

The iflang package

Heiko Oberdiek
<oberdiek@uni-freiburg.de>

2007/04/12 v1.2

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>	6
2.4	<code>\IfLanguageName</code>	6
2.5	Check plausibility of <code>\language</code>	8
3	Test	8
3.1	Test with L ^A T _E X	8
3.2	Test with plain-T _E X and ε -T _E X	9
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	11
4.4	Refresh file name databases	11
4.5	Some details for the interested	11
5	Acknowledgement	12
6	History	12
	[2007/04/10 v1.0]	12
	[2007/04/11 v1.1]	12
	[2007/04/12 v1.2]	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   \expandafter\let\expandafter\x\csname ver@iflang.sty\endcsname
4   \ifcase 0%
5     \ifx\x\relax % plain
6     \else
7       \ifx\x\empty % LaTeX
8       \else
9         1%
10      \fi
11    \fi
12  \else
13    \expandafter\ifx\csname PackageInfo\endcsname\relax
14      \def\x#1#2{%
15        \immediate\write-1{Package #1 Info: #2.}%
16      }%
17    \else
18      \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
19    \fi
20    \x@iflang{The package is already loaded}%
21  \endgroup
22  \expandafter\endinput
23  \fi
24 \endgroup
```

Package identification:

```
25 \begingroup
26   \expandafter\ifx\csname ProvidesPackage\endcsname\relax
27     \def\x#1#2#3[#4]{\endgroup
28       \immediate\write-1{Package: #3 #4}%
29       \xdef#1{#4}%
30     }%
31   \else
32     \def\x#1#2[#3]{\endgroup
33       #2[#{#3}]%
34       \ifx#1\relax
35         \xdef#1{#3}%
36       \fi
37     }%
38   \fi
39 \expandafter\x\csname ver@iflang.sty\endcsname
40 \ProvidesPackage{iflang}%
41 [2007/04/12 v1.2 Language checks (H0)]
42 \expandafter\edef\csname IfLang@endinput\endcsname{%
43   \catcode39 \the\catcode39\relax % '
44   \catcode40 \the\catcode40\relax % (
45   \catcode41 \the\catcode41\relax % )
46   \catcode61 \the\catcode61\relax % =
47   \catcode64 \the\catcode64\relax % @
48   \noexpand\endinput
49 }
50 \catcode39 12\relax % '
51 \catcode40 12\relax % (
52 \catcode41 12\relax % )
53 \catcode61 12\relax % =
54 \catcode64 11\relax % @
```

2.2 Tools

2.2.1 Provide some basic macros of L^AT_EX

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

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

2.2.2 Expandible existence check for macros

```
\IfLang@ifDefined
61 \begingroup\expandafter\expandafter\expandafter\endgroup
62 \expandafter\ifx\csname ifcname\endcsname\relax
63   \expandafter\@firstoftwo
64 \else
65   \expandafter\@secondoftwo
66 \fi
67 {%
68   \def\IfLang@ifDefined#1{%
69     \expandafter\ifx\csname#1\endcsname\relax
70       \expandafter\@secondoftwo
71     \else
72       \expandafter\@firstoftwo
73     \fi
74   }%
75 }{%
76   \def\IfLang@ifDefined#1{%
77     \ifnum\ifcname#1\endcsname
78       \expandafter\ifx\csname#1\endcsname\relax
79         1%
80       \else
81         0%
82       \fi
83     \else
84       1%
85     \fi
86     =0 %
87     \expandafter\@firstoftwo
88   \else
89     \expandafter\@secondoftwo
90   \fi
91 }%
92 }
```

2.2.3 Macros for messages

```
93 \begingroup
94 \ifnum\expandafter\ifx\csname PackageInfo\endcsname\relax
95   1%
96 \fi
97 \expandafter\ifx\csname PackageWarningNoLine\endcsname\relax
98   1%
99 \fi
100   0=0 %
101 \endgroup
102 \expandafter\@secondoftwo
```

