

# The iflang package

Heiko Oberdiek  
<oberdiek@uni-freiburg.de>

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 $\varepsilon$ -T <sub>E</sub> X . . . . .	10
3.4	Test with plain-T <sub>E</sub> X and without $\varepsilon$ -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 . . . . .	12
4.4	Refresh file name databases . . . . .	13
4.5	Some details for the interested . . . . .	13
<b>5</b>	<b>Acknowledgement</b>	<b>13</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   \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    \catcode35 6 % #
19    \catcode123 1 % {
20    \catcode125 2 % }
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  \endgroup
30  \expandafter\endinput
31 \fi
32 \endgroup
```

Package identification:

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

```

60 [2007/11/11 v1.5 Language checks (H0)]
61 \begingroup
62 \catcode123 1 % {
63 \catcode125 2 % }
64 \def\x{\endgroup
65 \expandafter\edef\csname IfLang@AtEnd\endcsname{%
66 \catcode35 \the\catcode35\relax
67 \catcode64 \the\catcode64\relax
68 \catcode123 \the\catcode123\relax
69 \catcode125 \the\catcode125\relax
70 }%
71 }%
72 \x
73 \catcode35 6 % #
74 \catcode64 11 % @
75 \catcode123 1 % {
76 \catcode125 2 % }
77 \def\TMP@EnsureCode#1#2{%
78 \edef\IfLang@AtEnd{%
79 \IfLang@AtEnd
80 \catcode#1 \the\catcode#1\relax
81 }%
82 \catcode#1 #2\relax
83 }
84 \TMP@EnsureCode{39}{12}% '
85 \TMP@EnsureCode{40}{12}% (
86 \TMP@EnsureCode{41}{12}% )
87 \TMP@EnsureCode{44}{12}% ,
88 \TMP@EnsureCode{46}{12}% .
89 \TMP@EnsureCode{47}{12}% /
90 \TMP@EnsureCode{58}{12}% :
91 \TMP@EnsureCode{61}{12}% =

```

## 2.2 Tools

### 2.2.1 Provide some basic macros of $\text{\LaTeX}$

```

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

```

```

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

```

### 2.2.2 Expandible existence check for macros

```

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

```

```

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

```

### 2.2.3 Macros for messages

```

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

```

### 2.2.4 Support for etex.src

\IfLang@prefix

```

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

```

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.

```

154 \begingroup\expandafter\expandafter\expandafter\endgroup
155 \expandafter\ifx\csname language\endcsname\relax
156   \begingroup\expandafter\expandafter\expandafter\endgroup
157   \expandafter\ifx\csname lang@USenglish\endcsname\relax
158     \@PackageWarningNoLine{iflang}{%

```

```

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

```

## 2.3 \IfLanguagePatterns

\IfLanguagePatterns

```

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

```

## 2.4 \IfLanguageName

```

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

```

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

```

199 \def\IfLang@StrNil{\relax}%
200 \def\IfLang@StrEqual#1{%
201     \number\IfLang@StrEqualStart{#1}\IfLang@StrNil
202 }%
203 \def\IfLang@StrEqualStart#1#2#3{%
204     \ifx#3\IfLang@StrNil

```

```

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

\IfLanguageName

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

240 }{%

\IfLanguageName

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

250 }

```

## 2.5 Check plausibility of \languageName

```

251 \begingroup\expandafter\expandafter\expandafter\endgroup
252 \expandafter\ifx\csname languageName\endcsname\relax
253 \else
254     \IfLanguagePatterns{\languageName}{}%
255     \@PackageWarningNoLine{iflang}{%
256         Mismatch between \string\language\space
257         (patterns)\MessageBreak
258         and setting of \string\languageName

```

```

259     }%
260 }%
261 \fi

262 \IfLang@AtEnd
263 \end{package}

```

## 3 Test

### 3.1 Catcode checks for loading

```

264 (*test1)

265 \catcode'\{=1 %
266 \catcode'\}=2 %
267 \catcode'\#=6 %
268 \catcode'\@=11 %
269 \expandafter\ifx\csname count@\endcsname\relax
270   \countdef\count@=255 %
271 \fi

272 \expandafter\ifx\csname @gobble\endcsname\relax
273   \long\def\@gobble#1{}%
274 \fi

275 \expandafter\ifx\csname @firstofone\endcsname\relax
276   \long\def\@firstofone#1{#1}%
277 \fi

278 \expandafter\ifx\csname loop\endcsname\relax
279   \expandafter\@firstofone
280 \else
281   \expandafter\@gobble
282 \fi

283 {%
284   \def\loop#1\repeat{%
285     \def\body{#1}%
286     \iterate
287   }%
288   \def\iterate{%
289     \body
290     \let\next\iterate
291   \else
292     \let\next\relax
293   \fi
294   \next
295 }%
296 \let\repeat=\fi
297 }%

