

# 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 . . . . .	4
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> . . . . .	7
<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 . . . . .	9
3.3	Test with plain T <sub>E</sub> X and $\epsilon$ -T <sub>E</sub> X . . . . .	10
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>12</b>
4.1	Download . . . . .	12
4.2	Bundle installation . . . . .	12
4.3	Package installation . . . . .	13
4.4	Refresh file name databases . . . . .	13
4.5	Some details for the interested . . . . .	13
<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] . . . . .	14
	[2007/11/11 v1.5] . . . . .	14
<b>7</b>	<b>Index</b>	<b>14</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
3   \catcode44 12 % ,
4   \catcode45 12 % -
5   \catcode46 12 % .
6   \catcode58 12 % :
7   \catcode64 11 % @
8   \catcode123 1 % {
9   \catcode125 2 % }
10  \expandafter\let\expandafter\x\csname ver@iflang.sty\endcsname
11  \ifx\x\relax % plain-TEX, first loading
12  \else
13    \def\empty{}%
14    \ifx\x\empty % LATEX, first loading,
15    % variable is initialized, but \ProvidesPackage not yet seen
16    \else
17      \catcode35 6 % #
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      \aftergroup\endinput
27    \fi
28  \fi
29 \endgroup
```

Package identification:

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

```

60 \expandafter\x\csname ver@iflang.sty\endcsname
61 \ProvidesPackage{iflang}%
62 [2007/11/11 v1.5 Language checks (H0)]

63 \begingroup
64 \catcode123 1 % {
65 \catcode125 2 % }
66 \def\x{\endgroup
67 \expandafter\edef\csname IfLang@AtEnd\endcsname{%
68 \catcode35 \the\catcode35\relax
69 \catcode64 \the\catcode64\relax
70 \catcode123 \the\catcode123\relax
71 \catcode125 \the\catcode125\relax
72 }%
73 }%
74 \x
75 \catcode35 6 % #
76 \catcode64 11 % @
77 \catcode123 1 % {
78 \catcode125 2 % }
79 \def\TMP@EnsureCode#1#2{%
80 \edef\IfLang@AtEnd{%
81 \IfLang@AtEnd
82 \catcode#1 \the\catcode#1\relax
83 }%
84 \catcode#1 #2\relax
85 }
86 \TMP@EnsureCode{39}{12}% '
87 \TMP@EnsureCode{40}{12}% (
88 \TMP@EnsureCode{41}{12}% )
89 \TMP@EnsureCode{44}{12}% ,
90 \TMP@EnsureCode{46}{12}% .
91 \TMP@EnsureCode{47}{12}% /
92 \TMP@EnsureCode{58}{12}% :
93 \TMP@EnsureCode{61}{12}% =

```

## 2.2 Tools

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

\@firstoftwo

```

94 \expandafter\ifx\csname @firstoftwo\endcsname\relax
95 \long\def\@firstoftwo#1#2{#1}%
96 \fi

```

\@secondoftwo

```

97 \expandafter\ifx\csname @secondoftwo\endcsname\relax
98 \long\def\@secondoftwo#1#2{#2}%
99 \fi

```

### 2.2.2 Expandible existence check for macros

\IfLang@IfDefined

```

100 \begingroup\expandafter\expandafter\expandafter\endgroup
101 \expandafter\ifx\csname ifcsname\endcsname\relax
102 \expandafter\@firstoftwo
103 \else
104 \expandafter\@secondoftwo
105 \fi
106 {%
107 \def\IfLang@IfDefined#1{%
108 \expandafter\ifx\csname#1\endcsname\relax

```

```

109     \expandafter\@secondoftwo
110   \else
111     \expandafter\@firstoftwo
112   \fi
113 }%
114 }{
115   \def\IfLang@IfDefined#1{%
116     \ifnum\ifcsname#1\endcsname
117       \expandafter\ifx\csname#1\endcsname\relax
118         1%
119       \else
120         0%
121       \fi
122     \else
123       1%
124     \fi
125     =0 %
126     \expandafter\@firstoftwo
127   \else
128     \expandafter\@secondoftwo
129   \fi
130 }%
131 }

```

### 2.2.3 Macros for messages

