

# The iflang package

Heiko Oberdiek  
<heiko.oberdiek at gmail.com>

2007/11/11 v1.5

## Abstract

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

## Contents

<b>1</b>	<b>Documentation</b>	<b>2</b>
<b>2</b>	<b>Implementation</b>	<b>2</b>
2.1	Reload check and package identification . . . . .	3
2.2	Tools . . . . .	4
2.2.1	Provide some basic macros of L <sup>A</sup> T <sub>E</sub> X . . . . .	4
2.2.2	Expandible existence check for macros . . . . .	5
2.2.3	Macros for messages . . . . .	5
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
<b>3</b>	<b>Test</b>	<b>8</b>
3.1	Catcode checks for loading . . . . .	8
3.2	Test with L <sup>A</sup> T <sub>E</sub> X . . . . .	10
3.3	Test with plain T <sub>E</sub> X and $\epsilon$ -T <sub>E</sub> X . . . . .	11
3.4	Test with plain T <sub>E</sub> X and without $\epsilon$ -T <sub>E</sub> X/pdfT <sub>E</sub> X . . . . .	11
<b>4</b>	<b>Installation</b>	<b>13</b>
4.1	Download . . . . .	13
4.2	Bundle installation . . . . .	13
4.3	Package installation . . . . .	13
4.4	Refresh file name databases . . . . .	13
4.5	Some details for the interested . . . . .	14
<b>5</b>	<b>Acknowledgement</b>	<b>14</b>
<b>6</b>	<b>History</b>	<b>14</b>
	[2007/04/10 v1.0] . . . . .	14
	[2007/04/11 v1.1] . . . . .	14
	[2007/04/12 v1.2] . . . . .	14
	[2007/04/26 v1.3] . . . . .	14
	[2007/09/09 v1.4] . . . . .	15
	[2007/11/11 v1.5] . . . . .	15
<b>7</b>	<b>Index</b>	<b>15</b>

# 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 $\epsilon$ -**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<sup>A</sup>T<sub>E</sub>X.

```
2 \begingroup\catcode61\catcode48\catcode32=10\relax%
3 \catcode13=5 % ^~M
4 \endlinechar=13 %
5 \catcode35=6 % #
6 \catcode39=12 % '
7 \catcode44=12 % ,
8 \catcode45=12 % -
9 \catcode46=12 % .
10 \catcode58=12 % :
11 \catcode64=11 % @
12 \catcode123=1 % {
13 \catcode125=2 % }
14 \expandafter\let\expandafter\x\csname ver@iflang.sty\endcsname
15 \ifx\x\relax % plain-TeX, first loading
16 \else
17 \def\empty{}%
18 \ifx\x\empty % LaTeX, first loading,
19 % variable is initialized, but \ProvidesPackage not yet seen
20 \else
21 \expandafter\ifx\csname PackageInfo\endcsname\relax
22 \def\x#1#2{%
23 \immediate\write-1{Package #1 Info: #2.}%
24 }%
25 \else
26 \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
27 \fi
28 \x@iflang}{The package is already loaded}%
29 \aftergroup\endinput
30 \fi
31 \fi
32 \endgroup%
```

Package identification:

```
33 \begingroup\catcode61\catcode48\catcode32=10\relax%
34 \catcode13=5 % ^~M
35 \endlinechar=13 %
36 \catcode35=6 % #
37 \catcode39=12 % '
38 \catcode40=12 % (
39 \catcode41=12 % )
40 \catcode44=12 % ,
41 \catcode45=12 % -
42 \catcode46=12 % .
43 \catcode47=12 % /
44 \catcode58=12 % :
45 \catcode64=11 % @
46 \catcode91=12 % [
47 \catcode93=12 % ]
48 \catcode123=1 % {
49 \catcode125=2 % }
50 \expandafter\ifx\csname ProvidesPackage\endcsname\relax
51 \def\x#1#2#3[#4]{\endgroup
52 \immediate\write-1{Package: #3 #4}%
53 \xdef#1{#4}%
54 }%
55 \else
56 \def\x#1#2[#3]{\endgroup
57 #2[#3]}%
58 \ifx#1\@undefined
59 \xdef#1{#3}%

