

# 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 $\epsilon$ -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   \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@\undefined
54         \xdef#1{#3}%
55       \fi
56       \ifx#1\relax
57         \xdef#1{#3}%
58       \fi
59     }%
```

```

60 \fi
61 \expandafter\x\csname ver@iflang.sty\endcsname
62 \ProvidesPackage{iflang}%
63 [2007/11/11 v1.5 Language checks (HO)]
64 \begin{group}
65 \catcode123 1 % {
66 \catcode125 2 % }
67 \def\x{\endgroup
68 \expandafter\edef\csname IfLang@AtEnd\endcsname{%
69 \catcode35 \the\catcode35\relax
70 \catcode64 \the\catcode64\relax
71 \catcode123 \the\catcode123\relax
72 \catcode125 \the\catcode125\relax
73 }%
74 }%
75 \x
76 \catcode35 6 % #
77 \catcode64 11 % @
78 \catcode123 1 % {
79 \catcode125 2 % }
80 \def\TMP@EnsureCode#1#2{%
81 \edef\IfLang@AtEnd{%
82 \IfLang@AtEnd
83 \catcode#1 \the\catcode#1\relax
84 }%
85 \catcode#1 #2\relax
86 }
87 \TMP@EnsureCode{39}{12}% '
88 \TMP@EnsureCode{40}{12}% (
89 \TMP@EnsureCode{41}{12}% )
90 \TMP@EnsureCode{44}{12}% ,
91 \TMP@EnsureCode{46}{12}% .
92 \TMP@EnsureCode{47}{12}% /
93 \TMP@EnsureCode{58}{12}% :
94 \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
95 \expandafter\ifx\csname @firstoftwo\endcsname\relax
96 \long\def\@firstoftwo#1#2{#1}%
97 \fi

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

```

### 2.2.2 Expandible existence check for macros

```

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

```

```

109 \expandafter\ifx\csname#1\endcsname\relax
110 \expandafter\@secondoftwo
111 \else
112 \expandafter\@firstoftwo
113 \fi
114 }%
115 }%
116 \def\IfLang@ifdefined#1{%
117 \ifnum\ifcsname#1\endcsname
118 \expandafter\ifx\csname#1\endcsname\relax
119 1%
120 \else
121 0%
122 \fi
123 \else
124 1%
125 \fi
126 =0 %
127 \expandafter\@firstoftwo
128 \else
129 \expandafter\@secondoftwo
130 \fi
131 }%
132 }

```

### 2.2.3 Macros for messages

```

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

```

### 2.2.4 Support for etex.src

\IfLang@prefix

```

141 \begingroup\expandafter\expandafter\expandafter\endgroup
142 \expandafter\ifx\csname uselanguage\endcsname\relax
143 \@PackageInfoNoLine{iflang}{%
144 Naming convention for patterns: babel%
145 }%
146 \def\IfLang@prefix{l@}%
147 \else
148 \@PackageInfoNoLine{iflang}{%
149 Naming convention for patterns: etex.src%
150 }%
151 \def\IfLang@prefix{lang}%
152 \let\IfLang@OrgUseLanguage\uselanguage
153 \def\uselanguage#1{%
154 \edef\language{#1}%
155 \IfLang@OrgUseLanguage{#1}%
156 }%

```

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.

```

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

```

```

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

```

## 2.3 \IfLanguagePatterns

\IfLanguagePatterns

```

182 \def\IfLanguagePatterns#1{%
183 \ifnum\IfLang@ifDefined{\IfLang@prefix#1}{%
184 \ifnum\csname\IfLang@prefix#1\endcsname=\language
185 0%
186 \else
187 1%
188 \fi
189 }{1}=0 %
190 \expandafter\@firstoftwo
191 \else
192 \expandafter\@secondoftwo
193 \fi
194 }

```

## 2.4 \IfLanguageName

```

195 \begingroup\expandafter\expandafter\expandafter\endgroup
196 \expandafter\ifx\csname pdf@strcmp\endcsname\relax
197 \expandafter\@firstoftwo
198 \else
199 \expandafter\@secondoftwo
200 \fi
201 {%

```

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