```

132 \begingroup\expandafter\expandafter\expandafter\endgroup
133 \expandafter\ifx\csname RequirePackage\endcsname\relax
134   \input infwarerr.sty\relax
135   \input pdftexcmds.sty\relax
136 \else
137   \RequirePackage{infwarerr}[2007/09/09]%
138   \RequirePackage{pdftexcmds}[2007/11/11]%
139 \fi

```

### 2.2.4 Support for etex.src

\IfLang@prefix

```

140 \begingroup\expandafter\expandafter\expandafter\endgroup
141 \expandafter\ifx\csname uselanguage\endcsname\relax
142   \@PackageInfoNoLine{iflang}{%
143     Naming convention for patterns: babel%
144   }%
145   \def\IfLang@prefix{l@}%
146 \else
147   \@PackageInfoNoLine{iflang}{%
148     Naming convention for patterns: etex.src%
149   }%
150   \def\IfLang@prefix{lang@}%
151   \let\IfLang@OrgUseLanguage\uselanguage
152   \def\uselanguage#1{%
153     \edef\language{#1}%
154     \IfLang@OrgUseLanguage{#1}%
155   }%

```

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.

```

156 \begingroup\expandafter\expandafter\expandafter\endgroup
157 \expandafter\ifx\csname language\endcsname\relax
158   \begingroup\expandafter\expandafter\expandafter\endgroup

```

```

159 \expandafter\ifx\csname lang@USenglish\endcsname\relax
160 \PackageWarningNoLine{iflang}{%
161 \string\lang@USenglish\space is missing%
162 }%
163 \else
164 \ifnum\lang@USenglish=\language
165 \def\languageName{USenglish}%
166 \else
167 \PackageWarningNoLine{iflang}{%
168 \string\languageName\space is not set,\MessageBreak
169 current language is unknown%
170 }%
171 \fi
172 \fi
173 \fi
174 \fi
175 \begingroup\expandafter\expandafter\expandafter\endgroup
176 \expandafter\ifx\csname languageName\endcsname\relax
177 \PackageInfoNoLine{iflang}{%
178 \string\languageName\space is not set%
179 }%
180 \fi

```

## 2.3 \IfLanguagePatterns

\IfLanguagePatterns

```

181 \def\IfLanguagePatterns#1{%
182 \ifnum\IfLang@ifDefined{\IfLang@prefix#1}{%
183 \ifnum\csname\IfLang@prefix#1\endcsname=\language
184 0%
185 \else
186 1%
187 \fi
188 }{1}=0 %
189 \expandafter\@firstoftwo
190 \else
191 \expandafter\@secondoftwo
192 \fi
193 }

```

## 2.4 \IfLanguageName

```

194 \begingroup\expandafter\expandafter\expandafter\endgroup
195 \expandafter\ifx\csname pdf@strcmp\endcsname\relax
196 \expandafter\@firstoftwo
197 \else
198 \expandafter\@secondoftwo
199 \fi
200 {%

```

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

```

201 \def\IfLang@StrNil{\relax}%
202 \def\IfLang@StrEqual#1{%
203 \number\IfLang@StrEqualStart{#1}\IfLang@StrNil
204 }%

```