```

```

60      \fi
61      \ifx#1\relax
62          \xdef#1{#3}%
63      \fi
64  }%
65  \fi
66  \expandafter\x\csname ver@iflang.sty\endcsname
67  \ProvidesPackage{iflang}%
68  [2007/11/11 v1.5 Language checks (H0)]%

69  \begingroup\catcode61\catcode48\catcode32=10\relax%
70  \catcode13=5 % ^^M
71  \endlinechar=13 %
72  \catcode123=1 % {
73  \catcode125=2 % }
74  \catcode64=11 % @
75  \def\x{\endgroup
76  \expandafter\edef\csname IfLang@AtEnd\endcsname{%
77  \endlinechar=\the\endlinechar\relax
78  \catcode13=\the\catcode13\relax
79  \catcode32=\the\catcode32\relax
80  \catcode35=\the\catcode35\relax
81  \catcode61=\the\catcode61\relax
82  \catcode64=\the\catcode64\relax
83  \catcode123=\the\catcode123\relax
84  \catcode125=\the\catcode125\relax
85  }%
86  }%
87  \x\catcode61\catcode48\catcode32=10\relax%
88  \catcode13=5 % ^^M
89  \endlinechar=13 %
90  \catcode35=6 % #
91  \catcode64=11 % @
92  \catcode123=1 % {
93  \catcode125=2 % }
94  \def\TMP@EnsureCode#1#2{%
95  \edef\IfLang@AtEnd{%
96  \IfLang@AtEnd
97  \catcode#1=\the\catcode#1\relax
98  }%
99  \catcode#1=#2\relax
100 }
101 \TMP@EnsureCode{39}{12}% '
102 \TMP@EnsureCode{40}{12}% (
103 \TMP@EnsureCode{41}{12}% )
104 \TMP@EnsureCode{44}{12}% ,
105 \TMP@EnsureCode{46}{12}% .
106 \TMP@EnsureCode{47}{12}% /
107 \TMP@EnsureCode{58}{12}% :
108 \TMP@EnsureCode{91}{12}% [
109 \TMP@EnsureCode{93}{12}% ]
110 \edef\IfLang@AtEnd{\IfLang@AtEnd\noexpand\endinput}

```

## 2.2 Tools

### 2.2.1 Provide some basic macros of L<sup>A</sup>T<sub>E</sub>X

\@firstoftwo

```

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

```

\@secondoftwo

```

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

```

## 2.2.2 Expandible existence check for macros

`\IfLang@IfDefined`

```

117 \begingroup\expandafter\expandafter\expandafter\endgroup
118 \expandafter\ifx\csname ifcsname\endcsname\relax
119   \expandafter\@firstoftwo
120 \else
121   \expandafter\@secondoftwo
122 \fi
123 {%
124   \def\IfLang@IfDefined#1{%
125     \expandafter\ifx\csname#1\endcsname\relax
126       \expandafter\@secondoftwo
127     \else
128       \expandafter\@firstoftwo
129     \fi
130   }%
131 }{%
132   \def\IfLang@IfDefined#1{%
133     \ifnum\ifcsname#1\endcsname
134       \expandafter\ifx\csname#1\endcsname\relax
135         1%
136       \else
137         0%
138       \fi
139     \else
140       1%
141     \fi
142     =0 %
143     \expandafter\@firstoftwo
144   \else
145     \expandafter\@secondoftwo
146   \fi
147 }%
148 }

```

## 2.2.3 Macros for messages

```

149 \begingroup\expandafter\expandafter\expandafter\endgroup
150 \expandafter\ifx\csname RequirePackage\endcsname\relax
151   \input infwarerr.sty\relax
152   \input pdftexcmds.sty\relax
153 \else
154   \RequirePackage{infwarerr}[2007/09/09]%
155   \RequirePackage{pdftexcmds}[2007/11/11]%
156 \fi

