

The iflang package

Heiko Oberdiek
<oberdiek@uni-freiburg.de>

2007/09/09 v1.4

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	7
3.1	Catcode checks for loading	7
3.2	Test with L ^A T _E X	8
3.3	Test with plain-T _E X and ε -T _E X	9
3.4	Test with plain-T _E X and without ε -T _E X/pdfT _E X	10
4	Installation	11
4.1	Download	11
4.2	Bundle installation	11
4.3	Package installation	12
4.4	Refresh file name databases	12
4.5	Some details for the interested	12
5	Acknowledgement	13
6	History	13
	[2007/04/10 v1.0]	13
	[2007/04/11 v1.1]	13
	[2007/04/12 v1.2]	13
	[2007/04/26 v1.3]	13
	[2007/09/09 v1.4]	13
7	Index	13

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    \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  \endgroup
27  \expandafter\endinput
28 \fi
29 \endgroup
```

Package identification:

```
30 \begingroup
31   \catcode40 12 % (
32   \catcode41 12 % )
33   \catcode44 12 % ,
34   \catcode45 12 % -
35   \catcode46 12 % .
36   \catcode47 12 % /
37   \catcode58 12 % :
38   \catcode64 11 % @
39   \expandafter\ifx\csname ProvidesPackage\endcsname\relax
40     \def\x#1#2#3[#4]{\endgroup
41       \immediate\write-1{Package: #3 #4}%
42       \xdef#1{#4}%
43     }%
44   \else
45     \def\x#1#2[#3]{\endgroup
46       #2[#{#3}]%
47       \ifx#1\relax
48         \xdef#1{#3}%
49       \fi
50     }%
51   \fi
52   \expandafter\x\csname ver@iflang.sty\endcsname
53   \ProvidesPackage{iflang}%
54   [2007/09/09 v1.4 Language checks (HO)]
55   \expandafter\edef\csname IfLang@AtEnd\endcsname{%
56     \catcode64 \the\catcode64\relax
57   }
58   \catcode64 11 % @
59   \def\TMP@EnsureCode#1#2{%
```

```

60 \edef\IfLang@AtEnd{%
61   \IfLang@AtEnd
62   \catcode#1 \the\catcode#1\relax
63 }%
64 \catcode#1 #2\relax
65 }
66 \TMP@EnsureCode{39}{12}% '
67 \TMP@EnsureCode{40}{12}% (
68 \TMP@EnsureCode{41}{12}% )
69 \TMP@EnsureCode{44}{12}% ,
70 \TMP@EnsureCode{46}{12}% .
71 \TMP@EnsureCode{47}{12}% /
72 \TMP@EnsureCode{58}{12}% :
73 \TMP@EnsureCode{61}{12}% =

```

2.2 Tools

2.2.1 Provide some basic macros of L^AT_EX

```

\@firstoftwo
74 \expandafter\ifx\csname @firstoftwo\endcsname\relax
75 \long\def\@firstoftwo#1#2{#1}%
76 \fi

\@secondoftwo
77 \expandafter\ifx\csname @secondoftwo\endcsname\relax
78 \long\def\@secondoftwo#1#2{#2}%
79 \fi

```

2.2.2 Expandible existence check for macros

```

\IfLang@IfDefined
80 \begingroup\expandafter\expandafter\expandafter\endgroup
81 \expandafter\ifx\csname ifcsname\endcsname\relax
82   \expandafter\@firstoftwo
83 \else
84   \expandafter\@secondoftwo
85 \fi
86 {%
87   \def\IfLang@IfDefined#1{%
88     \expandafter\ifx\csname#1\endcsname\relax
89     \expandafter\@secondoftwo
90     \else
91     \expandafter\@firstoftwo
92     \fi
93   }%
94 }{%
95   \def\IfLang@IfDefined#1{%
96     \ifnum\ifcsname#1\endcsname
97       \expandafter\ifx\csname#1\endcsname\relax
98       1%
99       \else
100       0%
101     \fi
102     \else
103     1%
104     \fi
105     =0 %
106     \expandafter\@firstoftwo
107   \else
108     \expandafter\@secondoftwo
109   \fi

```

