

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	8
4	Installation	10
4.1	Download	10
4.2	Bundle installation	10
4.3	Package installation	10
4.4	Refresh file name databases	10
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]	11
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   \catcode44 12 % ,
32   \catcode45 12 % -
33   \catcode46 12 % .
34   \catcode58 12 % :
35   \catcode64 11 % @
36   \expandafter\ifx\csname ProvidesPackage\endcsname\relax
37     \def\x#1#2#3[#4]{\endgroup
38       \immediate\write-1{Package: #3 #4}%
39       \xdef#1{#4}%
40     }%
41   \else
42     \def\x#1#2[#3]{\endgroup
43       #2[#{#3}]%
44       \ifx#1\relax
45         \xdef#1{#3}%
46       \fi
47     }%
48   \fi
49 \expandafter\x\csname ver@iflang.sty\endcsname
50 \ProvidesPackage{iflang}%
51 [2007/04/26 v1.3 Language checks (H0)]
52 \expandafter\edef\csname IfLang@endinput\endcsname{%
53   \catcode39 \the\catcode39\relax % '
54   \catcode40 \the\catcode40\relax % (
55   \catcode41 \the\catcode41\relax % )
56   \catcode61 \the\catcode61\relax % =
57   \catcode64 \the\catcode64\relax % @
58   \noexpand\endinput
59 }
```

```

60 \catcode39 12\relax % '
61 \catcode40 12\relax % (
62 \catcode41 12\relax % )
63 \catcode61 12\relax % =
64 \catcode64 11\relax % @

```

2.2 Tools

2.2.1 Provide some basic macros of L^AT_EX

```

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

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

```

2.2.2 Expandible existence check for macros

```

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

```

2.2.3 Macros for messages

```

103 \begingroup\expandafter\expandafter\expandafter\endgroup
104 \expandafter\ifx\csname RequirePackage\endcsname\relax
105 \input infwarerr.sty\relax
106 \else

```

```

107 \RequirePackage{infwarerr}%
108 \fi

```

2.2.4 Support for etex.src

\IfLang@prefix

```

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

```

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.

```

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

```

2.3 \IfLanguagePatterns

\IfLanguagePatterns

```

150 \def\IfLanguagePatterns#1{%
151   \ifnum\IfLang@IfDefined{\IfLang@prefix#1}%
152     \ifnum\csname\IfLang@prefix#1\endcsname=\language
153       0%
154     \else

```

```

155         1%
156     \fi
157     }{1}=0 %
158     \expandafter\@firstoftwo
159 \else
160     \expandafter\@secondoftwo
161 \fi
162 }

```

2.4 \IfLanguageName

```

163 \begingroup\expandafter\expandafter\expandafter\endgroup
164 \expandafter\ifx\csname pdfstrcmp\endcsname\relax
165     \expandafter\@firstoftwo
166 \else
167     \expandafter\@secondoftwo
168 \fi
169 {%

```

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 Bacho \TeX 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 `\languageName` might consists of further macros, we need a variant that allows macros in the first string, too.

```

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

```

\IfLanguageName

```

197 \def\IfLanguageName#1{%
198     \ifnum\IfLang@ifDefined{languageName}%
199         \if\expandafter\IfLang@strequal\expandafter%
200             {\languageName}{#1}%
201         0%
202     \else
203         1%
204     \fi

```

```

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

\IfLanguageName

```

212 \def\IfLanguageName#1{%
213     \ifnum\IfLang@IfDefined{language} {%
214         \pdfstrcmp{#1}{\language}%
215         }{1}=0 %
216         \expandafter\@firstoftwo
217     \else
218         \expandafter\@secondoftwo
219     \fi
220 }%
221 }
```

2.5 Check plausibility of \language

```

222 \begingroup\expandafter\expandafter\expandafter\endgroup
223 \expandafter\ifx\csname language\endcsname\relax
224 \else
225     \IfLanguagePatterns{\language}{}{%
226         \@PackageWarningNoLine{iflang}{%
227             Mismatch between \string\language\space
228             (patterns)\MessageBreak
229             and setting of \string\language
230         }%
231     }%
232 \fi
233 \IfLang@endinput
234 \end{package}
```

3 Test

3.1 Test with L^AT_EX

```

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

```

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

```

3.2 Test with plain- \TeX and ε - \TeX