```

## 2.2.4 Support for etex.src

`\IfLang@prefix`

```

157 \begingroup\expandafter\expandafter\expandafter\endgroup
158 \expandafter\ifx\csname uselanguage\endcsname\relax
159   \@PackageInfoNoLine{iflang}{%
160     Naming convention for patterns: babel%
161   }%
162   \def\IfLang@prefix{l}%
163 \else
164   \@PackageInfoNoLine{iflang}{%
165     Naming convention for patterns: etex.src%

```

```

166 }%
167 \def\IfLang@prefix{lang@}%
168 \let\IfLang@OrgUseLanguage\uselanguage
169 \def\uselanguage#1{%
170   \edef\language{#1}%
171   \IfLang@OrgUseLanguage{#1}%
172 }%

```

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.

```

173 \begingroup\expandafter\expandafter\expandafter\endgroup
174 \expandafter\ifx\csname language\endcsname\relax
175   \begingroup\expandafter\expandafter\expandafter\endgroup
176   \expandafter\ifx\csname lang@USenglish\endcsname\relax
177     \@PackageWarningNoLine{iflang}{%
178       \string\lang@USenglish\space is missing%
179     }%
180   \else
181     \ifnum\lang@USenglish=\language
182       \def\language{USenglish}%
183     \else
184       \@PackageWarningNoLine{iflang}{%
185         \string\language\space is not set,\MessageBreak
186         current language is unknown%
187       }%
188     \fi
189   \fi
190 \fi
191 \fi
192 \begingroup\expandafter\expandafter\expandafter\endgroup
193 \expandafter\ifx\csname language\endcsname\relax
194   \@PackageInfoNoLine{iflang}{%
195     \string\language\space is not set%
196   }%
197 \fi

```

## 2.3 \IfLanguagePatterns

`\IfLanguagePatterns`

```

198 \def\IfLanguagePatterns#1{%
199   \ifnum\IfLang@IfDefined{\IfLang@prefix#1}{%
200     \ifnum\csname\IfLang@prefix#1\endcsname=\language
201       0%
202     \else
203       1%
204     \fi
205   }{1}=0 %
206   \expandafter\@firstoftwo
207 \else
208   \expandafter\@secondoftwo
209 \fi
210 }

```

## 2.4 \IfLanguageName

```

211 \begingroup\expandafter\expandafter\expandafter\endgroup
212 \expandafter\ifx\csname pdf@strcmp\endcsname\relax
213   \expandafter\@firstoftwo

```

```

214 \else
215   \expandafter\@secondoftwo
216 \fi
217 {%

```

We do not have `\pdf@strcmp` (and `\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.

```

218 \def\IfLang@StrNil{\relax}%
219 \def\IfLang@StrEqual#1{%
220   \number\IfLang@StrEqualStart{#1}\IfLang@StrNil
221 }%
222 \def\IfLang@StrEqualStart#1#2#3{%
223   \ifx#3\IfLang@StrNil
224     \IfLang@StrEqualStop
225   \fi
226   \ifcat\noexpand#3\relax
227     \IfLang@StrExpand{#1}{#2}#3%
228   \fi
229   \IfLang@StrEqualStart{\if#3#1}{#2\fi}%
230 }%
231 \def\IfLang@StrEqualStop\fi#1\IfLang@StrEqualStart#2#3#4{%
232   \fi
233   #2#4\relax'#313 %
234 }%
235 \def\IfLang@StrExpand#1#2#3\fi\IfLang@StrEqualStart#4#5{%
236   \fi
237   \IfLang@@StrExpand{#1}{#2}#3%
238 }%
239 \def\IfLang@@StrExpand#1#2#3\IfLang@StrNil{%
240   \expandafter\IfLang@@@StrExpand#3\IfLang@StrNil{#1}{#2}%
241 }%
242 \def\IfLang@@@StrExpand#1\IfLang@StrNil#2#3{%
243   \IfLang@StrEqualStart{#2}{#3}#1\IfLang@StrNil
244 }%

