

The iflang package

Heiko Oberdiek
<oberdiek@uni-freiburg.de>

2007/04/26 v1.3

Abstract

This package provides expandible checks for the current language based on macro `\language` 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 L ^A T _E X	4
2.2.2	Expandible existence check for macros	4
2.2.3	Macros for messages	4
2.2.4	Support for <code>etex.src</code>	5
2.3	<code>\IfLanguagePatterns</code>	5
2.4	<code>\IfLanguageName</code>	6
2.5	Check plausibility of <code>\language</code>	7
3	Test	7
3.1	Test with L ^A T _E X	7
3.2	Test with plain-T _E X and ε -T _E X	8
3.3	Test with plain-T _E X and without ε -T _E X/pdfT _E X	9
4	Installation	10
4.1	Download	10
4.2	Bundle installation	10
4.3	Package installation	10
4.4	Refresh file name databases	11
4.5	Some details for the interested	11
5	Acknowledgement	11
6	History	11
	[2007/04/10 v1.0]	11
	[2007/04/11 v1.1]	11
	[2007/04/12 v1.2]	11
	[2007/04/26 v1.3]	12
7	Index	12

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 `<lang>` with the current setting of macro `\language`. If both contains the same name then the `<then>` part is called, otherwise the `<else>` part.

The macro is expandable. Thus it can be safely used inside `\edef` or `\csname`. If case of errors like an undefined `\language` the `<else>` 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` is set to this language. Otherwise a `\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 `<else>` part is called.

The following naming convention for the pattern are supported:

babel/`language.dat` : `\l@<language>`

etex.src/`language.def` : `\lang@<language>`

Package **iflang** looks for `\uselanguage` (defined in `etex.src`) to find out the naming convention in use.

2 Implementation

1 `*package`

2.1 Reload check and package identification

Reload check, especially if the package is not used with L^AT_EX.

```
2 \begingroup
3   \catcode44 12 % ,
4   \catcode45 12 % -
5   \catcode46 12 % .
6   \catcode58 12 % :
7   \catcode64 11 % @
8   \expandafter\let\expandafter\x\csname ver@iflang.sty\endcsname
9   \ifcase 0%
10    \ifx\x\relax % plain
11    \else
12      \ifx\x\empty % LaTeX
13      \else
14        1%
15      \fi
16    \fi
17  \else
18    \expandafter\ifx\csname PackageInfo\endcsname\relax
19      \def\x#1#2{%
20        \immediate\write-1{Package #1 Info: #2.}%
21      }%
22    \else
23      \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
24    \fi
25    \x@iflang}{The package is already loaded}%
26  \endgroup
27  \expandafter\endinput
28  \fi
29 \endgroup
```

Package identification:

```
30 \begingroup
31   \catcode40 12 % (
32   \catcode41 12 % )
33   \catcode44 12 % ,
34   \catcode45 12 % -
35   \catcode46 12 % .
36   \catcode47 12 % /
37   \catcode58 12 % :
38   \catcode64 11 % @
39   \expandafter\ifx\csname ProvidesPackage\endcsname\relax
40     \def\x#1#2#3[#4]{\endgroup
41       \immediate\write-1{Package: #3 #4}%
42       \xdef#1{#4}%
43     }%
44   \else
45     \def\x#1#2[#3]{\endgroup
46       #2[#{#3}]%
47       \ifx#1\relax
48         \xdef#1{#3}%
49       \fi
50     }%
51   \fi
52   \expandafter\x\csname ver@iflang.sty\endcsname
53   \ProvidesPackage{iflang}%
54   [2007/04/26 v1.3 Language checks (HO)]
55   \expandafter\edef\csname IfLang@endinput\endcsname{%
56     \catcode39 \the\catcode39\relax % '
57     \catcode40 \the\catcode40\relax % (
58     \catcode41 \the\catcode41\relax % )
59     \catcode61 \the\catcode61\relax % =
```

```

60 \catcode64 \the\catcode64\relax % @
61 \noexpand\endinput
62 }
63 \catcode39 12\relax % '
64 \catcode40 12\relax % (
65 \catcode41 12\relax % )
66 \catcode61 12\relax % =
67 \catcode64 11\relax % @

```

2.2 Tools

2.2.1 Provide some basic macros of \LaTeX

\backslash @firstoftwo

```

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

```

\backslash @secondoftwo

```

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