```

277 <*test2>
278 %% Format 'etex' based on 'language.def'
279 \input iflang.sty
280 \catcode64=12
281
282 \def\TestGeneric#1#2#3{%
283   \begingroup
284     \edef\x{#1{#2}{true}{false}}%
285     \edef\y{#3}%
286     \ifx\x\y
287     \else
288       \errmessage{Failed test: \string#1{#2} <> #3}%
289     \fi
290   \endgroup
291 }
292 \def\TestPatterns{\TestGeneric\IfLanguagePatterns}
293 \def\TestName{\TestGeneric\IfLanguageName}
294
295 \TestPatterns{USenglish}{true}
296 \TestPatterns{ngerman}{false}
297
298 \TestName{USenglish}{true}
299 \TestName{ngerman}{false}
300
301 \uselanguage{ngerman}
302
303 \TestPatterns{USenglish}{false}
304 \TestPatterns{ngerman}{true}
305
306 \TestName{USenglish}{false}
307 \TestName{ngerman}{true}
308
309 \csname @@end\endcsname
310 \end
311 </test2>

```

3.3 Test with plain- \TeX and without ε - \TeX /pdf \TeX


```

312 <*test3>

313 %% Format 'tex' (vanilla plain-TeX)
314 \let\ifcsname\UNDEFINED
315 \let\pdfstrcmp\UNDEFINED
316 \input iflang.sty
317 \catcode64=11
318
319 \def\TestDefined#1{%
320   \IfLang@IfDefined{foobar}{-}{-}%
321   \ifx\foobar#1%
322   \else
323     \errmessage{Failed test: \string\foobar <> \string#1}%
324   \fi
325 }
326 \let\foobar\relax
327 \TestDefined\relax
328 \let\foobar\UNDEFINED
329 \TestDefined\relax
330
331 \def\strip@prefix#1>{}
332 \def@onelevel@sanitize#1{%
333   \edef#1{\expandafter\strip@prefix\meaning#1}%
334 }
335 \def\TestCompare#1#2#3{%
336   \begingroup
337     \edef\x{%
338       \if\IfLang@StrEqual{#1}{#2}%
339         true%
340       \else
341         false%
342       \fi
343     }%
344     \def\expect{#3}%
345     \ifx\x\expect
346     \else
347       \def\a{#1}%
348       \@onelevel@sanitize\a
349       \def\b{#2}%
350       \@onelevel@sanitize\b
351       \errmessage{Failed test: '\a'='\b' <> \expect}%
352     \fi
353   \endgroup
354 }
355 \TestCompare{junk}{junk}{true}
356 \TestCompare{}{}{true}
357 \TestCompare{a}{b}{false}
358 \TestCompare{aa}{bb}{false}
359 \def\a{ax}
360 \def\b{bx}
361 \def\c{\a\b}
362 \def\d{\c\b}
363 \def\exch#1#2{#2#1}
364 \def\gobble#1{}
365 \TestCompare{\gobble a}{-}{true}
366 \TestCompare{}{\gobble a}{true}
367 \TestCompare{a\exch xyb}{ayxb}{true}
368 \TestCompare{\c}{\c}{true}
369 \TestCompare{\d}{\c\b}{true}
370
371 \csname @@end\endcsname
372 \end
373 </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](#)