```

`\IfLanguageName`

```

245 \def\IfLanguageName#1{%
246   \ifnum\IfLang@IfDefined{languageName}%
247     \if\expandafter\IfLang@StrEqual\expandafter%
248       {\languageName}{#1}%
249     0%
250   \else
251     1%
252   \fi
253   }{1}=0 %
254   \expandafter\@firstoftwo
255 \else
256   \expandafter\@secondoftwo
257 \fi
258 }%
259 }{%

```

`\IfLanguageName`

```

260 \def\IfLanguageName#1{%
261   \ifnum\IfLang@IfDefined{languageName}%
262     \pdf@strcmp{#1}{\languageName}%

```

```

263         }{1}=0 %
264     \expandafter\@firstoftwo
265     \else
266         \expandafter\@secondoftwo
267     \fi
268 }%
269 }

```

## 2.5 Check plausibility of \language

```

270 \begingroup\expandafter\expandafter\expandafter\endgroup
271 \expandafter\ifx\csname language\endcsname\relax
272 \else
273     \IfLanguagePatterns{\language}{}{%
274         \@PackageWarningNoLine{iflang}{%
275             Mismatch between \string\language\space
276             (patterns)\MessageBreak
277             and setting of \string\language
278         }%
279     }%
280 \fi

281 \IfLang@AtEnd%
282 </package>

```

## 3 Test

### 3.1 Catcode checks for loading

```

283 <*test1>

284 \catcode'\{=1 %
285 \catcode'\}=2 %
286 \catcode'\#=6 %
287 \catcode'\@=11 %
288 \expandafter\ifx\csname count\endcsname\relax
289     \countdef\count@=255 %
290 \fi

291 \expandafter\ifx\csname @gobble\endcsname\relax
292     \long\def\@gobble#1{%
293 \fi

294 \expandafter\ifx\csname @firstofone\endcsname\relax
295     \long\def\@firstofone#1{#1}%
296 \fi

297 \expandafter\ifx\csname loop\endcsname\relax
298     \expandafter\@firstofone
299 \else
300     \expandafter\@gobble
301 \fi

302 {%
303     \def\loop#1\repeat{%
304         \def\body{#1}%
305         \iterate
306     }%
307     \def\iterate{%
308         \body
309         \let\next\iterate
310     \else
311         \let\next\relax
312     \fi
313     \next
314 }%
315 \let\repeat=\fi

```



```

316 }%
317 \def\RestoreCatcodes{}
318 \count@=0 %
319 \loop
320   \edef\RestoreCatcodes{%
321     \RestoreCatcodes
322     \catcode\the\count@=\the\catcode\count@\relax
323   }%
324 \ifnum\count@<255 %
325   \advance\count@ 1 %
326 \repeat
327
328 \def\RangeCatcodeInvalid#1#2{%
329   \count@=#1\relax
330   \loop
331     \catcode\count@=15 %
332   \ifnum\count@<#2\relax
333     \advance\count@ 1 %
334   \repeat
335 }
336 \def\RangeCatcodeCheck#1#2#3{%
337   \count@=#1\relax
338   \loop
339     \ifnum#3=\catcode\count@
340     \else
341       \errmessage{%
342         Character \the\count@\space
343         with wrong catcode \the\catcode\count@\space
344         instead of \number#3%
345       }%
346     \fi
347   \ifnum\count@<#2\relax
348     \advance\count@ 1 %
349   \repeat
350 }
351 \def\space{ }
352 \expandafter\ifx\csname LoadCommand\endcsname\relax
353   \def\LoadCommand{\input iflang.sty\relax}%
354 \fi
355 \def\Test{%
356   \RangeCatcodeInvalid{0}{47}%
357   \RangeCatcodeInvalid{58}{64}%
358   \RangeCatcodeInvalid{91}{96}%
359   \RangeCatcodeInvalid{123}{255}%
360   \catcode'\@=12 %
361   \catcode'\=0 %
362   \catcode'\%=14 %
363   \LoadCommand
364   \RangeCatcodeCheck{0}{36}{15}%
365   \RangeCatcodeCheck{37}{37}{14}%
366   \RangeCatcodeCheck{38}{47}{15}%
367   \RangeCatcodeCheck{48}{57}{12}%
368   \RangeCatcodeCheck{58}{63}{15}%
369   \RangeCatcodeCheck{64}{64}{12}%
370   \RangeCatcodeCheck{65}{90}{11}%
371   \RangeCatcodeCheck{91}{91}{15}%
372   \RangeCatcodeCheck{92}{92}{0}%
373   \RangeCatcodeCheck{93}{96}{15}%
374   \RangeCatcodeCheck{97}{122}{11}%
375   \RangeCatcodeCheck{123}{255}{15}%
376   \RestoreCatcodes
377 }