```

205 \def\IfLang@StrEqualStart#1#2#3{%
206   \ifx#3\IfLang@StrNil
207     \IfLang@StrEqualStop
208   \fi
209   \ifcat\noexpand#3\relax
210     \IfLang@StrExpand{#1}{#2}#3%
211   \fi
212   \IfLang@StrEqualStart{\ifx#3#1}{#2\fi}%
213 }%
214 \def\IfLang@StrEqualStop\fi#1\IfLang@StrEqualStart#2#3#4{%
215   \fi
216   #2#4\relax'#313 %
217 }%
218 \def\IfLang@StrExpand#1#2#3\fi\IfLang@StrEqualStart#4#5{%
219   \fi
220   \IfLang@@StrExpand{#1}{#2}#3%
221 }%
222 \def\IfLang@@StrExpand#1#2#3\IfLang@StrNil{%
223   \expandafter\IfLang@@@StrExpand#3\IfLang@StrNil{#1}{#2}%
224 }%
225 \def\IfLang@@@StrExpand#1\IfLang@StrNil#2#3{%
226   \IfLang@StrEqualStart{#2}{#3}#1\IfLang@StrNil
227 }%

\IfLanguageName

228 \def\IfLanguageName#1{%
229   \ifnum\IfLang@IfDefined{language}name}{%
230     \if\expandafter\IfLang@StrEqual\expandafter%
231       {\language}name}{#1}%
232     0%
233   \else
234     1%
235   \fi
236   }{1}=0 %
237   \expandafter\@firstoftwo
238 \else
239   \expandafter\@secondoftwo
240 \fi
241 }%

242 }{

\IfLanguageName

243 \def\IfLanguageName#1{%
244   \ifnum\IfLang@IfDefined{language}name}{%
245     \pdf@strcmp{#1}{\language}name}%
246     }{1}=0 %
247     \expandafter\@firstoftwo
248 \else
249   \expandafter\@secondoftwo
250 \fi
251 }%

252 }

```

## 2.5 Check plausibility of \language

```

253 \begingroup\expandafter\expandafter\expandafter\endgroup
254 \expandafter\ifx\csname language\endcsname\relax
255 \else
256   \IfLanguagePatterns{\language}{}%
257   \@PackageWarningNoLine{iflang}{%
258     Mismatch between \string\language\space

```

```

259      (patterns)\MessageBreak
260      and setting of \string\languagename
261    }%
262  }%
263 \fi

264 \IfLang@AtEnd
265 </package>

```

## 3 Test

### 3.1 Catcode checks for loading

```

266 <*test1>

267 \catcode'\{=1 %
268 \catcode'\}=2 %
269 \catcode'\#=6 %
270 \catcode'\@=11 %
271 \expandafter\ifx\csname count@\endcsname\relax
272   \countdef\count@=255 %
273 \fi
274 \expandafter\ifx\csname @gobble\endcsname\relax
275   \long\def\@gobble#1{}%
276 \fi
277 \expandafter\ifx\csname @firstofone\endcsname\relax
278   \long\def\@firstofone#1{#1}%
279 \fi
280 \expandafter\ifx\csname loop\endcsname\relax
281   \expandafter\@firstofone
282 \else
283   \expandafter\@gobble
284 \fi
285 {%
286   \def\loop#1\repeat{%
287     \def\body{#1}%
288     \iterate
289   }%
290   \def\iterate{%
291     \body
292     \let\next\iterate
293   \else
294     \let\next\relax
295   \fi
296   \next
297 }%
298 \let\repeat=\fi
299 }%
300 \def\RestoreCatcodes{}
301 \count@=0 %
302 \loop
303   \edef\RestoreCatcodes{%
304     \RestoreCatcodes
305     \catcode\the\count@=\the\catcode\count@\relax
306   }%
307 \ifnum\count@<255 %
308   \advance\count@ 1 %
309 \repeat
310
311 \def\RangeCatcodeInvalid#1#2{%
312   \count@=#1\relax
313   \loop
314     \catcode\count@=15 %
315   \ifnum\count@<#2\relax

```