```

2.2.2 Expandible existence check for macros

\backslash IfLang@ifDefined

```

74 \begingroup\expandafter\expandafter\expandafter\endgroup
75 \expandafter\ifx\csname ifcename\endcsname\relax
76 \expandafter\@firstoftwo
77 \else
78 \expandafter\@secondoftwo
79 \fi
80 {%
81 \def\IfLang@ifDefined#1{%
82 \expandafter\ifx\csname#1\endcsname\relax
83 \expandafter\@secondoftwo
84 \else
85 \expandafter\@firstoftwo
86 \fi
87 }%
88 }{%
89 \def\IfLang@ifDefined#1{%
90 \ifnum\ifcename#1\endcsname
91 \expandafter\ifx\csname#1\endcsname\relax
92 1%
93 \else
94 0%
95 \fi
96 \else
97 1%
98 \fi
99 =0 %
100 \expandafter\@firstoftwo
101 \else
102 \expandafter\@secondoftwo
103 \fi
104 }%
105 }

```

2.2.3 Macros for messages

```

106 \begingroup\expandafter\expandafter\expandafter\endgroup

```

```

107 \expandafter\ifx\csname RequirePackage\endcsname\relax
108   \input infwarerr.sty\relax
109 \else
110   \RequirePackage{infwarerr}%
111 \fi

```

2.2.4 Support for etex.src

\IfLang@prefix

```

112 \begingroup\expandafter\expandafter\expandafter\endgroup
113 \expandafter\ifx\csname uselanguage\endcsname\relax
114   \@PackageInfoNoLine{iflang}{%
115     Naming convention for patterns: babel%
116   }%
117   \def\IfLang@prefix{l@}%
118 \else
119   \@PackageInfoNoLine{iflang}{%
120     Naming convention for patterns: etex.src%
121   }%
122   \def\IfLang@prefix{lang@}%
123   \let\IfLang@OrgUseLanguage\uselanguage
124   \def\uselanguage#1{%
125     \edef\language{#1}%
126     \IfLang@OrgUseLanguage{#1}%
127   }%

```

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

```

128 \begingroup\expandafter\expandafter\expandafter\endgroup
129 \expandafter\ifx\csname language\endcsname\relax
130 \begingroup\expandafter\expandafter\expandafter\endgroup
131 \expandafter\ifx\csname lang@USenglish\endcsname\relax
132   \@PackageWarningNoLine{iflang}{%
133     \string\lang@USenglish\space is missing%
134   }%
135 \else
136   \ifnum\lang@USenglish=\language
137     \def\language{USenglish}%
138   \else
139     \@PackageWarningNoLine{iflang}{%
140       \string\language\space is not set,\MessageBreak
141       current language is unknown%
142     }%
143   \fi
144 \fi
145 \fi
146 \fi
147 \begingroup\expandafter\expandafter\expandafter\endgroup
148 \expandafter\ifx\csname language\endcsname\relax
149   \@PackageInfoNoLine{iflang}{%
150     \string\language\space is not set%
151   }%
152 \fi

```

2.3 \IfLanguagePatterns

\IfLanguagePatterns

```

153 \def\IfLanguagePatterns#1{%
154   \ifnum\IfLang@IfDefined{\IfLang@prefix#1}{%

```

```

155         \ifnum\csname\IfLang@prefix#1\endcsname=\language
156         0%
157         \else
158         1%
159         \fi
160     }\{1}=0 %
161     \expandafter\@firstoftwo
162 \else
163     \expandafter\@secondoftwo
164 \fi
165 }

```

2.4 \IfLanguageName

```

166 \begingroup\expandafter\expandafter\expandafter\endgroup
167 \expandafter\ifx\csname pdfstrcmp\endcsname\relax
168     \expandafter\@firstoftwo
169 \else
170     \expandafter\@secondoftwo
171 \fi
172 {%

```

We do not have `\pdfstrcmp`. 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 BachTeX 2005: <http://www-stary.gust.org.pl/pearls/2005/david-kastrup/bachotex2005-david-kastrup-pearl1.pdf>