```

```

378 \Test
379 \csname @@end\endcsname
380 \end
381 \</test1>

```

## 3.2 Test with L<sup>A</sup>T<sub>E</sub>X

```

382 \*test2 | test3>
383 \NeedsTeXFormat{LaTeX2e}
384 \test3\let\pdfstrcmp\relax
385 \nofiles
386 \documentclass{minimal}
387 \usepackage{qstest}
388 \IncludeTests{*}
389 \LogTests{log}{*}{*}
390 \usepackage[english,naustrian,ngerman]{babel}
391 \usepackage{iflang}
392 \begin{document}
393 \begin{qstest}\IfLanguagePatterns{language, pattern}
394   \def\test#1#2{%
395     \Expect*\IfLanguagePatterns{#1}{true}{false}}{#2}%
396   }%
397   \test{ngerman}{true}%
398   \test{naustrian}{true}%
399   \test{english}{false}%
400   \test{foobar}{false}%
401 \end{qstest}
402 \begin{qstest}\IfLanguageName{language, name}
403   \def\test#1#2{%
404     \Expect*\IfLanguageName{#1}{true}{false}}{#2}%
405   }%
406   \test{ngerman}{true}%
407   \test{naustrian}{false}%
408   \selectlanguage{naustrian}%
409   \test{ngerman}{false}%
410   \test{naustrian}{true}%
411   \test{foobar}{false}%
412   %
413   \def\language{naustrian}%
414   \test{naustrian}{true}%
415   \test{ngerman}{false}%
416   %
417   \edef\language{\string naustrian}%
418   \test{naustrian}{true}%
419   \test{ngerman}{false}%
420   %
421   \def\language{naustrian}%
422   \makeatletter
423   \@onelevel@sanitize\language
424   \test{naustrian}{true}%
425   \test{ngerman}{false}%
426   %
427   \def\language{naustrian}%
428   \def\xaustrian{naustrian}%
429   \def\xgerman{ngerman}%
430   \test{\xaustrian}{true}%
431   \test{\xgerman}{false}%
432   %
433   \def\language{\xaustrian}%
434   \test{naustrian}{true}%
435   \test{ngerman}{false}%
436   \test{\xaustrian}{true}%
437   \test{\xgerman}{false}%

```

```

438 \test{\language\name}{true}%
439 \test{\language\name\space}{false}%
440 %
441 \def\language\name{\empty\xaustrian\empty}%
442 \test{\naustrian}{true}%
443 \test{\ngerman}{false}%
444 \test{\empty\xaustrian\empty}{true}%
445 \test{\empty\xgerman\empty}{false}%
446 \end{qstest}
447 \begin{qstest}{IfDefined}{defined}
448 \makeatletter
449 \let\foobar\relax
450 \Expect*{\IfLang@IfDefined{foobar}{true}{false}}{false}%
451 \Expect*{\ifx\foobar\relax true\else false\fi}{true}%
452 \let\foobar\UNDEFINED
453 \Expect*{\IfLang@IfDefined{foobar}{true}{false}}{false}%
454 \Expect*{\ifx\foobar\relax true\else false\fi}{false}%
455 \Expect*{\ifx\foobar\UNDEFINED true\else false\fi}{true}%
456 \end{qstest}
457 \end{document}
458 </test2 | test3>