```

103 \else
104   \endgroup
105   \expandafter\@firstoftwo
106 \fi
107 {%
108   \def\IfLang@GenericMessage#1#2#3{%
109     \begingroup
110     \newlinechar=10 %
111     \def\@spaces{\space\space\space\space}%
112     \ifcase#1 %
113       \def\MessageBreak{%
114         ^^J(iflang) \@spaces\@spaces\@spaces\@spaces
115       }%
116     \or
117       \def\MessageBreak{%
118         ^^J(iflang)\@spaces\@spaces\@spaces\@spaces
119       }%
120     \or
121       \def\MessageBreak{%
122         ^^J(iflang) \space\@spaces\@spaces\@spaces
123       }%
124     \fi
125     \expandafter\ifx\csname if#3\expandafter\endcsname
126       \csname iftrue\endcsname
127     \def\on@line{ on input line \the\inputlineno}%
128   \else
129     \def\on@line{}%
130   \fi
131   \immediate\write\ifcase#1 -1\or 16\fi{%
132     ^^JPackage iflang %
133     \ifcase#1 Info\or Warning\or Error\fi
134     : #2\on@line.^^J%
135   }%
136   \endgroup
137 }%
138 \def\IfLang@Info#1{%
139   \IfLang@GenericMessage{0}{#1}{false}%
140 }%
141 \def\IfLang@Warning#1{%
142   \IfLang@GenericMessage{1}{#1}{false}%
143 }%
144 }{%
145   \def\IfLang@Info#1{%
146     \begingroup
147     \def\on@line{}%
148     \PackageInfo{iflang}{#1}%
149   \endgroup
150 }%
151 \def\IfLang@Warning#1{%
152   \PackageWarningNoLine{iflang}{#1}%
153 }%
154 }

```

2.2.4 Support for etex.src

\IfLang@prefix

```

155 \begingroup\expandafter\expandafter\expandafter\endgroup
156 \expandafter\ifx\csname uselanguage\endcsname\relax
157   \IfLang@Info{Naming convention for patterns: babel}%
158   \def\IfLang@prefix{1@}%
159 \else
160   \IfLang@Info{Naming convention for patterns: etex.src}%
161   \def\IfLang@prefix{lang@}%

```

```

162 \let\IfLang@OrgUseLanguage\uselanguage
163 \def\uselanguage#1{%
164     \edef\language#1}%
165     \IfLang@OrgUseLanguage{#1}%
166 }%

```

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.

```

167 \begingroup\expandafter\expandafter\expandafter\endgroup
168 \expandafter\ifx\csname language\endcsname\relax
169     \begingroup\expandafter\expandafter\expandafter\endgroup
170     \expandafter\ifx\csname lang@USenglish\endcsname\relax
171         \IfLang@Warning{\string\lang@USenglish\space is missing}%
172     \else
173         \ifnum\lang@USenglish=\language
174             \def\language{USenglish}%
175         \else
176             \IfLang@Warning{%
177                 \string\language\space is not set,\MessageBreak
178                 current language is unknown%
179             }%
180         \fi
181     \fi
182 \fi
183 \fi
184 \begingroup\expandafter\expandafter\expandafter\endgroup
185 \expandafter\ifx\csname language\endcsname\relax
186     \IfLang@Info{\string\language\space is not set}%
187 \fi

```

2.3 \IfLanguagePatterns

`\IfLanguagePatterns`

```

188 \def\IfLanguagePatterns#1{%
189     \ifnum\IfLang@IfDefined{\IfLang@prefix#1}{%
190         \ifnum\csname\IfLang@prefix#1\endcsname=\language
191             0%
192         \else
193             1%
194         \fi
195     }{1}=0 %
196     \expandafter\@firstoftwo
197 \else
198     \expandafter\@secondoftwo
199 \fi
200 }

```

2.4 \IfLanguageName

```

201 \begingroup\expandafter\expandafter\expandafter\endgroup
202 \expandafter\ifx\csname pdfstrcmp\endcsname\relax
203     \expandafter\@firstoftwo
204 \else
205     \expandafter\@secondoftwo
206 \fi
207 {%

```

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 Kas-

trup, presented at the conference BachoTeX 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.