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

```

173 \def\IfLang@StrNil{\relax}%
174 \def\IfLang@StrEqual#1{%
175     \number\IfLang@StrEqualStart{}\{#1\IfLang@StrNil
176 }%
177 \def\IfLang@StrEqualStart#1#2#3{%
178     \ifx#3\IfLang@StrNil
179         \IfLang@StrEqualStop
180     \fi
181     \ifcat\noexpand#3\relax
182         \IfLang@StrExpand{#1}{#2}#3%
183     \fi
184     \IfLang@StrEqualStart{\if#3#1}{#2\fi}%
185 }%
186 \def\IfLang@StrEqualStop\fi#1\IfLang@StrEqualStart#2#3#4{%
187     \fi
188     #2#4\relax'#313 %
189 }%
190 \def\IfLang@StrExpand#1#2#3\fi\IfLang@StrEqualStart#4#5{%
191     \fi
192     \IfLang@@StrExpand{#1}{#2}#3%
193 }%
194 \def\IfLang@@StrExpand#1#2#3\IfLang@StrNil{%
195     \expandafter\IfLang@@@StrExpand#3\IfLang@StrNil{#1}{#2}%
196 }%
197 \def\IfLang@@@StrExpand#1\IfLang@StrNil#2#3{%
198     \IfLang@StrEqualStart{#2}{#3}#1\IfLang@StrNil
199 }%

```

\IfLanguageName

```

200 \def\IfLanguageName#1{%
201     \ifnum\IfLang@ifDefined{language}\relax
202         \if\expandafter\IfLang@strequal\expandafter%
203             {\language}{#1}%
204         0%

```

```

205         \else
206         1%
207         \fi
208     }{1}=0 %
209     \expandafter\@firstoftwo
210 \else
211     \expandafter\@secondoftwo
212 \fi
213 }%
214 }{%

```

`\IfLanguageName`

```

215 \def\IfLanguageName#1{%
216     \ifnum\IfLang@ifDefined{language#1}%
217         \pdfstrcmp{#1}{\language}%
218     }{1}=0 %
219     \expandafter\@firstoftwo
220 \else
221     \expandafter\@secondoftwo
222 \fi
223 }%
224 }

```

2.5 Check plausibility of `\language`

```

225 \begingroup\expandafter\expandafter\expandafter\endgroup
226 \expandafter\ifx\csname language\endcsname\relax
227 \else
228     \IfLanguagePatterns{\language}{}{%
229         \@PackageWarningNoLine{iflang}{%
230             Mismatch between \string\language\space
231             (patterns)\MessageBreak
232             and setting of \string\language
233         }%
234     }%
235 \fi
236 \IfLang@endinput
237 \</package>

```

3 Test

3.1 Test with L^AT_EX

```

238 \test1\
239 \NeedsTeXFormat{LaTeX2e}
240 \nofiles
241 \documentclass{minimal}
242 \usepackage{qstest}
243 \IncludeTests{*}
244 \LogTests{log}{*}{*}
245 \usepackage[english,naustrian,ngerman]{babel}
246 \usepackage{iflang}
247 \begin{document}
248 \begin{qstest}\IfLanguagePatterns{language, pattern}
249     \def\test#1#2{%
250         \Expect*{\IfLanguagePatterns{#1}{true}{false}}{#2}%
251     }%
252     \test{ngerman}{true}%
253     \test{naustrian}{true}%
254     \test{english}{false}%

```

```

255 \test{foobar}{false}%
256 \end{qstest}
257 \begin{qstest}{IfLanguageName}{language, name}
258 \def\test#1#2{%
259 \Expect*{\IfLanguageName{#1}{true}{false}}{#2}%
260 }%
261 \test{ngerman}{true}%
262 \test{naustrian}{false}%
263 \selectlanguage{naustrian}%
264 \test{ngerman}{false}%
265 \test{naustrian}{true}%
266 \test{foobar}{false}%
267 \end{qstest}
268 \begin{qstest}{IfDefined}{defined}
269 \makeatletter
270 \let\foobar\relax
271 \Expect*{\IfLang@IfDefined{foobar}{true}{false}}{false}%
272 \Expect*{\ifx\foobar\relax true\else false\fi}{true}%
273 \let\foobar\UNDEFINED
274 \Expect*{\IfLang@IfDefined{foobar}{true}{false}}{false}%
275 \Expect*{\ifx\foobar\relax true\else false\fi}{false}%
276 \Expect*{\ifx\foobar\UNDEFINED true\else false\fi}{true}%
277 \end{qstest}
278 \end{document}
279 </test1>

```

3.2 Test with plain-TeX and ε -TeX

```

280 (*test2)
281 %% Format 'etex' based on 'language.def'
282 \input iflang.sty
283 \catcode64=12
284
285 \def\TestGeneric#1#2#3{%
286 \begingroup
287 \edef\x{#1{#2}{true}{false}}%
288 \edef\y{#3}%
289 \ifx\x\y
290 \else
291 \errmessage{Failed test: \string#1{#2} <> #3}%
292 \fi
293 \endgroup
294 }
295 \def\TestPatterns{\TestGeneric\IfLanguagePatterns}
296 \def\TestName{\TestGeneric\IfLanguageName}
297
298 \TestPatterns{USenglish}{true}
299 \TestPatterns{ngerman}{false}
300
301 \TestName{USenglish}{true}
302 \TestName{ngerman}{false}
303
304 \uselanguage{ngerman}
305
306 \TestPatterns{USenglish}{false}
307 \TestPatterns{ngerman}{true}
308
309 \TestName{USenglish}{false}
310 \TestName{ngerman}{true}
311
312 \csname @@end\endcsname
313 \end