```

316     \advance\count@ 1 %
317 \repeat
318 }
319 \expandafter\ifx\csname LoadCommand\endcsname\relax
320 \def\LoadCommand{\input iflang.sty\relax}%
321 \fi
322 \def\Test{%
323   \RangeCatcodeInvalid{0}{47}%
324   \RangeCatcodeInvalid{58}{64}%
325   \RangeCatcodeInvalid{91}{96}%
326   \RangeCatcodeInvalid{123}{255}%
327   \catcode'\@=12 %
328   \catcode'\=0 %
329   \catcode'\{=1 %
330   \catcode'\}=2 %
331   \catcode'\#=6 %
332   \catcode'\[=12 %
333   \catcode'\]=12 %
334   \catcode'\%=14 %
335   \catcode'\ =10 %
336   \catcode13=5 %
337   \LoadCommand
338   \RestoreCatcodes
339 }
340 \Test
341 \csname @@end\endcsname
342 \end
343 </test1>

```

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

```

344 <*test2 | test3>
345 \NeedsTeXFormat{LaTeX2e}
346 <test3>\let\pdfstrcmp\relax
347 \nofiles
348 \documentclass{minimal}
349 \usepackage{qstest}
350 \IncludeTests{*}
351 \LogTests{log}{*}{*}
352 \usepackage[english,naustrian,ngerman]{babel}
353 \usepackage{iflang}
354 \begin{document}
355 \begin{qstest}\IfLanguagePatterns{language, pattern}
356   \def\test#1#2{%
357     \Expect*{\IfLanguagePatterns{#1}{true}{false}}{#2}%
358   }%
359   \test{ngerman}{true}%
360   \test{naustrian}{true}%
361   \test{english}{false}%
362   \test{foobar}{false}%
363 \end{qstest}
364 \begin{qstest}\IfLanguageName{language, name}
365   \def\test#1#2{%
366     \Expect*{\IfLanguageName{#1}{true}{false}}{#2}%
367   }%
368   \test{ngerman}{true}%
369   \test{naustrian}{false}%
370   \selectlanguage{naustrian}%
371   \test{ngerman}{false}%
372   \test{naustrian}{true}%
373   \test{foobar}{false}%
374   %
375   \def\languagename{naustrian}%

```

```

376 \test{naustrian}{true}%
377 \test{ngerman}{false}%
378 %
379 \edef\language{\string naustrian}%
380 \test{naustrian}{true}%
381 \test{ngerman}{false}%
382 %
383 \def\language{naustrian}%
384 \makeatletter
385 \@onelevel@sanitize\language
386 \test{naustrian}{true}%
387 \test{ngerman}{false}%
388 %
389 \def\language{naustrian}%
390 \def\xaustrian{naustrian}%
391 \def\xgerman{ngerman}%
392 \test{\xaustrian}{true}%
393 \test{\xgerman}{false}%
394 %
395 \def\language{\xaustrian}%
396 \test{naustrian}{true}%
397 \test{ngerman}{false}%
398 \test{\xaustrian}{true}%
399 \test{\xgerman}{false}%
400 \test{\language}{true}%
401 \test{\language\space}{false}%
402 %
403 \def\language{\empty\xaustrian\empty}%
404 \test{naustrian}{true}%
405 \test{ngerman}{false}%
406 \test{\empty\xaustrian\empty}{true}%
407 \test{\empty\xgerman\empty}{false}%
408 \end{qstest}
409 \begin{qstest}\IfDefined{defined}
410 \makeatletter
411 \let\foobar\relax
412 \Expect*{\IfLang\IfDefined{foobar}{true}{false}}{false}%
413 \Expect*{\ifx\foobar\relax true\else false\fi}{true}%
414 \let\foobar\UNDEFINED
415 \Expect*{\IfLang\IfDefined{foobar}{true}{false}}{false}%
416 \Expect*{\ifx\foobar\relax true\else false\fi}{false}%
417 \Expect*{\ifx\foobar\UNDEFINED true\else false\fi}{true}%
418 \end{qstest}
419 \end{document}
420 </test2 | test3>

```

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