```

208 \def\IfLang@StrNil{\relax}%
209 \def\IfLang@StrEqual#1{%
210   \number\IfLang@StrEqualStart{#1}\IfLang@StrNil
211 }%
212 \def\IfLang@StrEqualStart#1#2#3{%
213   \ifx#3\IfLang@StrNil
214     \IfLang@StrEqualStop
215   \fi
216   \ifcat\noexpand#3\relax
217     \IfLang@StrExpand{#1}{#2}#3%
218   \fi
219   \IfLang@StrEqualStart{\if#3#1}{#2\fi}%
220 }%
221 \def\IfLang@StrEqualStop\fi#1\IfLang@StrEqualStart#2#3#4{%
222   \fi
223   #2#4\relax'#313 %
224 }%
225 \def\IfLang@StrExpand#1#2#3\fi\IfLang@StrEqualStart#4#5{%
226   \fi
227   \IfLang@@StrExpand{#1}{#2}#3%
228 }%
229 \def\IfLang@@StrExpand#1#2#3\IfLang@StrNil{%
230   \expandafter\IfLang@@@StrExpand#3\IfLang@StrNil{#1}{#2}%
231 }%
232 \def\IfLang@@@StrExpand#1\IfLang@StrNil#2#3{%
233   \IfLang@StrEqualStart{#2}{#3}#1\IfLang@StrNil
234 }%

```

`\IfLanguageName`

```

235 \def\IfLanguageName#1{%
236   \ifnum\IfLang@IfDefined{languageName}{%
237     \if\expandafter\IfLang@strequal\expandafter%
238       {\languageName}{#1}%
239     0%
240   \else
241     1%
242   \fi
243   }{1}=0 %
244   \expandafter\@firstoftwo
245 \else
246   \expandafter\@secondoftwo
247 \fi
248 }%
249 }%

```

`\IfLanguageName`

```

250 \def\IfLanguageName#1{%
251   \ifnum\IfLang@IfDefined{languageName}{%
252     \pdfstrcmp{#1}{\languageName}%
253     }{1}=0 %
254   \expandafter\@firstoftwo
255 \else
256   \expandafter\@secondoftwo
257 \fi
258 }%
259 }

```

2.5 Check plausibility of \language

```
260 \begingroup\expandafter\expandafter\expandafter\endgroup
261 \expandafter\ifx\csname language\endcsname\relax
262 \else
263   \IfLanguagePatterns{\language}{}{%
264     \IfLang@Warning{%
265       Mismatch between \string\language\space
266       (patterns)\MessageBreak
267       and setting of \string\language
268     }%
269   }%
270 \fi

271 \IfLang@endinput
272 \end{package}
```

3 Test

3.1 Test with L^AT_EX

```
273 \testl

274 \NeedsTeXFormat{LaTeX2e}
275 \nofiles
276 \documentclass{minimal}
277 \usepackage{qstest}
278 \IncludeTests{*}
279 \LogTests{lgout}{*}{*}
280 \usepackage[english,naustrian,ngerman]{babel}
281 \usepackage{iflang}
282 \begin{document}
283 \begin{qstest}\IfLanguagePatterns{language, pattern}
284   \def\test#1#2{%
285     \Expect*{\IfLanguagePatterns{#1}{true}{false}}{#2}%
286   }%
287   \test{ngerman}{true}%
288   \test{naustrian}{true}%
289   \test{english}{false}%
290   \test{foobar}{false}%
291 \end{qstest}
292 \begin{qstest}\IfLanguageName{language, name}
293   \def\test#1#2{%
294     \Expect*{\IfLanguageName{#1}{true}{false}}{#2}%
295   }%
296   \test{ngerman}{true}%
297   \test{naustrian}{false}%
298   \selectlanguage{naustrian}%
299   \test{ngerman}{false}%
300   \test{naustrian}{true}%
301   \test{foobar}{false}%
302 \end{qstest}
303 \begin{qstest}\IfDefined{defined}
304   \makeatletter
305   \let\foobar\relax
306   \Expect*{\IfLang@IfDefined{foobar}{true}{false}}{false}%
307   \Expect*{\ifx\foobar\relax true\else false\fi}{true}%
308   \let\foobar\UNDEFINED
309   \Expect*{\IfLang@IfDefined{foobar}{true}{false}}{false}%
310   \Expect*{\ifx\foobar\relax true\else false\fi}{false}%
311   \Expect*{\ifx\foobar\UNDEFINED true\else false\fi}{true}%
312 \end{qstest}
313 \end{document}

