

The ifpdf package

Heiko Oberdiek
<heiko.oberdiek at gmail.com>

2011/01/30 v2.3

Abstract

This package looks for pdfTeX in pdf mode and implements and sets the switch `\ifpdf`. The detection is based on `\pdfoutput` and the package will not change this value. It works with plain or L^AT_EX formats.

Contents

1	Documentation	2
1.1	Introduction	2
1.2	Usage	2
1.3	Specification	3
2	Implementation	3
2.1	Reload check and package identification	3
2.2	Catcodes	4
2.3	Check for previously defined <code>\ifpdf</code>	5
2.4	<code>\pdfoutput</code> and Lua _T _E _X	6
2.5	<code>\ifpdf</code>	6
2.6	Test for fool attempts	7
2.7	Protocol entry	7
3	Test	8
3.1	Catcode checks for loading	8
4	Installation	9
4.1	Download	9
4.2	Bundle installation	10
4.3	Package installation	10
4.4	Refresh file name databases	10
4.5	Some details for the interested	10
5	History	11
	[2001/06/14 v1.0]	11
	[2001/07/14 v1.1]	11
	[2001/09/26 v1.2]	11
	[2005/07/22 v1.3]	11
	[2006/02/20 v1.4]	11
	[2007/09/09 v1.5]	11
	[2007/12/12 v1.6]	11
	[2008/12/12 v1.7]	12
	[2009/04/10 v2.0]	12
	[2010/01/28 v2.1]	12
	[2010/09/13 v2.2]	12
	[2011/01/30 v2.3]	12

1 Documentation

1.1 Introduction

It is commonly known that Hàn Thê Thành's pdfT_EX generates PDF output directly and many people uses pdfT_EX for this purpose. However the DVI output was never thrown away. In contrary, he new features for typesetting that works in both PDF and DVI mode.

In the meantime many T_EX distributions replace the traditional T_EX binary with pdfT_EX. Then, for example, called as `latex` pdfT_EX works in DVI mode with the L^AT_EX format preloaded, called as `pdflatex` pdfT_EX starts in PDF mode.

Often packages or users want to know, whether the current document is typset by pdfT_EX in PDF mode, because the different modes have different capabilities (color setting, graphics inclusion, ...). For this purpose pdfT_EX's `\pdfoutput` can be asked.

As regulary reader of T_EX newsgroups and mailing lists I could observe many problems with this task. Common errors are:

- pdfT_EX has *two* modes. Using pdfT_EX does not mean that the user always want to have PDF mode. For example, the PostScript support is better in DVI mode in conjunction with a PostScript aware DVI driver (e.g. dvips). Also the additional typesetting features are mode independent and also available in DVI mode.
- L^AT_EX's `\@ifundefined` inherited the side effect from `\csname`. Unknown commands are defined with the meaning of `\relax`. If it is checked, whether `\pdfoutput` is defined, then this should not be forgotten.
- Having `\pdfoutput` does not automatically mean PDF mode. Also the value of `\pdfoutput` must be asked.
- `\pdfoutput` must not be destroyed in some way. Later code and packages are fooled then and will perhaps make wrong decisions. For example they may drop support for PDF mode, because they do not know that pdfT_EX is running at all.

Robin Fairbairns provides an entry for this topic in his excellent FAQ (<http://www.tex.ac.uk/faq>): *Am I using PDFT_EX?*

1.2 Usage

The package `ifpdf` can be used with both plain T_EX and L^AT_EX:

plain T_EX: `\input ifpdf.sty`

L^AT_EX 2_ε: `\usepackage{ifpdf}`

```
\ifpdf      The package provides the switch \ifpdf:

\ifpdf
... do things, if pdfTEX is running in pdf mode ...
\else
... other TEX or pdfTEX in dvi mode ...
\fi
```

Users of the package `ifthen` can use the switch as boolean:

```
\boolean{pdf}
```

The package can also be used to set global documentclass options:

```

\RequirePackage{ifpdf}
\ifpdf
  \documentclass[pdftex,...]{...}
\else
  \documentclass[...]{...}
\fi

```

1.3 Specification

The package have the following properties:

- It asks the setting of `\pdfoutput` for detecting pdfTeX in PDF mode.
- It never changes `\pdfoutput`.
- If `\pdfoutput` is undefined or has the meaning `\relax`, but the engine provides the primitive `\pdfoutput`, then `\pdfoutput` is enabled or restored if possible (only LuaTeX, version 0.36.0 or higher).
- It can be used with many formats including plain TeX and L^AT_EX.

The mode detection implements the following algorithm:

```

if undefined(\pdfoutput)
  \ifpdf := \iffalse % pdfTeX is not running
else
  if \pdfoutput ≤ 0
    \ifpdf := \iffalse % pdfTeX in DVI mode
  else
    \ifpdf := \iftrue % pdfTeX in PDF mode
  fi
fi

```

The function `undefined` checks both cases, undefined command and `\relax`.