```

110 }%
111 }

```

2.2.3 Macros for messages

```

112 \begingroup\expandafter\expandafter\expandafter\endgroup
113 \expandafter\ifx\csname RequirePackage\endcsname\relax
114 \input infwarerr.sty\relax
115 \else
116 \RequirePackage{infwarerr}[2007/09/09]%
117 \fi

```

2.2.4 Support for etex.src

`\IfLang@prefix`

```

118 \begingroup\expandafter\expandafter\expandafter\endgroup
119 \expandafter\ifx\csname uselanguage\endcsname\relax
120 \@PackageInfoNoLine{iflang}{%
121 Naming convention for patterns: babel%
122 }%
123 \def\IfLang@prefix{l}%
124 \else
125 \@PackageInfoNoLine{iflang}{%
126 Naming convention for patterns: etex.src%
127 }%
128 \def\IfLang@prefix{lang}%
129 \let\IfLang@OrgUseLanguage\uselanguage
130 \def\uselanguage#1{%
131 \edef\language{#1}%
132 \IfLang@OrgUseLanguage{#1}%
133 }%

```

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.

```

134 \begingroup\expandafter\expandafter\expandafter\endgroup
135 \expandafter\ifx\csname language\endcsname\relax
136 \begingroup\expandafter\expandafter\expandafter\endgroup
137 \expandafter\ifx\csname lang@USenglish\endcsname\relax
138 \@PackageWarningNoLine{iflang}{%
139 \string\lang@USenglish\space is missing%
140 }%
141 \else
142 \ifnum\lang@USenglish=\language
143 \def\language{USenglish}%
144 \else
145 \@PackageWarningNoLine{iflang}{%
146 \string\language\space is not set,\MessageBreak
147 current language is unknown%
148 }%
149 \fi
150 \fi
151 \fi
152 \fi
153 \begingroup\expandafter\expandafter\expandafter\endgroup
154 \expandafter\ifx\csname language\endcsname\relax
155 \@PackageInfoNoLine{iflang}{%
156 \string\language\space is not set%
157 }%
158 \fi

```

2.3 \IfLanguagePatterns

\IfLanguagePatterns

```

159 \def\IfLanguagePatterns#1{%
160   \ifnum\IfLang@IfDefined{\IfLang@prefix#1}{%
161     \ifnum\csname\IfLang@prefix#1\endcsname=\language
162       0%
163     \else
164       1%
165     \fi
166   }{1}=0 %
167   \expandafter\@firstoftwo
168 \else
169   \expandafter\@secondoftwo
170 \fi
171 }
```

2.4 \IfLanguageName

```

172 \begingroup\expandafter\expandafter\expandafter\endgroup
173 \expandafter\ifx\csname pdfstrcmp\endcsname\relax
174   \expandafter\@firstoftwo
175 \else
176   \expandafter\@secondoftwo
177 \fi
178 {%
```

We do not have `\pdfstrcmp`. Thus we must define our own expandable string comparison. The following implementation is based on a T_EX 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 consist of further macros, we need a variant that allows macros in the first string, too.

```

179   \def\IfLang@StrNil{\relax}%
180   \def\IfLang@StrEqual#1{%
181     \number\IfLang@StrEqualStart{#1}\IfLang@StrNil
182   }%
183   \def\IfLang@StrEqualStart#1#2#3{%
184     \ifx#3\IfLang@StrNil
185       \IfLang@StrEqualStop
186     \fi
187     \ifcat\noexpand#3\relax
188       \IfLang@StrExpand{#1}{#2}#3%
189     \fi
190     \IfLang@StrEqualStart{\if#3#1}{#2\fi}%
191   }%
192   \def\IfLang@StrEqualStop\fi#1\IfLang@StrEqualStart#2#3#4{%
193     \fi
194     #2#4\relax'#313 %
195   }%
196   \def\IfLang@StrExpand#1#2#3\fi\IfLang@StrEqualStart#4#5{%
197     \fi
198     \IfLang@@StrExpand{#1}{#2}#3%
199   }%
200   \def\IfLang@@StrExpand#1#2#3\IfLang@StrNil{%
201     \expandafter\IfLang@@@StrExpand#3\IfLang@StrNil{#1}{#2}%
202   }%
203   \def\IfLang@@@StrExpand#1\IfLang@StrNil#2#3{%
204     \IfLang@StrEqualStart{#2}{#3}#1\IfLang@StrNil
205   }%
```

\IfLanguageName

