

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

1	Documentation	2
2	Implementation	2
2.1	Reload check and package identification	3
2.2	Tools	4
2.2.1	Provide some basic macros of L ^A T _E X	4
2.2.2	Expandible existence check for macros	4
2.2.3	Macros for messages	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
3	Test	8
3.1	Catcode checks for loading	8
3.2	Test with L ^A T _E X	9
3.3	Test with plain-T _E X and ε -T _E X	10
3.4	Test with plain-T _E X and without ε -T _E X/pdfT _E X	11
4	Installation	12
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
5	Acknowledgement	13
6	History	14
	[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
7	Index	14

1 Documentation

Package **babel** defines `\iflanguage`. As first argument it takes a language name and executes the second or third argument depending on the current language. This language test is based on hyphenation patterns. However, it is possible that different languages or dialects share the same patterns. In such cases `\iflanguage` fails.

However, package **babel** and some other packages such as **german** or **ngerman** store the language name in the macro `\language` if `\selectlanguage` is called.

`\IfLanguageName {<lang>} {<then>} {<else>}`

Makro `\IfLanguageName` compares language `<lang>` with the current setting of macro `\language`. If both contains the same name then the `<then>` part is called, otherwise the `<else>` part.

The macro is expandable. Thus it can be safely used inside `\edef` or `\csname`. If case of errors like an undefined `\language` the `<else>` part is executed.

Note: Macro `\IfLanguageName` relies on the fact, that `\language` is set correctly:

Package **babel**:

Full support of `\language` in its language switching commands.

Format based on **babel** (`language.dat`):

If package **babel** is not used (or not yet loaded), then **babel**'s `hyphen.cfg` has set `\language` to the last language in `language.dat`, but `\language` (current patterns) is zero and points to the first language. Thus the value of `\language` is basically garbage. Package **iflang** warns if `\language` and `\language` do not fit. This can be fixed by loading package **babel** previously.

Format based on ϵ -**TeX**'s `etex.src` (`language.def`):

Unhappily it does not support `\language`. Thus this package hooks into `\uselanguage` to get `\language` defined and updated there. At package loading time the changed `\uselanguage` has not been called yet. Thus package **iflang** tries `USenglish`. This is the definite default language of `etex.src`. If the current patterns suit this default language, an undefined `\language` is set to this language. Otherwise a `\language` remains undefined and a warning is given.

`\IfLanguagePatterns {<lang>} {<then>} {<else>}`

This macro behaves similar to `\IfLanguageName`. But the language test is based on the current pattern in force (`\language`). Also this macro is expandable, in case of errors the `<else>` part is called.

The following naming convention for the pattern are supported:

babel/`language.dat` : `\l@<language>`

etex.src/`language.def` : `\lang@<language>`

Package **iflang** looks for `\uselanguage` (defined in `etex.src`) to find out the naming convention in use.

2 Implementation

1 `*package`

2.1 Reload check and package identification

Reload check, especially if the package is not used with L^AT_EX.

```
2 \begingroup
3   \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 L^AT_EX

```

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

```

3.2 Test with L^AT_EX

```

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

```

```

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

```

3.3 Test with plain-TeX and ε -TeX

```

416 (*test4)
417 %% Format 'etex' based on 'language.def'
418 \input iflang.sty
419 \catcode64=12
420
421 \def\TestGeneric#1#2#3{%
422   \begingroup
423     \edef\x{#1#2}{true}{false}}%
424     \edef\y{#3}%
425     \ifx\x\y
426     \else
427       \errmessage{Failed test: \string#1{#2} <> #3}%
428     \fi
429   \endgroup
430 }
431 \def\TestPatterns{\TestGeneric\IfLanguagePatterns}
432 \def\TestName{\TestGeneric\IfLanguageName}
433
434 \TestPatterns{USenglish}{true}
435 \TestPatterns{ngerman}{false}

```

```

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

```

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