```

### 3.3 Test with plain T<sub>E</sub>X and $\epsilon$ -T<sub>E</sub>X

```

459 <*test4>
460 %% Format 'etex' based on 'language.def'
461 \input iflang.sty
462 \catcode64=12
463
464 \def\TestGeneric#1#2#3{%
465   \begingroup
466     \edef\x{#1{#2}}{true}{false}}%
467     \edef\y{#3}%
468     \ifx\x\y
469       \else
470         \errmessage{Failed test: \string#1{#2} <> #3}%
471       \fi
472   \endgroup
473 }
474 \def\TestPatterns{\TestGeneric\IfLanguagePatterns}
475 \def\TestName{\TestGeneric\IfLanguageName}
476
477 \TestPatterns{USenglish}{true}
478 \TestPatterns{ngerman}{false}
479
480 \TestName{USenglish}{true}
481 \TestName{ngerman}{false}
482
483 \uselanguage{ngerman}
484
485 \TestPatterns{USenglish}{false}
486 \TestPatterns{ngerman}{true}
487
488 \TestName{USenglish}{false}
489 \TestName{ngerman}{true}
490
491 \csname @@end\endcsname
492 \end
493 </test4>

```

### 3.4 Test with plain T<sub>E</sub>X and without $\epsilon$ -T<sub>E</sub>X/pdfT<sub>E</sub>X

```

494 <*test5>

```

```

495 %% Format 'tex' (vanilla plain-TeX)
496 \let\ifcsname\UNDEFINED
497 \let\pdfstrcmp\UNDEFINED
498 \input iflang.sty
499 \catcode64=11
500
501 \def\TestDefined#1{%
502   \IfLang@ifDefined{foobar}{-}{-}%
503   \ifx\foobar#1%
504   \else
505     \errmessage{Failed test: \string\foobar <> \string#1}%
506   \fi
507 }
508 \let\foobar\relax
509 \TestDefined\relax
510 \let\foobar\UNDEFINED
511 \TestDefined\relax
512
513 \def\strip@prefix#1>{}
514 \def@onelevel@sanitize#1{%
515   \edef#1{\expandafter\strip@prefix\meaning#1}%
516 }
517 \def\TestCompare#1#2#3{%
518   \begingroup
519     \edef\x{%
520       \if\IfLang@StrEqual{#1}{#2}%
521         true%
522       \else
523         false%
524       \fi
525     }%
526     \def\expect{#3}%
527     \ifx\x\expect
528     \else
529       \def\a{#1}%
530       \@onelevel@sanitize\a
531       \def\b{#2}%
532       \@onelevel@sanitize\b
533       \errmessage{Failed test: '\a'='\b' <> \expect}%
534     \fi
535   \endgroup
536 }
537 \TestCompare{junk}{junk}{true}
538 \TestCompare{}{}{true}
539 \TestCompare{a}{b}{false}
540 \TestCompare{aa}{bb}{false}
541 \def\a{ax}
542 \def\b{bx}
543 \def\c{\a\b}
544 \def\d{\c\b}
545 \def\exch#1#2{#2#1}
546 \def\gobble#1{}
547 \TestCompare{\gobble a}{-}{true}
548 \TestCompare{}{\gobble a}{true}
549 \TestCompare{a\exch xyb}{ayxb}{true}
550 \TestCompare{\c}{\c}{true}
551 \TestCompare{\d}{\c\b}{true}
552
553 \csname @@end\endcsname
554 \end
555 </test5>