```

421 <*test4>
422 %% Format 'etex' based on 'language.def'
423 \input iflang.sty
424 \catcode64=12
425
426 \def\TestGeneric#1#2#3{%
427   \begingroup
428     \edef\x{#1{#2}{true}{false}}%
429     \edef\y{#3}%
430     \ifx\x\y
431     \else
432       \errmessage{Failed test: \string#1{#2} <> #3}%
433     \fi
434   \endgroup
435 }

```

```

436 \def\TestPatterns{\TestGeneric\IfLanguagePatterns}
437 \def\TestName{\TestGeneric\IfLanguageName}
438
439 \TestPatterns{USenglish}{true}
440 \TestPatterns{ngerman}{false}
441
442 \TestName{USenglish}{true}
443 \TestName{ngerman}{false}
444
445 \uselanguage{ngerman}
446
447 \TestPatterns{USenglish}{false}
448 \TestPatterns{ngerman}{true}
449
450 \TestName{USenglish}{false}
451 \TestName{ngerman}{true}
452
453 \csname @@end\endcsname
454 \end
455 </test4>

```

### 3.4 Test with plain TeX and without $\epsilon$ -TeX/pdfTeX

```

456 <*test5>
457 %% Format 'tex' (vanilla plain-TeX)
458 \let\ifcname\UNDEFINED
459 \let\pdfstrcmp\UNDEFINED
460 \input iflang.sty
461 \catcode64=11
462
463 \def\TestDefined#1{%
464   \IfLang@IfDefined{foobar}{-}{-}%
465   \ifx\foobar#1%
466   \else
467     \errmessage{Failed test: \string\foobar <> \string#1}%
468   \fi
469 }
470 \let\foobar\relax
471 \TestDefined\relax
472 \let\foobar\UNDEFINED
473 \TestDefined\relax
474
475 \def\strip@prefix#1>{}
476 \def\@onelevel@sanitize#1{%
477   \edef#1{\expandafter\strip@prefix\meaning#1}%
478 }
479 \def\TestCompare#1#2#3{%
480   \begingroup
481     \edef\x{%
482       \if\IfLang@StrEqual{#1}{#2}%
483       true%
484       \else
485       false%
486       \fi
487     }%
488   \def\expect{#3}%
489   \ifx\x\expect
490   \else
491     \def\a{#1}%
492     \@onelevel@sanitize\a
493     \def\b{#2}%
494     \@onelevel@sanitize\b
495     \errmessage{Failed test: '\a'='b' <> \expect}%

```

```

496     \fi
497 \endgroup
498 }
499 \TestCompare{junk}{junk}{true}
500 \TestCompare{}{}{true}
501 \TestCompare{a}{b}{false}
502 \TestCompare{aa}{bb}{false}
503 \def\ax{}
504 \def\bx{}
505 \def\c{\a\b}
506 \def\d{\c\b}
507 \def\exch#1#2{#2#1}
508 \def\gobble#1{}
509 \TestCompare{\gobble a}{}{true}
510 \TestCompare{}{\gobble a}{true}
511 \TestCompare{a\exch xyb}{ayxb}{true}
512 \TestCompare{\c}{\c}{true}
513 \TestCompare{\d}{\c\b}{true}
514
515 \csname @@end\endcsname
516 \end
517 </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  $\TeX$  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/
```

---

<sup>1</sup>[ftp://ftp.ctan.org/tex-archive/](http://ftp.ctan.org/tex-archive/)

### 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
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 `TEX` distribution (`teTEX`, `mikTEX`, ...) relies on file name databases, you must refresh these. For example, `teTEX` 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 L<sup>A</sup>T<sub>E</sub>X.** The `.dtx` chooses its action depending on the format:

**plain T<sub>E</sub>X:** Run `docstrip` and extract the files.

**L<sup>A</sup>T<sub>E</sub>X:** Generate the documentation.

If you insist on using L<sup>A</sup>T<sub>E</sub>X for `docstrip` (really, `docstrip` does not need L<sup>A</sup>T<sub>E</sub>X), 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<sup>A</sup>T<sub>E</sub>X:

```
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 `pdfdoccmds` 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		
<code>\#</code> .....	269, 331	<code>\{</code> ..... 267, 329
<code>\%</code> .....	334	<code>\}</code> ..... 268, 330
<code>\@</code> .....	270, 327	<code>\]</code> ..... 333
<code>\@PackageInfoNoLine</code> ...	142, 147, 177	
<code>\@PackageWarningNoLine</code>	160, 167, 257	<code>\_</code> ..... 335
<code>\@firstofone</code> .....	278, 281	
<code>\@firstoftwo</code> .....	94,	
	102, 111, 126, 189, 196, 237, 247	<b>A</b>
<code>\@gobble</code> .....	275, 283	<code>\a</code> ..... 491, 492, 495, 503, 505
<code>\@onelevel@sanitize</code>	385, 476, 492, 494	<code>\advance</code> ..... 308, 316
<code>\@secondoftwo</code> .....	97,	<code>\aftergroup</code> ..... 26
	104, 109, 128, 191, 198, 239, 249	
<code>\@undefined</code> .....	52	<b>B</b>
<code>\[</code> .....	332	<code>\b</code> ... 493, 494, 495, 504, 505, 506, 513
<code>\</code> .....	328	<code>\begin</code> ..... 354, 355, 364, 409
		<code>\body</code> ..... 287, 291