314 \end{testl}
```


3.2 Test with plain-TeX and ε -TeX

```
315 (*test2)
316 %% Format 'etex' based on 'language.def'
317 \input iflang.sty
318 \catcode64=12
319
320 \def\TestGeneric#1#2#3{%
321   \begingroup
322     \edef\x{#1{#2}{true}{false}}%
323     \edef\y{#3}%
324     \ifx\x\y
325     \else
326       \errmessage{Failed test: \string#1{#2} <> #3}%
327     \fi
328   \endgroup
329 }
330 \def\TestPatterns{\TestGeneric\IfLanguagePatterns}
331 \def\TestName{\TestGeneric\IfLanguageName}
332
333 \TestPatterns{USenglish}{true}
334 \TestPatterns{ngerman}{false}
335
336 \TestName{USenglish}{true}
337 \TestName{ngerman}{false}
338
339 \uselanguage{ngerman}
340
341 \TestPatterns{USenglish}{false}
342 \TestPatterns{ngerman}{true}
343
344 \TestName{USenglish}{false}
345 \TestName{ngerman}{true}
346
347 \csname @@end\endcsname
348 \end
349 (/test2)
```

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

```
350 (*test3)
351 %% Format 'tex' (vanilla plain-TeX)
352 \let\ifcname\UNDEFINED
353 \let\pdfstrcmp\UNDEFINED
354 \input iflang.sty
355 \catcode64=11
356
357 \def\TestDefined#1{%
358   \IfLang@IfDefined{foobar}{-}{-}%
359   \ifx\foobar#1%
360   \else
361     \errmessage{Failed test: \string\foobar <> \string#1}%
362   \fi
363 }
364 \let\foobar\relax
365 \TestDefined\relax
366 \let\foobar\UNDEFINED
367 \TestDefined\relax
368
369 \def\strip@prefix#1>{}
370 \def\@onelevel@sanitize#1{%
371   \edef#1{\expandafter\strip@prefix\meaning#1}%
372 }
```

```

373 \def\TestCompare#1#2#3{%
374   \begingroup
375   \edef\x{%
376     \if\IfLang@StrEqual{#1}{#2}%
377     true%
378     \else
379     false%
380     \fi
381   }%
382   \def\expect{#3}%
383   \ifx\x\expect
384   \else
385     \def\a{#1}%
386     \@onelevel@sanitize\a
387     \def\b{#2}%
388     \@onelevel@sanitize\b
389     \errmessage{Failed test: '\a'='\b' <> \expect}%
390   \fi
391   \endgroup
392 }
393 \TestCompare{junk}{junk}{true}
394 \TestCompare{}{}{true}
395 \TestCompare{a}{b}{false}
396 \TestCompare{aa}{bb}{false}
397 \def\a{ax}
398 \def\b{bx}
399 \def\c{\a\b}
400 \def\d{\c\b}
401 \def\exch#1#2#2#1}
402 \def\gobble#1{}
403 \TestCompare{\gobble a}{}{true}
404 \TestCompare{}{\gobble a}{true}
405 \TestCompare{a\exch xyb}{ayxb}{true}
406 \TestCompare{\c}{\c}{true}
407 \TestCompare{\d}{\c\b}{true}
408
409 \csname @@end\endcsname
410 \end
411 </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
```

¹<http://ftp.ctan.org/tex-archive/>

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 (te \TeX , mi \TeX , ...) relies on file name databases, you must refresh these. For example, te \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 pdfL^AT_EX:

```
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.

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	D
<code>\@firstoftwo</code> <u>55</u> , 63, 72, 87, 105, 196, 203, 244, 254	<code>\d</code> 400, 407
<code>\@onelevel@sanitize</code> ... 370, 386, 388	<code>\documentclass</code> 276
<code>\@secondoftwo</code> <u>58</u> , 65, 70, 89, 102, 198, 205, 246, 256	E
<code>\@spaces</code> 111, 114, 118, 122	<code>\empty</code> 7
A	<code>\end</code> 291, 302, 312, 313, 348, 410
<code>\a</code> 385, 386, 389, 397, 399	<code>\endcsname</code> 3, 13, 26, 39, 42, 55, 58, 62, 69, 77, 78, 94, 97, 125, 126, 156, 168, 170, 185, 190, 202, 261, 347, 409
B	<code>\endinput</code> 22, 48
<code>\b</code> ... 387, 388, 389, 398, 399, 400, 407	<code>\errmessage</code> 326, 361, 389
<code>\begin</code> 282, 283, 292, 303	<code>\exch</code> 401, 405
C	<code>\Expect</code> 285, 294, 306, 307, 309, 310, 311
<code>\c</code> 399, 400, 406, 407	<code>\expect</code> 382, 383, 389
<code>\catcode</code> 43, 44, 45, 46, 47, 50, 51, 52, 53, 54, 318, 355	F
<code>\csname</code> 3, 13, 26, 39, 42, 55, 58, 62, 69, 78, 94, 97, 125, 126, 156, 168, 170, 185, 190, 202, 261, 347, 409	<code>\foobar</code> 305, 307, 308, 310, 311, 359, 361, 364, 366
	G
	<code>\gobble</code> 402, 403, 404