```

206 \def\IfLanguageName#1{%
207   \ifnum\IfLang@IfDefined{language#1}{%
208     \if\expandafter\IfLang@StrEqual\expandafter%
209       {\language#1}{\language#1}%
210     0%
211   \else
212     1%
213   \fi
214   }{1}=0 %
215   \expandafter\@firstoftwo
216 \else
217   \expandafter\@secondoftwo
218 \fi
219 }%
220 }{%
```

\IfLanguageName

```

221 \def\IfLanguageName#1{%
222   \ifnum\IfLang@IfDefined{language#1}{%
223     \pdfstrcmp{#1}{\language#1}%
224     }{1}=0 %
225     \expandafter\@firstoftwo
226   \else
227     \expandafter\@secondoftwo
228   \fi
229 }%
230 }
```

2.5 Check plausibility of \language#1

```

231 \begin{group}\expandafter\expandafter\expandafter\end{group}
232 \expandafter\ifx\csname language#1\endcsname\relax
233 \else
234   \IfLanguagePatterns{\language#1}{%
235     \@PackageWarningNoLine{iflang}{%
236       Mismatch between \string\language\space
237       (patterns)\MessageBreak
238       and setting of \string\language#1
239     }%
240   }%
241 \fi
242 \IfLang@AtEnd
243 \end{package}
```

3 Test

3.1 Catcode checks for loading

```

244 (*test1)
245 \catcode'\@=11 %
246 \def\RestoreCatcodes{%
247   \count@=0 %
248   \loop
249   \edef\RestoreCatcodes{%
250     \RestoreCatcodes
251     \catcode\the\count@=\the\catcode\count@\relax
252   }%
253   \ifnum\count@<255 %
254     \advance\count@\@ne
255   \repeat
```

```

256
257 \def\RangeCatcodeInvalid#1#2{%
258   \count@=#1\relax
259   \loop
260     \catcode\count@=15 %
261     \ifnum\count@<#2\relax
262       \advance\count@\@ne
263     \repeat
264 }
265 \def\Test{%
266   \RangeCatcodeInvalid{0}{47}%
267   \RangeCatcodeInvalid{58}{64}%
268   \RangeCatcodeInvalid{91}{96}%
269   \RangeCatcodeInvalid{123}{255}%
270   \catcode'\@=12 %
271   \catcode'\=0 %
272   \catcode'\{=1 %
273   \catcode'\}=2 %
274   \catcode'\#=6 %
275   \catcode'\[=12 %
276   \catcode'\]=12 %
277   \catcode'\%=14 %
278   \catcode'\ =10 %
279   \catcode13=5 %
280   \input iflang.sty\relax
281   \RestoreCatcodes
282 }
283 \Test
284 \csname @@end\endcsname
285 \end
286 </test1>

```

3.2 Test with L^AT_EX

```

287 (*test2 | test3)
288 \NeedsTeXFormat{LaTeX2e}
289 \test3\let\pdfstrcmp\relax
290 \nofiles
291 \documentclass{minimal}
292 \usepackage{qstest}
293 \IncludeTests{*}
294 \LogTests{log}{*}{*}
295 \usepackage[english,naustrian,ngerman]{babel}
296 \usepackage{iflang}
297 \begin{document}
298 \begin{qstest}\IfLanguagePatterns{language, pattern}
299   \def\test#1#2{%
300     \Expect*{\IfLanguagePatterns{#1}{true}{false}}{#2}%
301   }%
302   \test{ngerman}{true}%
303   \test{naustrian}{true}%
304   \test{english}{false}%
305   \test{foobar}{false}%
306 \end{qstest}
307 \begin{qstest}\IfLanguageName{language, name}
308   \def\test#1#2{%
309     \Expect*{\IfLanguageName{#1}{true}{false}}{#2}%
310   }%
311   \test{ngerman}{true}%
312   \test{naustrian}{false}%
313   \selectlanguage{naustrian}%
314   \test{ngerman}{false}%
315   \test{naustrian}{true}%

```