```


314 </test2>

3.3 Test with plain-TeX and without ϵ -TeX/pdfTeX

315 <*test3>

```
316 %% Format 'tex' (vanilla plain-TeX)
317 \let\ifcsname\UNDEFINED
318 \let\pdfstrcmp\UNDEFINED
319 \input iflang.sty
320 \catcode64=11
321
322 \def\TestDefined#1{%
323   \IfLang@IfDefined{foobar}{-}{-}%
324   \ifx\foobar#1%
325   \else
326     \errmessage{Failed test: \string\foobar <> \string#1}%
327   \fi
328 }
329 \let\foobar\relax
330 \TestDefined\relax
331 \let\foobar\UNDEFINED
332 \TestDefined\relax
333
334 \def\strip@prefix#1>{}
335 \def@onelevel@sanitize#1{%
336   \edef#1{\expandafter\strip@prefix\meaning#1}%
337 }
338 \def\TestCompare#1#2#3{%
339   \begingroup
340     \edef\x{%
341       \if\IfLang@StrEqual{#1}{#2}%
342       true%
343       \else
344       false%
345       \fi
346     }%
347   \def\expect{#3}%
348   \ifx\x\expect
349   \else
350     \def\a{#1}%
351     \@onelevel@sanitize\a
352     \def\b{#2}%
353     \@onelevel@sanitize\b
354     \errmessage{Failed test: '\a'='\b' <> \expect}%
355   \fi
356   \endgroup
357 }
358 \TestCompare{junk}{junk}{true}
359 \TestCompare{}{}{true}
360 \TestCompare{a}{b}{false}
361 \TestCompare{aa}{bb}{false}
362 \def\a{ax}
363 \def\b{bx}
364 \def\c{\a\b}
365 \def\d{\c\b}
366 \def\exch#1#2{#2#1}
367 \def\gobble#1{}
368 \TestCompare{\gobble a}{}{true}
369 \TestCompare{}{\gobble a}{true}
370 \TestCompare{a\exch xyb}{ayxb}{true}
371 \TestCompare{\c}{\c}{true}
372 \TestCompare{\d}{\c\b}{true}
373
```

```

374 \csname @@end\endcsname
375 \end
376 </test3>

```

4 Installation

4.1 Download

Package. This package is available on CTAN¹:

[CTAN:macros/latex/contrib/oberdiek/iflang.dtx](#) The source file.

[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.

[CTAN:macros/latex/contrib/oberdiek/oberdiek-tds.zip](#)

TDS refers to the standard “A Directory Structure for T_EX 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/

```

4.3 Package installation

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