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\catcode61\catcode48\catcode32=10\relax%
3 \catcode13=5 % ^~M
4 \endlinechar=13 %
5 \catcode35=6 % #
6 \catcode39=12 % '
7 \catcode44=12 % ,
8 \catcode45=12 % -
9 \catcode46=12 % .
10 \catcode58=12 % :
11 \catcode64=11 % @
12 \catcode123=1 % {
13 \catcode125=2 % }
14 \expandafter\let\expandafter\x\csname ver@ifpdf.sty\endcsname
15 \ifx\x\relax % plain-TeX, first loading
16 \else
17 \def\empty{}%
18 \ifx\x\empty % LaTeX, first loading,
19 % variable is initialized, but \ProvidesPackage not yet seen
20 \else
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{ifpdf}{The package is already loaded}%
29    \aftergroup\endinput
30  \fi
31 \fi
32 \endgroup%

```

Package identification:

```

33 \begingroup\catcode61\catcode48\catcode32=10\relax%
34 \catcode13=5 % ^~M
35 \endlinechar=13 %
36 \catcode35=6 % #
37 \catcode39=12 % '
38 \catcode40=12 % (
39 \catcode41=12 % )
40 \catcode44=12 % ,
41 \catcode45=12 % -
42 \catcode46=12 % .
43 \catcode47=12 % /
44 \catcode58=12 % :
45 \catcode64=11 % @
46 \catcode91=12 % [
47 \catcode93=12 % ]
48 \catcode123=1 % {
49 \catcode125=2 % }
50 \expandafter\ifx\csname ProvidesPackage\endcsname\relax
51   \def\x#1#2#3[#4]{\endgroup
52     \immediate\write-1{Package: #3 #4}%
53     \xdef#1{#4}%
54   }%
55 \else
56   \def\x#1#2[#3]{\endgroup
57     #2[#{#3}]%
58     \ifx#1\@undefined
59       \xdef#1{#3}%
60     \fi
61     \ifx#1\relax
62       \xdef#1{#3}%
63     \fi
64   }%
65 \fi
66 \expandafter\x\csname ver@ifpdf.sty\endcsname
67 \ProvidesPackage{ifpdf}%
68 [2011/01/30 v2.3 Provides the ifpdf switch (H0)]%

```

2.2 Catcodes

```

69 \begingroup\catcode61\catcode48\catcode32=10\relax%
70 \catcode13=5 % ^~M
71 \endlinechar=13 %
72 \catcode123=1 % {
73 \catcode125=2 % }
74 \catcode64=11 % @
75 \def\x{\endgroup
76   \expandafter\edef\csname ifpdf@AtEnd\endcsname{%
77     \endlinechar=\the\endlinechar\relax
78     \catcode13=\the\catcode13\relax
79     \catcode32=\the\catcode32\relax

```

```

80      \catcode35=\the\catcode35\relax
81      \catcode61=\the\catcode61\relax
82      \catcode64=\the\catcode64\relax
83      \catcode123=\the\catcode123\relax
84      \catcode125=\the\catcode125\relax
85    }%
86  }%
87  \x\catcode61\catcode48\catcode32=10\relax%
88  \catcode13=5 % ^^M
89  \newlinechar=13 %
90  \catcode35=6 % #
91  \catcode64=11 % @
92  \catcode123=1 % {
93  \catcode125=2 % }
94  \def\TMP@EnsureCode#1#2{%
95    \edef\ifpdf@AtEnd{%
96      \ifpdf@AtEnd
97        \catcode#1=\the\catcode#1\relax
98      }%
99    \catcode#1=#2\relax
100  }
101  \TMP@EnsureCode{10}{12}% ^^J
102  \TMP@EnsureCode{39}{12}% '
103  \TMP@EnsureCode{40}{12}% (
104  \TMP@EnsureCode{41}{12}% )
105  \TMP@EnsureCode{44}{12}% ,
106  \TMP@EnsureCode{45}{12}% -
107  \TMP@EnsureCode{46}{12}% .
108  \TMP@EnsureCode{47}{12}% /
109  \TMP@EnsureCode{58}{12}% :
110  \TMP@EnsureCode{60}{12}% <
111  \TMP@EnsureCode{91}{12}% [
112  \TMP@EnsureCode{93}{12}% ]
113  \TMP@EnsureCode{94}{7}% ^
114  \TMP@EnsureCode{96}{12}% `
115  \edef\ifpdf@AtEnd{\ifpdf@AtEnd\noexpand\endinput}

```

2.3 Check for previously defined \ifpdf