298 \def\RestoreCatcodes{}
299 \count@=0 %
300 \loop
301   \edef\RestoreCatcodes{%
302     \RestoreCatcodes
303     \catcode\the\count@=\the\catcode\count@\relax
304   }%
305 \ifnum\count@<255 %
306   \advance\count@ 1 %
307 \repeat
308
309 \def\RangeCatcodeInvalid#1#2{%
310   \count@=#1\relax
311   \loop
312     \catcode\count@=15 %
313   \ifnum\count@<#2\relax
314     \advance\count@ 1 %
315   \repeat

```



```

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

```

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

```

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

```

```

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

```

### 3.3 Test with plain-TeX and $\varepsilon$ -TeX

```

419 (*test4)
420 %% Format 'etex' based on 'language.def'
421 \input iflang.sty
422 \catcode64=12
423
424 \def\TestGeneric#1#2#3{%
425   \begingroup
426     \edef\x{#1{#2}{true}{false}}%
427     \edef\y{#3}%
428     \ifx\x\y
429       \else
430         \errmessage{Failed test: \string#1{#2} <> #3}%
431       \fi
432   \endgroup
433 }
434 \def\TestPatterns{\TestGeneric\IfLanguagePatterns}
435 \def\TestName{\TestGeneric\IfLanguageName}

```

```

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

```

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

```

454 (*test5)
455 %% Format 'tex' (vanilla plain-TeX)
456 \let\ifcsname\UNDEFINED
457 \let\pdfstrcmp\UNDEFINED
458 \input iflang.sty
459 \catcode64=11
460
461 \def\TestDefined#1{%
462   \IfLang@IfDefined{foobar}{}{}%
463   \ifx\foobar#1%
464     \else
465       \errmessage{Failed test: \string\foobar <> \string#1}%
466     \fi
467 }
468 \let\foobar\relax
469 \TestDefined\relax
470 \let\foobar\UNDEFINED
471 \TestDefined\relax
472
473 \def\strip@prefix#1>{}
474 \def@onelevel@sanitize#1{%
475   \edef#1{\expandafter\strip@prefix\meaning#1}%
476 }
477 \def\TestCompare#1#2#3{%
478   \begingroup
479     \edef\x{%
480       \if\IfLang@StrEqual{#1}{#2}%
481         true%
482       \else
483         false%
484       \fi
485     }%
486     \def\expect{#3}%
487     \ifx\x\expect
488       \else
489         \def\a{#1}%
490         \@onelevel@sanitize\a
491         \def\b{#2}%
492         \@onelevel@sanitize\b
493         \errmessage{Failed test: '\a'='b' <> \expect}%
494       \fi
495   \endgroup

```

```

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

---

<sup>1</sup><http://ftp.ctan.org/tex-archive/>

**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  $\LaTeX$ .** 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 `pdf texcmds` for L<sup>A</sup>T<sub>E</sub>X 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; numbers in roman refer to the code lines where the entry is used.

Symbols		A	
<code>\#</code>	267, 329	<code>\a</code>	489, 490, 493, 501, 503
<code>\%</code>	332	<code>\advance</code>	306, 314
<code>\@</code>	268, 325	B	
<code>\@PackageInfoNoLine</code>	140, 145, 175	<code>\b</code>	491, 492, 493, 502, 503, 504, 511
<code>\@PackageWarningNoLine</code>	158, 165, 255	<code>\begin</code>	352, 353, 362, 407
<code>\@firstofone</code>	276, 279	<code>\body</code>	285, 289
<code>\@firstoftwo</code>	<u>92</u> , 100, 109, 124, 187, 194, 235, 245	C	
<code>\@gobble</code>	273, 281	<code>\c</code>	503, 504, 510, 511
<code>\@onelevel@sanitize</code>	383, 474, 490, 492	<code>\catcode</code>	3, 4, 5, 6, 7, 18, 19, 20, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 62, 63, 66, 67, 68, 69, 73, 74, 75, 76, 80, 82, 265, 266, 267, 268, 303, 312, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 422, 459
<code>\@secondoftwo</code>	<u>95</u> , 102, 107, 126, 189, 196, 237, 247	<code>\count@</code>	270, 299, 303, 305, 306, 310, 312, 313, 314
<code>\[</code>	330	<code>\countdef</code>	270
<code>\]</code>	326	<code>\csname</code>	8, 21, 45, 58, 65, 92, 95, 99, 106, 115, 131, 139, 155,
<code>\{</code>	265, 327		
<code>\}</code>	266, 328		
<code>\]</code>	331		
<code>\_</code>	333		