```

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

```

```

205 }%
206 \def\IfLang@StrEqualStart#1#2#3{%
207   \ifx#3\IfLang@StrNil
208     \IfLang@StrEqualStop
209   \fi
210   \ifcat\noexpand#3\relax
211     \IfLang@StrExpand{#1}{#2}#3%
212   \fi
213   \IfLang@StrEqualStart{\if#3#1}{#2\fi}%
214 }%
215 \def\IfLang@StrEqualStop\fi#1\IfLang@StrEqualStart#2#3#4{%
216   \fi
217   #2#4\relax'#313 %
218 }%
219 \def\IfLang@StrExpand#1#2#3\fi\IfLang@StrEqualStart#4#5{%
220   \fi
221   \IfLang@@StrExpand{#1}{#2}#3%
222 }%
223 \def\IfLang@@StrExpand#1#2#3\IfLang@StrNil{%
224   \expandafter\IfLang@@@StrExpand#3\IfLang@StrNil{#1}{#2}%
225 }%
226 \def\IfLang@@@StrExpand#1\IfLang@StrNil#2#3{%
227   \IfLang@StrEqualStart{#2}{#3}#1\IfLang@StrNil
228 }%

\IfLanguageName

229 \def\IfLanguageName#1{%
230   \ifnum\IfLang@IfDefined{languageName}{%
231     \if\expandafter\IfLang@StrEqual\expandafter%
232       {\languageName}{#1}%
233       0%
234     \else
235       1%
236     \fi
237     }{1}=0 %
238     \expandafter\@firstoftwo
239   \else
240     \expandafter\@secondoftwo
241   \fi
242 }%

243 }{

\IfLanguageName

244 \def\IfLanguageName#1{%
245   \ifnum\IfLang@IfDefined{languageName}{%
246     \pdf@strcmp{#1}{\languageName}%
247     }{1}=0 %
248     \expandafter\@firstoftwo
249   \else
250     \expandafter\@secondoftwo
251   \fi
252 }%

253 }

```

## 2.5 Check plausibility of \languageName

```

254 \begingroup\expandafter\expandafter\expandafter\endgroup
255 \expandafter\ifx\csname languageName\endcsname\relax
256 \else
257   \IfLanguagePatterns{\languageName}{}%
258   \@PackageWarningNoLine{iflang}{%

```

```

259      Mismatch between \string\language\space
260      (patterns)\MessageBreak
261      and setting of \string\languagename
262    }%
263  }%
264 \fi

265 \IfLang@AtEnd
266 \</package>

```

## 3 Test

### 3.1 Catcode checks for loading

```

267 <*test1>

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

```



```

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

```

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

```

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

```

```

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

421 /test2 | test3

```

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

```

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

```

```

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

```

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

```

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

```

```

496      \errmessage{Failed test: '\a='\b' <> \expect}%
497      \fi
498    \endgroup
499  }
500  \TestCompare{junk}{junk}{true}
501  \TestCompare{}{}{true}
502  \TestCompare{a}{b}{false}
503  \TestCompare{aa}{bb}{false}
504  \def\a{ax}
505  \def\b{bx}
506  \def\c{\a\b}
507  \def\d{\c\b}
508  \def\exch#1#2{#2#1}
509  \def\gobble#1{}
510  \TestCompare{\gobble a}{}{true}
511  \TestCompare{}{\gobble a}{true}
512  \TestCompare{a\exch xyb}{ayxb}{true}
513  \TestCompare{\c}{\c}{true}
514  \TestCompare{\d}{\c\b}{true}
515
516  \csname @@end\endcsname
517  \end
518  </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/
```

---

<sup>1</sup><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- $\text{\TeX}$ :

```
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  $\text{\TeX}$  distribution (`te $\text{\TeX}$` , `mik $\text{\TeX}$` , ...) relies on file name databases, you must refresh these. For example, `te $\text{\TeX}$`  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  $\text{\LaTeX}$ .** The `.dtx` chooses its action depending on the format:

**plain- $\text{\TeX}$ :** Run `docstrip` and extract the files.

**$\text{\LaTeX}$ :** Generate the documentation.

If you insist on using  $\text{\LaTeX}$  for `docstrip` (really, `docstrip` does not need  $\text{\LaTeX}$ ), 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 $\text{\LaTeX}$` :

```
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 `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; numbers in roman refer to the code lines where the entry is used.

Symbols		
<code>\#</code> .....	270, 332	<code>\{</code> ..... 268, 330
<code>\%</code> .....	335	<code>\}</code> ..... 269, 331
<code>\@</code> .....	271, 328	<code>\]</code> ..... 334
<code>\@PackageInfoNoLine</code> ...	143, 148, 178	
<code>\@PackageWarningNoLine</code>	161, 168, 258	<code>\_</code> ..... 336
<code>\@firstofone</code> .....	279, 282	
<code>\@firstoftwo</code> .....	95,	
	103, 112, 127, 190, 197, 238, 248	<b>A</b>
<code>\@gobble</code> .....	276, 284	<code>\a</code> ..... 492, 493, 496, 504, 506
<code>\@onelevel@sanitize</code>	386, 477, 493, 495	<code>\advance</code> ..... 309, 317
<code>\@secondoftwo</code> .....	98,	
	105, 110, 129, 192, 199, 240, 250	<b>B</b>
<code>\@undefined</code> .....	53	<code>\b</code> ... 494, 495, 496, 505, 506, 507, 514
<code>\[</code> .....	333	<code>\begin</code> ..... 355, 356, 365, 410
<code>\</code> .....	329	<code>\body</code> ..... 288, 292

<b>C</b>	
<code>\c</code> .....	506, 507, 513, 514
<code>\catcode</code> .....	3, 4, 5, 6, 7, 18, 19, 20, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 65, 66, 69, 70, 71, 72, 76, 77, 78, 79, 83, 85, 268, 269, 270, 271, 306, 315, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 425, 462
<code>\count@</code> .....	273, 302, 306, 308, 309, 313, 315, 316, 317
<code>\countdef</code> .....	273
<code>\csname</code> .....	8, 21, 45, 61, 68, 95, 98, 102, 109, 118, 134, 142, 158, 160, 177, 184, 196, 255, 272, 275, 278, 281, 320, 342, 454, 516
<b>D</b>	
<code>\d</code> .....	507, 514
<code>\documentclass</code> .....	349
<b>E</b>	
<code>\empty</code> .....	12, 404, 407, 408
<code>\end</code> ..	343, 364, 409, 419, 420, 455, 517
<code>\endcsname</code> .....	. 8, 21, 45, 61, 68, 95, 98, 102, 109, 117, 118, 134, 142, 158, 160, 177, 184, 196, 255, 272, 275, 278, 281, 320, 342, 454, 516
<code>\endinput</code> .....	30
<code>\errmessage</code> .....	433, 468, 496
<code>\exch</code> .....	508, 512
<code>\Expect</code> ..	358, 367, 413, 414, 416, 417, 418
<code>\expect</code> .....	489, 490, 496
<b>F</b>	
<code>\foobar</code> .....	412, 414, 415, 417, 418, 466, 468, 471, 473
<b>G</b>	
<code>\gobble</code> .....	509, 510, 511
<b>I</b>	
<code>\if</code> .....	213, 231, 483
<code>\ifcase</code> .....	9
<code>\ifcat</code> .....	210
<code>\ifcsname</code> .....	117, 459
<code>\IfLang@@@StrExpand</code> .....	224, 226
<code>\IfLang@@StrExpand</code> .....	221, 223
<code>\IfLang@AtEnd</code> .....	81, 82, 265
<code>\IfLang@IfDefined</code> .....	. 101, 183, 230, 245, 413, 416, 465
<code>\IfLang@OrgUseLanguage</code> ....	152, 155
<code>\IfLang@prefix</code> .....	141, 183, 184