```

## 4 Installation

### 4.1 Download

**Package.** This package is available on CTAN<sup>1</sup>:

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

*TDS* refers to the standard “A Directory Structure for T<sub>E</sub>X 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<sub>E</sub>X:

```
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
test/iflang-test1.tex → doc/latex/oberdiek/test/iflang-test1.tex
test/iflang-test2.tex → doc/latex/oberdiek/test/iflang-test2.tex
test/iflang-test3.tex → doc/latex/oberdiek/test/iflang-test3.tex
test/iflang-test4.tex → doc/latex/oberdiek/test/iflang-test4.tex
test/iflang-test5.tex → doc/latex/oberdiek/test/iflang-test5.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 T<sub>E</sub>X distribution (teT<sub>E</sub>X, mikT<sub>E</sub>X, ...) relies on file name databases, you must refresh these. For example, teT<sub>E</sub>X users run `texhash` or `mktextlsr`.

---

<sup>1</sup><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 `\language` 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.

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

Symbols	D
<code>\#</code> ..... 286	<code>\d</code> ..... 544, 551
<code>\%</code> ..... 362	<code>\documentclass</code> ..... 386
<code>\@</code> ..... 287, 360	
<code>\@PackageInfoNoLine</code> ... 159, 164, 194	
<code>\@PackageWarningNoLine</code> 177, 184, 274	
<code>\@firstofone</code> ..... 295, 298	
<code>\@firstoftwo</code> ..... <u>111</u> , 119, 128, 143, 206, 213, 254, 264	
<code>\@gobble</code> ..... 292, 300	
<code>\@onelevel@sanitize</code> 423, 514, 530, 532	
<code>\@secondoftwo</code> ..... <u>114</u> , 121, 126, 145, 208, 215, 256, 266	
<code>\@undefined</code> ..... 58	
<code>\</code> ..... 361	
<code>\{</code> ..... 284	
<code>\}</code> ..... 285	
	<b>E</b>
	<code>\empty</code> ..... 17, 18, 441, 444, 445
	<code>\end</code> .. 380, 401, 446, 456, 457, 492, 554
	<code>\endcsname</code> ..... 14, 21, 50, 66, 76, 111, 114, 118, 125, 133, 134, 150, 158, 174, 176, 193, 200, 212, 271, 288, 291, 294, 297, 352, 379, 491, 553
	<code>\endinput</code> ..... 29, 110
	<code>\endlinechar</code> ..... 4, 35, 71, 77, 89
	<code>\errmessage</code> ..... 341, 470, 505, 533
	<code>\exch</code> ..... 545, 549
	<code>\Expect</code> 395, 404, 450, 451, 453, 454, 455
	<code>\expect</code> ..... 526, 527, 533
	<b>F</b>
	<code>\foobar</code> ..... 449, 451, 452, 454, 455, 503, 505, 508, 510
	<b>G</b>
	<code>\gobble</code> ..... 546, 547, 548
	<b>I</b>
	<code>\if</code> ..... 229, 247, 520
	<code>\ifcat</code> ..... 226
	<code>\ifcsname</code> ..... 133, 496
	<code>\IfLang@@@StrExpand</code> ..... 240, 242
	<code>\IfLang@@StrExpand</code> ..... 237, 239
	<code>\IfLang@AtEnd</code> ..... 95, 96, 110, 281
	<code>\IfLang@ifDefined</code> ..... . <u>117</u> , 199, 246, 261, 450, 453, 502
	<code>\IfLang@OrgUseLanguage</code> .... 168, 171
	<code>\IfLang@prefix</code> ..... 157, 199, 200
	<code>\IfLang@StrEqual</code> ..... 219, 247, 520
	<code>\IfLang@StrEqualStart</code> ..... ..... 220, 222, 229, 231, 235, 243
	<code>\IfLang@StrEqualStop</code> ..... 224, 231
	<code>\IfLang@StrExpand</code> ..... 227, 235
	<code>\IfLang@StrNil</code> ..... . 218, 220, 223, 239, 240, 242, 243
	<code>\IfLanguageName</code> . 2, <u>245</u> , <u>260</u> , 404, 475
<b>A</b>	
<code>\a</code> ..... 529, 530, 533, 541, 543	