```

451 <*test5>
452 %% Format 'tex' (vanilla plain-TeX)
453 \let\ifcsname\UNDEFINED
454 \let\pdfstrcmp\UNDEFINED
455 \input iflang.sty
456 \catcode64=11
457
458 \def\TestDefined#1{%
459   \IfLang@IfDefined{foobar}{-}%
460   \ifx\foobar#1%
461   \else
462     \errmessage{Failed test: \string\foobar <> \string#1}%
463   \fi
464 }
465 \let\foobar\relax
466 \TestDefined\relax
467 \let\foobar\UNDEFINED
468 \TestDefined\relax
469
470 \def\strip@prefix#1>{}
471 \def\@onelevel@sanitize#1{%
472   \edef#1{\expandafter\strip@prefix\meaning#1}%
473 }
474 \def\TestCompare#1#2#3{%
475   \begingroup
476     \edef\x{%
477       \if\IfLang@StrEqual{#1}{#2}%
478       true%
479       \else
480       false%
481       \fi
482     }%
483     \def\expect{#3}%
484     \ifx\x\expect
485     \else
486       \def\a{#1}%
487       \@onelevel@sanitize\a
488       \def\b{#2}%
489       \@onelevel@sanitize\b
490       \errmessage{Failed test: '\a'='b' <> \expect}%
491     \fi
492   \endgroup
493 }
494 \TestCompare{junk}{junk}{true}
495 \TestCompare{}{}{true}

```

```

496 \TestCompare{a}{b}{false}
497 \TestCompare{aa}{bb}{false}
498 \def\ax{
499 \def\bx{
500 \def\c{\a\b}
501 \def\d{\c\b}
502 \def\exch#1#2{#2#1}
503 \def\gobble#1{}
504 \TestCompare{\gobble a}{}{true}
505 \TestCompare{}{\gobble a}{true}
506 \TestCompare{a\exch xyb}{ayxb}{true}
507 \TestCompare{\c}{\c}{true}
508 \TestCompare{\d}{\c\b}{true}
509
510 \csname @@end\endcsname
511 \end
512 </test5>

```

4 Installation

4.1 Download

Package. This package is available on CTAN¹:

[CTAN:macros/latex/contrib/oberdiek/iflang.dtx](#) The source file.

[CTAN:macros/latex/contrib/oberdiek/iflang.pdf](#) Documentation.

Bundle. All the packages of the bundle ‘oberdiek’ are also available in a TDS compliant ZIP archive. There the packages are already unpacked and the documentation files are generated. The files and directories obey the TDS standard.

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

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/

```

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
```

¹<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 $\text{T}_{\text{E}}\text{X}$ distribution (`te $\text{T}_{\text{E}}\text{X}$` , `mik $\text{T}_{\text{E}}\text{X}$` , ...) relies on file name databases, you must refresh these. For example, `te $\text{T}_{\text{E}}\text{X}$` 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 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$. The `.dtx` chooses its action depending on the format:

plain- $\text{T}_{\text{E}}\text{X}$: Run `docstrip` and extract the files.

$\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$: Generate the documentation.

If you insist on using $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ for `docstrip` (really, `docstrip` does not need $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$), 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 `pdf $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$` :

```
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 `pdf texcmds` for L^AT_EX 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, 326	<code>\a</code>	486, 487, 490, 498, 500
<code>\%</code>	329	<code>\advance</code>	306, 314
<code>\@</code>	268, 322	B	
<code>\@PackageInfoNoLine</code>	140, 145, 175	<code>\b</code>	488, 489, 490, 499, 500, 501, 508
<code>\@PackageWarningNoLine</code>	158, 165, 255	<code>\begin</code>	349, 350, 359, 404
<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>	500, 501, 507, 508
<code>\@onelevel@sanitize</code>	380, 471, 487, 489	<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, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 419, 456
<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>\[</code>	327	<code>\countdef</code>	270
<code>\]</code>	323	<code>\csname</code>	8, 21, 45, 58, 65, 92, 95, 99, 106, 115, 131, 139,
<code>\{</code>	265, 324		
<code>\}</code>	266, 325		
<code>\]</code>	328		
<code>_</code>	330		