```

316 \test{foobar}{false}%
317 %
318 \def\language{naustrian}%
319 \test{naustrian}{true}%
320 \test{ngerman}{false}%
321 %
322 \edef\language{\string naustrian}%
323 \test{naustrian}{true}%
324 \test{ngerman}{false}%
325 %
326 \def\language{naustrian}%
327 \makeatletter
328 \@onelevel@sanitize\language
329 \test{naustrian}{true}%
330 \test{ngerman}{false}%
331 %
332 \def\language{naustrian}%
333 \def\xaustrian{naustrian}%
334 \def\xgerman{ngerman}%
335 \test{\xaustrian}{true}%
336 \test{\xgerman}{false}%
337 %
338 \def\language{\xaustrian}%
339 \test{naustrian}{true}%
340 \test{ngerman}{false}%
341 \test{\xaustrian}{true}%
342 \test{\xgerman}{false}%
343 \test{\language}{true}%
344 \test{\language\space}{false}%
345 %
346 \def\language{\empty\xaustrian\empty}%
347 \test{naustrian}{true}%
348 \test{ngerman}{false}%
349 \test{\empty\xaustrian\empty}{true}%
350 \test{\empty\xgerman\empty}{false}%
351 \end{qstest}
352 \begin{qstest}\IfDefined{defined}
353 \makeatletter
354 \let\foobar\relax
355 \Expect*{\IfLang\IfDefined{foobar}{true}{false}}{false}%
356 \Expect*{\ifx\foobar\relax true\else false\fi}{true}%
357 \let\foobar\UNDEFINED
358 \Expect*{\IfLang\IfDefined{foobar}{true}{false}}{false}%
359 \Expect*{\ifx\foobar\relax true\else false\fi}{false}%
360 \Expect*{\ifx\foobar\UNDEFINED true\else false\fi}{true}%
361 \end{qstest}
362 \end{document}
363 /test2 | test3)

```

3.3 Test with plain-TeX and ε -TeX

```

364 (*test4)
365 %% Format 'etex' based on 'language.def'
366 \input iflang.sty
367 \catcode64=12
368
369 \def\TestGeneric#1#2#3{%
370 \begingroup
371 \edef\x{#1#2}{true}{false}}%
372 \edef\y{#3}%
373 \ifx\x\y
374 \else
375 \errmessage{Failed test: \string#1#2 <> #3}%

```

```

376     \fi
377 \endgroup
378 }
379 \def\TestPatterns{\TestGeneric\IfLanguagePatterns}
380 \def\TestName{\TestGeneric\IfLanguageName}
381
382 \TestPatterns{USenglish}{true}
383 \TestPatterns{ngerman}{false}
384
385 \TestName{USenglish}{true}
386 \TestName{ngerman}{false}
387
388 \uselanguage{ngerman}
389
390 \TestPatterns{USenglish}{false}
391 \TestPatterns{ngerman}{true}
392
393 \TestName{USenglish}{false}
394 \TestName{ngerman}{true}
395
396 \csname @@end\endcsname
397 \end
398 </test4>

```

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

```

399 (*test5)
400 %% Format 'tex' (vanilla plain-TeX)
401 \let\ifcsname\UNDEFINED
402 \let\pdfstrcmp\UNDEFINED
403 \input iflang.sty
404 \catcode64=11
405
406 \def\TestDefined#1{%
407   \IfLang@IfDefined{foobar}{}{}%
408   \ifx\foobar#1%
409   \else
410     \errmessage{Failed test: \string\foobar <> \string#1}%
411   \fi
412 }
413 \let\foobar\relax
414 \TestDefined\relax
415 \let\foobar\UNDEFINED
416 \TestDefined\relax
417
418 \def\strip@prefix#1>{}
419 \def@onelevel@sanitize#1{%
420   \edef#1{\expandafter\strip@prefix\meaning#1}%
421 }
422 \def\TestCompare#1#2#3{%
423   \begingroup
424     \edef\x{%
425       \if\IfLang@StrEqual{#1}{#2}%
426       true%
427       \else
428       false%
429     \fi
430   }%
431   \def\expect{#3}%
432   \ifx\x\expect
433   \else
434     \def\a{#1}%
435     \@onelevel@sanitize\a

```