<code>\IfLang@StrEqual</code> .....	203, 231, 483
<code>\IfLang@StrEqualStart</code> .....	.... 204, 206, 213, 215, 219, 227
<code>\IfLang@StrEqualStop</code> .....	208, 215
<code>\IfLang@StrExpand</code> .....	211, 219
<code>\IfLang@StrNil</code> .....	. 202, 204, 207, 223, 224, 226, 227
<code>\IfLanguageName</code> ..	2, 229, 244, 367, 438
<code>\IfLanguagePatterns</code> .....	..... 2, 182, 257, 358, 437
<code>\ifnum</code> .....	117, 165, 183, 184, 230, 245, 308, 316
<code>\ifx</code> .....	10, 12, 21, 45, 53, 56, 95, 98, 102, 109, 118, 134, 142, 158, 160, 177, 196, 207, 255, 272, 275, 278, 281, 320, 414, 417, 418, 431, 466, 490
<code>\immediate</code> .....	23, 47
<code>\IncludeTests</code> .....	351
<code>\input</code> .....	135, 136, 321, 424, 461
<code>\iterate</code> .....	289, 291, 293
<b>L</b>	
<code>\lang@USenglish</code> .....	162, 165
<code>\language</code> .....	165, 184, 259
<code>\languageName</code> ..	154, 166, 169, 179, 232, 246, 257, 261, 376, 380, 384, 386, 390, 396, 401, 402, 404
<code>\LoadCommand</code> .....	321, 338
<code>\LogTests</code> .....	352
<code>\loop</code> .....	287, 303, 314
<b>M</b>	
<code>\makeatletter</code> .....	385, 411
<code>\meaning</code> .....	478
<code>\MessageBreak</code> .....	169, 260
<b>N</b>	
<code>\NeedsTeXFormat</code> .....	346
<code>\next</code> .....	293, 295, 297
<code>\nofiles</code> .....	348
<code>\number</code> .....	204
<b>P</b>	
<code>\PackageInfo</code> .....	26
<code>\pdf@stricmp</code> .....	246
<code>\pdfstricmp</code> .....	347, 460
<code>\ProvidesPackage</code> .....	62
<b>R</b>	
<code>\RangeCatcodeInvalid</code> .....	..... 312, 324, 325, 326, 327
<code>\repeat</code> .....	287, 299, 310, 318
<code>\RequirePackage</code> .....	138, 139
<code>\RestoreCatcodes</code> ..	301, 304, 305, 339
<b>S</b>	
<code>\selectlanguage</code> .....	371
<code>\space</code> .....	162, 169, 179, 259, 402
<code>\strip@prefix</code> .....	476, 478
<b>T</b>	
<code>\Test</code> .....	323, 341
<code>\test</code> .....	357, 360, 361, 362, 363, 366, 369, 370, 372, 373, 374, 377, 378, 381, 382, 387, 388, 393, 394, 397, 398, 399, 400, 401, 402, 405, 406, 407, 408
<code>\TestCompare</code> .....	480, 500, 501, 502, 503, 510, 511, 512, 513, 514
<code>\TestDefined</code> .....	464, 472, 474

<code>\TestGeneric</code> . . . . .	427, 437, 438				<b>W</b>
<code>\TestName</code> . . . .	438, 443, 444, 451, 452	<code>\write</code> . . . . .	23, 47		
<code>\TestPatterns</code> .	437, 440, 441, 448, 449				<b>X</b>
<code>\the</code> . . . . .	69, 70, 71, 72, 83, 306	<code>\x</code> . . . . .	8, 10, 12, 22, 26, 28,		
<code>\TMP@EnsureCode</code> . . . . .			46, 51, 61, 67, 75, 429, 431, 482, 490		
. . . . .	80, 87, 88, 89, 90, 91, 92, 93, 94	<code>\xaustrian</code>	391, 393, 396, 399, 404, 407		
		<code>\xgerman</code> . . . . .	392, 394, 400, 408		
					<b>Y</b>
<b>U</b>		<code>\y</code> . . . . .	430, 431		
<code>\UNDEFINED</code> . . .	415, 418, 459, 460, 473				
<code>\uselanguage</code> . . . . .	152, 153, 446				
<code>\usepackage</code> . . . . .	350, 353, 354				