<code>\advance</code> ..... 325, 333, 348	
<code>\aftergroup</code> ..... 29	
<b>B</b>	
<code>\b</code> ... 531, 532, 533, 542, 543, 544, 551	
<code>\begin</code> ..... 392, 393, 402, 447	
<code>\body</code> ..... 304, 308	
<b>C</b>	
<code>\c</code> ..... 543, 544, 550, 551	
<code>\catcode</code> 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, 284, 285, 286, 287, 322, 331, 339, 343, 360, 361, 362, 462, 499	
<code>\count@</code> ..... 289, 318, 322, 324, 325, 329, 331, 332, 333, 337, 339, 342, 343, 347, 348	
<code>\countdef</code> ..... 289	
<code>\csname</code> 14, 21, 50, 66, 76, 111, 114, 118, 125, 134, 150, 158, 174, 176, 193, 200, 212, 271, 288, 291, 294, 297, 352, 379, 491, 553	

<code>\IfLanguagePatterns</code> .....	<code>\RangeCatcodeInvalid</code> .....
..... 2, 198, 273, 395, 474	..... 328, 356, 357, 358, 359
<code>\ifnum</code> .....	<code>\repeat</code> .....
200, 246, 261, 324, 332, 339, 347	303, 315, 326, 334, 349
<code>\ifx</code> .....	<code>\RequirePackage</code> .....
15, 18, 21, 50,	154, 155
58, 61, 111, 114, 118, 125, 134,	<code>\RestoreCatcodes</code> ..
150, 158, 174, 176, 193, 212,	317, 320, 321, 376
223, 271, 288, 291, 294, 297,	
352, 451, 454, 455, 468, 503, 527	<b>S</b>
<code>\immediate</code> .....	<code>\selectlanguage</code> .....
23, 52	408
<code>\IncludeTests</code> .....	<code>\space</code> .....
388	178,
<code>\input</code> .....	185, 195, 275, 342, 343, 351, 439
151, 152, 353, 461, 498	<code>\strip@prefix</code> .....
<code>\iterate</code> .....	513, 515
305, 307, 309	
<b>L</b>	<b>T</b>
<code>\lang@USenglish</code> .....	<code>\Test</code> .....
178, 181	355, 378
<code>\language</code> .....	<code>\test</code> .....
181, 200, 275	394, 397, 398, 399,
<code>\languageName</code> ..	400, 403, 406, 407, 409, 410,
170, 182, 185, 195,	411, 414, 415, 418, 419, 424,
248, 262, 273, 277, 413, 417,	425, 430, 431, 434, 435, 436,
421, 423, 427, 433, 438, 439, 441	437, 438, 439, 442, 443, 444, 445
<code>\LoadCommand</code> .....	<code>\TestCompare</code> .....
353, 363	517, 537, 538,
<code>\LogTests</code> .....	539, 540, 547, 548, 549, 550, 551
389	<code>\TestDefined</code> .....
<code>\loop</code> .....	501, 509, 511
303, 319, 330, 338	<code>\TestGeneric</code> .....
	464, 474, 475
	<code>\TestName</code> ....
	475, 480, 481, 488, 489
	<code>\TestPatterns</code> .
	474, 477, 478, 485, 486
	<code>\the</code> .....
	77, 78, 79,
	80, 81, 82, 83, 84, 97, 322, 342, 343
	<code>\TMP@EnsureCode</code> ....
	94, 101, 102,
	103, 104, 105, 106, 107, 108, 109
	<b>U</b>
<b>N</b>	<code>\UNDEFINED</code> ...
<code>\NeedsTeXFormat</code> .....	452, 455, 496, 497, 510
383	<code>\uselanguage</code> .....
<code>\next</code> .....	168, 169, 483
309, 311, 313	<code>\usepackage</code> .....
<code>\nofiles</code> .....	387, 390, 391
385	
<code>\number</code> .....	<b>W</b>
220, 344	<code>\write</code> .....
	23, 52
<b>P</b>	
<code>\PackageInfo</code> .....	<b>X</b>
26	<code>\x</code> .....
<code>\pdf@strcmp</code> .....	14, 15, 18, 22, 26, 28,
262	51, 56, 66, 75, 87, 466, 468, 519, 527
<code>\pdfstrcmp</code> .....	<code>\xaustrian</code> 428, 430, 433, 436, 441, 444
384, 497	<code>\xgerman</code> .....
<code>\ProvidesPackage</code> .....	429, 431, 437, 445
19, 67	
<b>R</b>	<b>Y</b>
<code>\RangeCatcodeCheck</code> .....	<code>\y</code> .....
.....	467, 468
336, 364, 365, 366, 367, 368,	
369, 370, 371, 372, 373, 374, 375	