```

436 \def\b{#2}%
437 \@onelevel@sanitize\b
438 \errmessage{Failed test: '\a'='\b' <> \expect}%
439 \fi
440 \endgroup
441 }
442 \TestCompare{junk}{junk}{true}
443 \TestCompare{}{}{true}
444 \TestCompare{a}{b}{false}
445 \TestCompare{aa}{bb}{false}
446 \def\a{ax}
447 \def\b{bx}
448 \def\c{\a\b}
449 \def\d{\c\b}
450 \def\exch#1#2{#2#1}
451 \def\gobble#1{}
452 \TestCompare{\gobble a}{}{true}
453 \TestCompare{}{\gobble a}{true}
454 \TestCompare{a\exch xyb}{ayxb}{true}
455 \TestCompare{\c}{\c}{true}
456 \TestCompare{\d}{\c\b}{true}
457
458 \csname @@end\endcsname
459 \end
460 </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 T_EX 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/
```

¹<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
iflang-test1.tex → doc/latex/oberdiek/iflang-test1.tex
iflang-test2.tex → doc/latex/oberdiek/iflang-test2.tex
iflang-test3.tex → doc/latex/oberdiek/iflang-test3.tex
iflang-test4.tex → doc/latex/oberdiek/iflang-test4.tex
iflang-test5.tex → doc/latex/oberdiek/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 (`te \TeX` , `mik \TeX` , ...) relies on file name databases, you must refresh these. For example, `te \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 \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.

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>	274	<code>\a</code>	434, 435, 438, 446, 448
<code>\%</code>	277	<code>\advance</code>	254, 262
<code>\@</code>	245, 270	B	
<code>\@PackageInfoNoLine</code>	120, 125, 155	<code>\b</code>	436, 437, 438, 447, 448, 449, 456
<code>\@PackageWarningNoLine</code>	138, 145, 235	<code>\begin</code>	297, 298, 307, 352
<code>\@firstoftwo</code>	74, 82, 91, 106, 167, 174, 215, 225	C	
<code>\@one</code>	254, 262	<code>\c</code>	448, 449, 455, 456
<code>\@onelevel@sanitize</code>	328, 419, 435, 437	<code>\catcode</code>	3, 4, 5, 6, 7, 31, 32, 33, 34, 35, 36, 37, 38, 56, 58, 62, 64, 245, 251, 260, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 367, 404
<code>\@secondoftwo</code>	77, 84, 89, 108, 169, 176, 217, 227	<code>\count@</code>	247, 251, 253, 254, 258, 260, 261, 262
<code>\[</code>	275	<code>\csname</code>	8, 18, 39, 52, 55, 74, 77, 81, 88, 97, 113, 119, 135, 137, 154, 161, 173, 232, 284, 396, 458
<code>\]</code>	271		
<code>\{</code>	272		
<code>\}</code>	273		
<code>\]</code>	276		
<code>_</code>	278		

D		<code>\loop</code> 248, 259
<code>\d</code> 449, 456		M
<code>\documentclass</code> 291		<code>\makeatletter</code> 327, 353
E		<code>\meaning</code> 420
<code>\empty</code> 12, 346, 349, 350		<code>\MessageBreak</code> 146, 237
<code>\end</code> 285, 306, 351, 361, 362, 397, 459		N
<code>\endcsname</code> 8, 18, 39, 52, 55, 74, 77, 81, 88, 96, 97, 113, 119, 135, 137, 154, 161, 173, 232, 284, 396, 458		<code>\NeedsTeXFormat</code> 288
<code>\endinginput</code> 27		<code>\nofiles</code> 290