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-TeX:

```
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.

4.4 Refresh file name databases

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

¹<http://ftp.ctan.org/tex-archive/>

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

[2007/04/26 v1.3]

- 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> . . .	111, 116, 146
<code>\@PackageWarningNoLine</code>	129, 136, 226
<code>\@firstoftwo</code>	65, 73, 82, 97, 158, 165, 206, 216
<code>\@onelevel@sanitize</code> . . .	332, 348, 350
<code>\@secondoftwo</code>	68, 75, 80, 99, 160, 167, 208, 218
A	
<code>\a</code>	347, 348, 351, 359, 361
B	
<code>\b</code> . . .	349, 350, 351, 360, 361, 362, 369
<code>\begin</code>	244, 245, 254, 265
C	
<code>\c</code>	361, 362, 368, 369
<code>\catcode</code>	3, 4, 5, 6, 7, 31, 32, 33, 34, 35, 53, 54, 55, 56, 57, 60, 61, 62, 63, 64, 280, 317
<code>\csname</code>	8, 18, 36, 49, 52, 65, 68, 72, 79, 88, 104, 110, 126, 128, 145, 152, 164, 223, 309, 371
D	
<code>\d</code>	362, 369
<code>\documentclass</code>	238
E	
<code>\empty</code>	12
<code>\end</code>	253, 264, 274, 275, 310, 372
<code>\endcsname</code> . . .	8, 18, 36, 49, 52, 65, 68, 72, 79, 87, 88, 104, 110, 126, 128, 145, 152, 164, 223, 309, 371
<code>\endinginput</code>	27, 58
<code>\errmessage</code>	288, 323, 351
<code>\exch</code>	363, 367
<code>\Expect</code>	247, 256, 268, 269, 271, 272, 273
<code>\expect</code>	344, 345, 351
F	
<code>\foobar</code>	267, 269, 270, 272, 273, 321, 323, 326, 328
G	
<code>\gobble</code>	364, 365, 366
I	
<code>\if</code>	181, 199, 338
<code>\ifcase</code>	9
<code>\ifcat</code>	178
<code>\ifcsname</code>	87, 314
<code>\IfLang@@@StrExpand</code>	192, 194
<code>\IfLang@@StrExpand</code>	189, 191
<code>\IfLang@endinput</code>	233
<code>\IfLang@ifDefined</code>	71, 151, 198, 213, 268, 271, 320
<code>\IfLang@OrgUseLanguage</code>	120, 123
<code>\IfLang@prefix</code>	109, 151, 152
<code>\IfLang@StrEqual</code>	171, 338
<code>\IfLang@strequal</code>	199
<code>\IfLang@StrEqualStart</code>	172, 174, 181, 183, 187, 195
<code>\IfLang@StrEqualStop</code>	176, 183
<code>\IfLang@StrExpand</code>	179, 187
<code>\IfLang@StrNil</code>	170, 172, 175, 191, 192, 194, 195
<code>\IfLanguageName</code> .	2, 197, 212, 256, 293
<code>\IfLanguagePatterns</code>	2, 150, 225, 247, 292
<code>\ifnum</code>	87, 133, 151, 152, 198, 213
<code>\ifx</code>	10, 12, 18, 36, 44, 65, 68, 72, 79, 88, 104, 110, 126, 128, 145, 164, 175, 223, 269, 272, 273, 286, 321, 345
<code>\immediate</code>	20, 38
<code>\IncludeTests</code>	240
<code>\input</code>	105, 279, 316
L	
<code>\lang@USenglish</code>	130, 133
<code>\language</code>	133, 152, 227
<code>\languageName</code>	122, 134, 137, 147, 200, 214, 225, 229
<code>\LogTests</code>	241
M	
<code>\makeatletter</code>	266
<code>\meaning</code>	333
<code>\MessageBreak</code>	137, 228
N	
<code>\NeedsTeXFormat</code>	236
<code>\nofiles</code>	237
<code>\number</code>	172
P	
<code>\PackageInfo</code>	23
<code>\pdfstrcmp</code>	214, 315
<code>\ProvidesPackage</code>	50
R	
<code>\RequirePackage</code>	107
S	
<code>\selectlanguage</code>	260
<code>\space</code>	130, 137, 147, 227
<code>\strip@prefix</code>	331, 333
T	
<code>\test</code>	246, 249, 250, 251, 252, 255, 258, 259, 261, 262, 263
<code>\TestCompare</code>	335, 355, 356, 357, 358, 365, 366, 367, 368, 369
<code>\TestDefined</code>	319, 327, 329
<code>\TestGeneric</code>	282, 292, 293

<code>\TestName</code>	293, 298, 299, 306, 307	W
<code>\TestPatterns</code> .	292, 295, 296, 303, 304	<code>\write</code> 20, 38
<code>\the</code>	53, 54, 55, 56, 57	X
U		<code>\x</code> 8, 10, 12, 19,
<code>\UNDEFINED</code> . . .	270, 273, 314, 315, 328	23, 25, 37, 42, 49, 284, 286, 337, 345
<code>\uselanguage</code>	120, 121, 301	Y
<code>\usepackage</code>	239, 242, 243	<code>\y</code> 285, 286