I		<code>\MessageBreak</code> . 113, 117, 121, 177, 266
<code>\if</code>	219, 237, 376	N
<code>\ifcase</code>	4, 112, 131, 133	<code>\NeedsTeXFormat</code> 274
<code>\ifcat</code>	216	<code>\newlinechar</code> 110
<code>\ifcsname</code>	77, 352	<code>\nofiles</code> 275
<code>\IfLang@@@StrExpand</code>	230, 232	<code>\number</code> 210
<code>\IfLang@@StrExpand</code>	227, 229	O
<code>\IfLang@endinput</code>	271	<code>\on@line</code> 127, 129, 134, 147
<code>\IfLang@GenericMessage</code>	108, 139, 142	P
<code>\IfLang@ifDefined</code>	61, 189, 236, 251, 306, 309, 358	<code>\PackageInfo</code> 18, 148
<code>\IfLang@Info</code>	138, 145, 157, 160, 186	<code>\PackageWarningNoLine</code> 152
<code>\IfLang@OrgUseLanguage</code>	162, 165	<code>\pdfstrcmp</code> 252, 353
<code>\IfLang@prefix</code>	155, 189, 190	<code>\ProvidesPackage</code> 40
<code>\IfLang@StrEqual</code>	209, 376	S
<code>\IfLang@strequal</code>	237	<code>\selectlanguage</code> 298
<code>\IfLang@StrEqualStart</code>	210, 212, 219, 221, 225, 233	<code>\space</code> 111, 122, 171, 177, 186, 265
<code>\IfLang@StrEqualStop</code>	214, 221	<code>\strip@prefix</code> 369, 371
<code>\IfLang@StrExpand</code>	217, 225	T
<code>\IfLang@StrNil</code>	208, 210, 213, 229, 230, 232, 233	<code>\test</code> 284, 287, 288, 289, 290, 293, 296, 297, 299, 300, 301
<code>\IfLang@Warning</code>	141, 151, 171, 176, 264	<code>\TestCompare</code> 373, 393, 394, 395, 396, 403, 404, 405, 406, 407
<code>\IfLanguageName</code>	2, 235, 250, 294, 331	<code>\TestDefined</code> 357, 365, 367
<code>\IfLanguagePatterns</code>	2, 188, 263, 285, 330	<code>\TestGeneric</code> 320, 330, 331
<code>\ifnum</code>	77, 94, 173, 189, 190, 236, 251	<code>\TestName</code> 331, 336, 337, 344, 345
<code>\ifx</code>	5, 7, 13, 26, 34, 55, 58, 62, 69, 78, 94, 97, 125, 156, 168, 170, 185, 202, 213, 261, 307, 310, 311, 324, 359, 383	<code>\TestPatterns</code> 330, 333, 334, 341, 342
<code>\immediate</code>	15, 28, 131	<code>\the</code> 43, 44, 45, 46, 47, 127
<code>\IncludeTests</code>	278	U
<code>\input</code>	317, 354	<code>\UNDEFINED</code> 308, 311, 352, 353, 366
<code>\inputlineno</code>	127	<code>\uselanguage</code> 162, 163, 339
L		<code>\usepackage</code> 277, 280, 281
<code>\lang@USenglish</code>	171, 173	W
<code>\language</code>	173, 190, 265	<code>\write</code> 15, 28, 131
<code>\languagename</code>	164, 174, 177, 186, 238, 252, 263, 267	X
<code>\LogTests</code>	279	<code>\x</code> 3, 5, 7, 14, 18, 20, 27, 32, 39, 322, 324, 375, 383
M		Y
<code>\makeatletter</code>	304	<code>\y</code> 323, 324
<code>\meaning</code>	371	