157, 174, 181, 193, 252, 269, 272, 275, 278, 317, 339, 451, 513	\LoadCommand ..... 318, 335
	\LogTests ..... 349
	\loop ..... 284, 300, 311
<b>D</b>	<b>M</b>
\d ..... 504, 511	\makeatletter ..... 382, 408
\documentclass ..... 346	\meaning ..... 475
<b>E</b>	\MessageBreak ..... 166, 257
\empty ..... 12, 401, 404, 405	<b>N</b>
\end .. 340, 361, 406, 416, 417, 452, 514	\NeedsTeXFormat ..... 343
\endcsname 8, 21, 45, 58, 65, 92, 95, 99, 106, 114, 115, 131, 139, 155, 157, 174, 181, 193, 252, 269, 272, 275, 278, 317, 339, 451, 513	\next ..... 290, 292, 294
\endinginput ..... 30	\nofiles ..... 345
\errmessage ..... 430, 465, 493	\number ..... 201
\exch ..... 505, 509	<b>P</b>
\Expect 355, 364, 410, 411, 413, 414, 415	\PackageInfo ..... 26
\expect ..... 486, 487, 493	\pdf@stricmp ..... 243
<b>F</b>	\pdfstricmp ..... 344, 457
\foobar ..... 409, 411, 412, 414, 415, 463, 465, 468, 470	\ProvidesPackage ..... 59
<b>G</b>	<b>R</b>
\gobble ..... 506, 507, 508	\RangeCatcodeInvalid ..... 309, 321, 322, 323, 324
<b>I</b>	\repeat ..... 284, 296, 307, 315
\if ..... 210, 228, 480	\RequirePackage ..... 135, 136
\ifcase ..... 9	\RestoreCatcodes .. 298, 301, 302, 336
\ifcat ..... 207	<b>S</b>
\ifcsname ..... 114, 456	\selectlanguage ..... 368
\IfLang@@@StrExpand ..... 221, 223	\space ..... 159, 166, 176, 256, 399
\IfLang@@StrExpand ..... 218, 220	\strip@prefix ..... 473, 475
\IfLang@AtEnd ..... 78, 79, 262	<b>T</b>
\IfLang@IfDefined ..... .. 98, 180, 227, 242, 410, 413, 462	\Test ..... 320, 338
\IfLang@OrgUseLanguage ..... 149, 152	\test ..... 354, 357, 358, 359, 360, 363, 366, 367, 369, 370, 371, 374, 375, 378, 379, 384, 385, 390, 391, 394, 395, 396, 397, 398, 399, 402, 403, 404, 405
\IfLang@prefix ..... 138, 180, 181	\TestCompare ..... 477, 497, 498, 499, 500, 507, 508, 509, 510, 511
\IfLang@StrEqual ..... 200, 228, 480	\TestDefined ..... 461, 469, 471
\IfLang@StrEqualStart ..... ..... 201, 203, 210, 212, 216, 224	\TestGeneric ..... 424, 434, 435
\IfLang@StrEqualStop ..... 205, 212	\TestName .... 435, 440, 441, 448, 449
\IfLang@StrExpand ..... 208, 216	\TestPatterns . 434, 437, 438, 445, 446
\IfLang@StrNil ..... .. 199, 201, 204, 220, 221, 223, 224	\the ..... 66, 67, 68, 69, 80, 303
\IfLanguageName . 2, 226, 241, 364, 435	\TMP@EnsureCode ..... .. 77, 84, 85, 86, 87, 88, 89, 90, 91
\IfLanguagePatterns ..... ..... 2, 179, 254, 355, 434	<b>U</b>
\ifnum ..... 114, 162, 180, 181, 227, 242, 305, 313	\UNDEFINED ... 412, 415, 456, 457, 470
\ifx ..... 10, 12, 21, 45, 53, 92, 95, 99, 106, 115, 131, 139, 155, 157, 174, 193, 204, 252, 269, 272, 275, 278, 317, 411, 414, 415, 428, 463, 487	\uselanguage ..... 149, 150, 443
\immediate ..... 23, 47	\usepackage ..... 347, 350, 351
\IncludeTests ..... 348	<b>W</b>
\input ..... 132, 133, 318, 421, 458	\write ..... 23, 47
\iterate ..... 286, 288, 290	<b>X</b>
<b>L</b>	\x ..... 8, 10, 12, 22, 26, 28, 46, 51, 58, 64, 72, 426, 428, 479, 487
\lang@USenglish ..... 159, 162	\xaustrian 388, 390, 393, 396, 401, 404
\language ..... 162, 181, 256	\xgerman ..... 389, 391, 397, 405
\languageName .. 151, 163, 166, 176, 229, 243, 254, 258, 373, 377, 381, 383, 387, 393, 398, 399, 401	<b>Y</b>
	\y ..... 427, 428