek/iflang.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`.

3.4 Refresh file name databases

If your \TeX\ distribution (\TeX\ Live, MiK\TeX\, …) relies on file name databases, you must refresh these. For example, \TeX\ Live users run `texhash` or `mktexlsr`.

3.5 Some details for the interested

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{iflang.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 pd\LaTeX:\

```bash
pdflatex iflang.dtx
makeindex -s gind.ist iflang.idx
pdflatex iflang.dtx
makeindex -s gind.ist iflang.idx
pdflatex iflang.dtx
```

4 Acknowledgement

I wish to thank:

Markus Kohm Useful hints for version 1.2.

5 History

[2007/04/10 v1.0]
- First public version.

[2007/04/11 v1.1]
- Line ends sanitized.
[2007/04/12 v1.2]
• Initialization of \languagename in case of etex.src.
• Some sanity tests added.
• Documentation improved.

[2007/04/26 v1.3]
• Use of package infwarerr.

[2007/09/09 v1.4]
• Bug fix: \IfLang@StrEqual \rightarrow \IfLangStrEqual (Gabriele Balducci).
• Catcode section rewritten.

[2007/11/11 v1.5]
• Use of package pdftexcmds for LuaTEX support.

[2016/05/16 v1.6]
• Documentation updates.

[2018/01/21 v1.7]
• Fix test for etex.src.

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; plain numbers refer to the code lines where the entry is used.

Symbols
\@PackageInfoNoLine ... 159, 164, 194
\@PackageWarningNoLine 177, 184, 274
\@firstoftwo ................ 111, 119, 128, 143, 206, 213, 254, 264
\@secondoftwo ............. 114, 121, 126, 145, 208, 215, 256, 266
\@undefined ................ 58
\aftergroup .................. 29
\catcode 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 69, 70, 72, 73, 74, 78, 79, 80, 81, 82, 83, 84, 87, 88, 90, 91, 92, 93, 97, 99
\csname ...................... 14, 21, 50, 66, 76, 111, 114, 118, 125, 134, 150, 158, 174, 176, 193, 200, 212, 271
\endcsname ................. 14, 21, 50, 66, 76, 111, 114, 118, 125, 134, 150, 158, 174, 176, 193, 200, 212, 271
\endinput .................. 4, 33, 71, 77, 89
\endlinechar ............... 29, 110
\IfLang@@@StrExpand ........ 240, 242
\IfLang@@StrExpand ......... 237, 239
\IfLang@AtEnd ............... 95, 96, 110, 281
\IfLang@IfDefined ........... 117, 199, 246, 261
\IfLang@OrgUseLanguage ...... 168, 171
\IfLang@prefix .............. 157, 199, 200
\IfLang@StrEqual ........... 219, 247
\IfLang@StrEqualStart ...... 220, 222, 229, 231, 235, 243
\IfLang@StrEqualStop ...... 220, 222, 229, 231, 235, 243
\IfLang@StrExpand ........... 227, 235
\IfLang@StrNil ............. 218, 220, 223, 239, 240, 242, 243
\IfLanguageName ............ 2, 245, 260
\IfLanguagePatterns ....... 2, 198, 273