155, 157, 174, 181, 193, 252, 269, 272, 275, 278, 336, 448, 510	\LogTests 346 \loop 284, 300, 311
D	M
\d 501, 508 \documentclass 343	\makeatletter 379, 405 \meaning 472 \MessageBreak 166, 257
E	N
\empty 12, 398, 401, 402 \end .. 337, 358, 403, 413, 414, 449, 511 \endcsname ... 8, 21, 45, 58, 65, 92, 95, 99, 106, 114, 115, 131, 139, 155, 157, 174, 181, 193, 252, 269, 272, 275, 278, 336, 448, 510 \endinginput 30 \errmessage 427, 462, 490 \exch 502, 506 \Expect 352, 361, 407, 408, 410, 411, 412 \expect 483, 484, 490	\NeedsTeXFormat 340 \next 290, 292, 294 \nofiles 342 \number 201
F	P
\foobar 406, 408, 409, 411, 412, 460, 462, 465, 467	\PackageInfo 26 \pdf@strcmp 243 \pdfstrcmp 341, 454 \ProvidesPackage 59
G	R
\gobble 503, 504, 505	\RangeCatcodeInvalid 309, 318, 319, 320, 321 \repeat 284, 296, 307, 315 \RequirePackage 135, 136 \RestoreCatcodes .. 298, 301, 302, 333
I	S
\if 210, 228, 477 \ifcase 9 \ifcat 207 \ifcsname 114, 453 \IfLang@@@StrExpand 221, 223 \IfLang@@StrExpand 218, 220 \IfLang@AtEnd 78, 79, 262 \IfLang@IfDefined 98, 180, 227, 242, 407, 410, 459 \IfLang@OrgUseLanguage 149, 152 \IfLang@prefix 138, 180, 181 \IfLang@StrEqual 200, 228, 477 \IfLang@StrEqualStart 201, 203, 210, 212, 216, 224 \IfLang@StrEqualStop 205, 212 \IfLang@StrExpand 208, 216 \IfLang@StrNil 199, 201, 204, 220, 221, 223, 224 \IfLanguageName . 2, 226, 241, 361, 432 \IfLanguagePatterns 2, 179, 254, 352, 431 \ifnum 114, 162, 180, 181, 227, 242, 305, 313 \ifx 10, 12, 21, 45, 53, 92, 95, 99, 106, 115, 131, 139, 155, 157, 174, 193, 204, 252, 269, 272, 275, 278, 408, 411, 412, 425, 460, 484 \immediate 23, 47 \IncludeTests 345 \input 132, 133, 332, 418, 455 \iterate 286, 288, 290	\selectlanguage 365 \space 159, 166, 176, 256, 396 \strip@prefix 470, 472
L	T
\lang@USenglish 159, 162 \language 162, 181, 256 \languageName .. 151, 163, 166, 176, 229, 243, 254, 258, 370, 374, 378, 380, 384, 390, 395, 396, 398	\Test 317, 335 \test 351, 354, 355, 356, 357, 360, 363, 364, 366, 367, 368, 371, 372, 375, 376, 381, 382, 387, 388, 391, 392, 393, 394, 395, 396, 399, 400, 401, 402 \TestCompare 474, 494, 495, 496, 497, 504, 505, 506, 507, 508 \TestDefined 458, 466, 468 \TestGeneric 421, 431, 432 \TestName 432, 437, 438, 445, 446 \TestPatterns . 431, 434, 435, 442, 443 \the 66, 67, 68, 69, 80, 303 \TMP@EnsureCode 77, 84, 85, 86, 87, 88, 89, 90, 91
U	W
\UNDEFINED ... 409, 412, 453, 454, 467 \uselanguage 149, 150, 440 \usepackage 344, 347, 348	\write 23, 47
X	Y
\x 8, 10, 12, 22, 26, 28, 46, 51, 58, 64, 72, 423, 425, 476, 484 \xaustrian 385, 387, 390, 393, 398, 401 \xgerman 386, 388, 394, 402	\y 424, 425