```
tex iflang.dtx
```

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-test1.tex → doc/latex/oberdiek/iflang-test1.tex
iflang-test2.tex → doc/latex/oberdiek/iflang-test2.tex
iflang-test3.tex → doc/latex/oberdiek/iflang-test3.tex
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`.

¹<ftp://ftp.ctan.org/tex-archive/>

4.4 Refresh file name databases

If your \TeX distribution ($\text{te}\text{\TeX}$, $\text{mik}\text{\TeX}$, ...) relies on file name databases, you must refresh these. For example, $\text{te}\text{\TeX}$ users run `texhash` or `mktextlsr`.

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 iflang.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{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 `pdf \LaTeX` :

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

5 Acknowledgement

I wish to thank:

Markus Kohm Useful hints for version 1.2.

6 History

[2007/04/10 v1.0]

- First public version.

[2007/04/11 v1.1]

- Line ends sanitized.

[2007/04/12 v1.2]

- Initialization of `\language` in case of `etex.src`.
- Some sanity tests added.
- Documentation improved.

- Use of package infwarerr.

7 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.

Symbols	
<code>\@PackageInfoNoLine</code> ...	114, 119, 149
<code>\@PackageWarningNoLine</code>	132, 139, 229
<code>\@firstoftwo</code>	68, 76, 85, 100, 161, 168, 209, 219
<code>\@onelevel@sanitize</code> ...	335, 351, 353
<code>\@secondoftwo</code>	71, 78, 83, 102, 163, 170, 211, 221
A	
<code>\a</code>	350, 351, 354, 362, 364
B	
<code>\b</code> ...	352, 353, 354, 363, 364, 365, 372
<code>\begin</code>	247, 248, 257, 268
C	
<code>\c</code>	364, 365, 371, 372
<code>\catcode</code>	3, 4, 5, 6, 7, 31, 32, 33, 34, 35, 36, 37, 38, 56, 57, 58, 59, 60, 63, 64, 65, 66, 67, 283, 320
<code>\csname</code>	8, 18, 39, 52, 55, 68, 71, 75, 82, 91, 107, 113, 129, 131, 148, 155, 167, 226, 312, 374
D	
<code>\d</code>	365, 372
<code>\documentclass</code>	241
E	
<code>\empty</code>	12
<code>\end</code>	256, 267, 277, 278, 313, 375
<code>\endcsname</code> ...	8, 18, 39, 52, 55, 68, 71, 75, 82, 90, 91, 107, 113, 129, 131, 148, 155, 167, 226, 312, 374
<code>\endinput</code>	27, 61
<code>\errmessage</code>	291, 326, 354
<code>\exch</code>	366, 370
<code>\Expect</code>	250, 259, 271, 272, 274, 275, 276
<code>\expect</code>	347, 348, 354
F	
<code>\foobar</code>	270, 272, 273, 275, 276, 324, 326, 329, 331
G	
<code>\gobble</code>	367, 368, 369
I	
<code>\if</code>	184, 202, 341
<code>\ifcase</code>	9
<code>\ifcat</code>	181
<code>\ifcsname</code>	90, 317
<code>\IfLang@@@StrExpand</code>	195, 197
<code>\IfLang@@StrExpand</code>	192, 194
<code>\IfLang@endinput</code>	236
<code>\IfLang@ifDefined</code>	74, 154, 201, 216, 271, 274, 323
<code>\IfLang@OrgUseLanguage</code>	123, 126
<code>\IfLang@prefix</code>	112, 154, 155
<code>\IfLang@StrEqual</code>	174, 341
<code>\IfLang@strequal</code>	202
<code>\IfLang@StrEqualStart</code>	175, 177, 184, 186, 190, 198
<code>\IfLang@StrEqualStop</code>	179, 186
<code>\IfLang@StrExpand</code>	182, 190
<code>\IfLang@StrNil</code>	173, 175, 178, 194, 195, 197, 198
<code>\IfLanguageName</code> .	2, 200, 215, 259, 296
<code>\IfLanguagePatterns</code>	2, 153, 228, 250, 295
<code>\ifnum</code>	90, 136, 154, 155, 201, 216
<code>\ifx</code>	10, 12, 18, 39, 47, 68, 71, 75, 82, 91, 107, 113, 129, 131, 148, 167, 178, 226, 272, 275, 276, 289, 324, 348
<code>\immediate</code>	20, 41
<code>\IncludeTests</code>	243
<code>\input</code>	108, 282, 319
L	
<code>\lang@USenglish</code>	133, 136
<code>\language</code>	136, 155, 230
<code>\languageName</code>	125, 137, 140, 150, 203, 217, 228, 232
<code>\LogTests</code>	244
M	
<code>\makeatletter</code>	269
<code>\meaning</code>	336
<code>\MessageBreak</code>	140, 231
N	
<code>\NeedsTeXFormat</code>	239
<code>\nofiles</code>	240
<code>\number</code>	175
P	
<code>\PackageInfo</code>	23
<code>\pdfstrcmp</code>	217, 318
<code>\ProvidesPackage</code>	53
R	
<code>\RequirePackage</code>	110

S		U	
\selectlanguage 263	\UNDEFINED	... 273, 276, 317, 318, 331
\space 133, 140, 150, 230	\uselanguage 123, 124, 304
\strip@prefix 334, 336	\usepackage 242, 245, 246
T		W	
\test 249, 252, 253, 254, 255, 258, 261, 262, 264, 265, 266	\write 20, 41
\TestCompare 338, 358, 359, 360, 361, 368, 369, 370, 371, 372		
\TestDefined 322, 330, 332	X	
\TestGeneric 285, 295, 296	\x 8, 10, 12, 19, 23, 25, 40, 45, 52, 287, 289, 340, 348
\TestName 296, 301, 302, 309, 310		
\TestPatterns	. 295, 298, 299, 306, 307	Y	
\the 56, 57, 58, 59, 60	\y 288, 289