<b>C</b>	
<code>\c</code> .....	505, 506, 512, 513
<code>\catcode</code> .....	3, 4, 5, 6, 7, 8, 9, 17, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 64, 65, 68, 69, 70, 71, 75, 76, 77, 78, 82, 84, 267, 268, 269, 270, 305, 314, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 424, 461
<code>\count@</code> .....	272, 301, 305, 307, 308, 312, 314, 315, 316
<code>\countdef</code> .....	272
<code>\csname</code> .....	10, 18, 44, 60, 67, 94, 97, 101, 108, 117, 133, 141, 157, 159, 176, 183, 195, 254, 271, 274, 277, 280, 319, 341, 453, 515
<b>D</b>	
<code>\d</code> .....	506, 513
<code>\documentclass</code> .....	348
<b>E</b>	
<code>\empty</code> .....	13, 14, 403, 406, 407
<code>\end</code> ..	342, 363, 408, 418, 419, 454, 516
<code>\endcsname</code> .....	10, 18, 44, 60, 67, 94, 97, 101, 108, 116, 117, 133, 141, 157, 159, 176, 183, 195, 254, 271, 274, 277, 280, 319, 341, 453, 515
<code>\endinput</code> .....	26
<code>\errmessage</code> .....	432, 467, 495
<code>\exch</code> .....	507, 511
<code>\Expect</code> ..	357, 366, 412, 413, 415, 416, 417
<code>\expect</code> .....	488, 489, 495
<b>F</b>	
<code>\foobar</code> .....	411, 413, 414, 416, 417, 465, 467, 470, 472
<b>G</b>	
<code>\gobble</code> .....	508, 509, 510
<b>I</b>	
<code>\if</code> .....	212, 230, 482
<code>\ifcat</code> .....	209
<code>\ifcsname</code> .....	116, 458
<code>\IfLang@@@StrExpand</code> .....	223, 225
<code>\IfLang@@StrExpand</code> .....	220, 222
<code>\IfLang@AtEnd</code> .....	80, 81, 264
<code>\IfLang@IfDefined</code> .....	100, 182, 229, 244, 412, 415, 464
<code>\IfLang@OrgUseLanguage</code> .....	151, 154
<code>\IfLang@prefix</code> .....	140, 182, 183
<code>\IfLang@StrEqual</code> .....	202, 230, 482
<code>\IfLang@StrEqualStart</code> .....	203, 205, 212, 214, 218, 226
<code>\IfLang@StrEqualStop</code> .....	207, 214
<code>\IfLang@StrExpand</code> .....	210, 218
<code>\IfLang@StrNil</code> .....	201, 203, 206, 222, 223, 225, 226
<code>\IfLanguageName</code> ..	2, 228, 243, 366, 437
<code>\IfLanguagePatterns</code> .....	2, 181, 256, 357, 436
<code>\ifnum</code> .....	116, 164, 182, 183, 229, 244, 307, 315
<code>\ifx</code> .....	11, 14, 18, 44, 52, 55, 94, 97, 101, 108, 117, 133, 141, 157, 159, 176, 195, 206, 254, 271, 274, 277, 280, 319, 413, 416, 417, 430, 465, 489
<code>\immediate</code> .....	20, 46
<code>\IncludeTests</code> .....	350
<code>\input</code> .....	134, 135, 320, 423, 460
<code>\iterate</code> .....	288, 290, 292
<b>L</b>	
<code>\lang@USenglish</code> .....	161, 164
<code>\language</code> .....	164, 183, 258
<code>\languageName</code> ..	153, 165, 168, 178, 231, 245, 256, 260, 375, 379, 383, 385, 389, 395, 400, 401, 403
<code>\LoadCommand</code> .....	320, 337
<code>\LogTests</code> .....	351
<code>\loop</code> .....	286, 302, 313
<b>M</b>	
<code>\makeatletter</code> .....	384, 410
<code>\meaning</code> .....	477
<code>\MessageBreak</code> .....	168, 259
<b>N</b>	
<code>\NeedsTeXFormat</code> .....	345
<code>\next</code> .....	292, 294, 296
<code>\nofiles</code> .....	347
<code>\number</code> .....	203
<b>P</b>	
<code>\PackageInfo</code> .....	23
<code>\pdf@stricmp</code> .....	245
<code>\pdfstricmp</code> .....	346, 459
<code>\ProvidesPackage</code> .....	15, 61
<b>R</b>	
<code>\RangeCatcodeInvalid</code> .....	311, 323, 324, 325, 326
<code>\repeat</code> .....	286, 298, 309, 317
<code>\RequirePackage</code> .....	137, 138
<code>\RestoreCatcodes</code> ..	300, 303, 304, 338
<b>S</b>	
<code>\selectlanguage</code> .....	370
<code>\space</code> .....	161, 168, 178, 258, 401
<code>\strip@prefix</code> .....	475, 477
<b>T</b>	
<code>\Test</code> .....	322, 340
<code>\test</code> .....	356, 359, 360, 361, 362, 365, 368, 369, 371, 372, 373, 376, 377, 380, 381, 386, 387, 392, 393, 396, 397, 398, 399, 400, 401, 404, 405, 406, 407
<code>\TestCompare</code> .....	479, 499, 500, 501, 502, 509, 510, 511, 512, 513
<code>\TestDefined</code> .....	463, 471, 473
<code>\TestGeneric</code> .....	426, 436, 437
<code>\TestName</code> .....	437, 442, 443, 450, 451
<code>\TestPatterns</code> ..	436, 439, 440, 447, 448
<code>\the</code> .....	68, 69, 70, 71, 82, 305

<b>X</b>	
<code>\TMP@EnsureCode</code> .....	
. 79, 86, 87, 88, 89, 90, 91, 92, 93	<code>\x</code> ..... 10, 11, 14, 19, 23, 25,
	45, 50, 60, 66, 74, 428, 430, 481, 489
<b>U</b>	
<code>\UNDEFINED</code> ... 414, 417, 458, 459, 472	<code>\xaustrian</code> 390, 392, 395, 398, 403, 406
<code>\uselanguage</code> ..... 151, 152, 445	<code>\xgerman</code> ..... 391, 393, 399, 407
<code>\usepackage</code> ..... 349, 352, 353	
<b>Y</b>	
<b>W</b>	
<code>\write</code> ..... 20, 46	<code>\y</code> ..... 429, 430