<code>\errmessage</code> 375, 410, 438		<code>\number</code> 181
<code>\exch</code> 450, 454		P
<code>\Expect</code> 300, 309, 355, 356, 358, 359, 360		<code>\PackageInfo</code> 23
<code>\expect</code> 431, 432, 438		<code>\pdfstrcmp</code> 223, 289, 402
F		<code>\ProvidesPackage</code> 53
<code>\foobar</code> 354, 356, 357, 359, 360, 408, 410, 413, 415		R
G		<code>\RangeCatcodeInvalid</code> 257, 266, 267, 268, 269
<code>\gobble</code> 451, 452, 453		<code>\repeat</code> 255, 263
I		<code>\RequirePackage</code> 116
<code>\if</code> 190, 208, 425		<code>\RestoreCatcodes</code> 246, 249, 250, 281
<code>\ifcase</code> 9		S
<code>\ifcat</code> 187		<code>\selectlanguage</code> 313
<code>\ifcsname</code> 96, 401		<code>\space</code> 139, 146, 156, 236, 344
<code>\IfLang@@@StrExpand</code> 201, 203		<code>\strip@prefix</code> 418, 420
<code>\IfLang@@StrExpand</code> 198, 200		T
<code>\IfLang@AtEnd</code> 60, 61, 242		<code>\Test</code> 265, 283
<code>\IfLang@IfDefined</code> 80, 160, 207, 222, 355, 358, 407		<code>\test</code> 299, 302, 303, 304, 305, 308, 311, 312, 314, 315, 316, 319, 320, 323, 324, 329, 330, 335, 336, 339, 340, 341, 342, 343, 344, 347, 348, 349, 350
<code>\IfLang@OrgUseLanguage</code> 129, 132		<code>\TestCompare</code> 422, 442, 443, 444, 445, 452, 453, 454, 455, 456
<code>\IfLang@prefix</code> 118, 160, 161		<code>\TestDefined</code> 406, 414, 416
<code>\IfLang@StrEqual</code> 180, 208, 425		<code>\TestGeneric</code> 369, 379, 380
<code>\IfLang@StrEqualStart</code> 181, 183, 190, 192, 196, 204		<code>\TestName</code> 380, 385, 386, 393, 394
<code>\IfLang@StrEqualStop</code> 185, 192		<code>\TestPatterns</code> 379, 382, 383, 390, 391
<code>\IfLang@StrExpand</code> 188, 196		<code>\the</code> 56, 62, 251
<code>\IfLang@StrNil</code> 179, 181, 184, 200, 201, 203, 204		<code>\TMP@EnsureCode</code> 59, 66, 67, 68, 69, 70, 71, 72, 73
<code>\IfLanguageName</code> 2, 206, 221, 309, 380		U
<code>\IfLanguagePatterns</code> 2, 159, 234, 300, 379		<code>\UNDEFINED</code> 357, 360, 401, 402, 415
<code>\ifnum</code> 96, 142, 160, 161, 207, 222, 253, 261		<code>\uselanguage</code> 129, 130, 388
<code>\ifx</code> 10, 12, 18, 39, 47, 74, 77, 81, 88, 97, 113, 119, 135, 137, 154, 173, 184, 232, 356, 359, 360, 373, 408, 432		<code>\usepackage</code> 292, 295, 296
<code>\immediate</code> 20, 41		W
<code>\IncludeTests</code> 293		<code>\write</code> 20, 41
<code>\input</code> 114, 280, 366, 403		X
L		<code>\x</code> 8, 10, 12, 19, 23, 25, 40, 45, 52, 371, 373, 424, 432
<code>\lang@USenglish</code> 139, 142		<code>\xaustrian</code> 333, 335, 338, 341, 346, 349
<code>\language</code> 142, 161, 236		<code>\xgerman</code> 334, 336, 342, 350
<code>\languageName</code> 131, 143, 146, 156, 209, 223, 234, 238, 318, 322, 326, 328, 332, 338, 343, 344, 346		Y
<code>\LogTests</code> 294		<code>\y</code> 372, 373