```

116 \begingroup
117   \expandafter\ifx\csname ifpdf\endcsname\relax
118   \else
119     \edef\i/{\expandafter\string\csname ifpdf\endcsname}%
120     \expandafter\ifx\csname PackageError\endcsname\relax
121       \def\x#1#2{%
122         \edef\z{#2}%
123         \expandafter\errhelp\expandafter{\z}%
124         \errmessage{Package ifpdf Error: #1}%
125       }%
126       \def\y{^^J}%
127       \newlinechar=10 %
128     \else
129       \def\x#1#2{%
130         \PackageError{ifpdf}{#1}{#2}%
131       }%
132       \def\y{\MessageBreak}%
133     \fi
134     \x{Name clash, \i/ is already defined}{%
135       Incompatible versions of \i/ can cause problems,\y
136       therefore package loading is aborted.%
137     }%
138   \endgroup
139   \expandafter\ifpdf@AtEnd

```

```

140 \fi%
141 \endgroup

```

2.4 \pdfoutput and LuaTeX

It might happen, that LuaTeX is running, but \pdfoutput does not exist. In version 0.40 only \directlua is available at startup time. The enabling Lua function was already added in version 0.36. Thus we can ignore older versions, here \pdfoutput is available at startup time.

```

142 \begingroup
143 \def\skip#1\relax\begingroup{}%
144 \expandafter\ifx\csname pdfoutput\endcsname\relax
145 \else
146 \expandafter\skip
147 \fi
148 \expandafter\ifx\csname directlua\endcsname\relax
149 \expandafter\skip
150 \fi
151 \endgroup
152 \begingroup\expandafter\expandafter\expandafter\endgroup
153 \expandafter\ifx\csname RequirePackage\endcsname\relax
154 \def\TMP@RequirePackage#1[#2]{%
155 \begingroup\expandafter\expandafter\expandafter\endgroup
156 \expandafter\ifx\csname ver@#1.sty\endcsname\relax
157 \input #1.sty\relax
158 \fi
159 }%
160 \TMP@RequirePackage{ifluatex}[2009/04/10]%
161 \else
162 \RequirePackage{ifluatex}[2009/04/10]%
163 \fi
164 \ifluatex
165 \ifnum\luatexversion<36 %

```

Unhappily LuaTeX's \primitive (derived from pdfTeX's \pdfprimitive) cannot be used:

```
\protected\gdef\pdfoutput{\primitive\pdfoutput}
```

Setting a value works, but getting fails, because TeX does no longer see it as number. It is unexpandable and breaks numerical contexts. This was fixed in pdfTeX 1.40.10 (bugfix #4289: "\primitive\pdfoutput cannot be queried").

```

166 \else
167 \begingroup
168 \directlua{tex.enableprimitives('ifpdf', {'pdfoutput'})}%
169 \global\let\pdfoutput\ifpdfpdfoutput
170 \endgroup
171 \fi
172 \fi
173 \relax\begingroup\endgroup

```

2.5 \ifpdf

\ifpdf Create and set the switch. \newif initializes the switch with \iffalse. \newif is \outer in plain TeX.

```

174 \begingroup\expandafter\expandafter\expandafter\endgroup
175 \expandafter\ifx\csname newif\endcsname\relax
176 \edef\pdffalse{%
177 \let
178 \expandafter\noexpand\csname ifpdf\endcsname
179 \expandafter\noexpand\csname iffalse\endcsname
180 }%

```

```

181 \edef\pdftrue{%
182   \let
183   \expandafter\noexpand\csname ifpdf\endcsname
184   \expandafter\noexpand\csname iftrue\endcsname
185 }%
186 \pdffalse
187 \else
188   \csname newif\expandafter\endcsname\csname ifpdf\endcsname
189 \fi

Test \pdfoutput. Is it defined and different from \relax? Someone could have
used LATEX internal \ifundefined, or something else involving. Notice, \csname
is executed inside a group for the test to cancel the side effect of \csname.

190 \begingroup\expandafter\expandafter\expandafter\endgroup
191 \expandafter\ifx\csname pdfoutput\endcsname\relax
192 \else
193   \ifnum\pdfoutput<1 %
\pdfoutput=0 or negative, so not generating pdf.
194   \else
195     \pdftrue
196   \fi
197 \fi

```

2.6 Test for fool attempts

```

198 \begingroup
199   \expandafter\ifx\csname pdfoutput\endcsname\relax
200   \else
201     \escapechar=92 %
202     \edef\m{\meaning\pdfoutput}%
203     \edef\p{\string\pdfoutput}%
204     \ifx\m\p
205     \else
206       \expandafter\ifx\csname PackageWarningNoLine\endcsname\relax
207       \def\PackageWarningNoLine#1#2{%
208         \immediate\write16{%
209           Package ‘#1’ Warning: #2.%
210         }%
211       }%
212     \fi
213     \PackageWarningNoLine{ifpdf}{%
214       Someone has redefined \string\pdfoutput%
215     }%
216   \fi
217 \fi
218 \endgroup

```

2.7 Protocol entry

Log comment:

```

219 \begingroup
220   \expandafter\ifx\csname PackageInfo\endcsname\relax
221   \def\x#1#2{%
222     \immediate\write-1{Package #1 Info: #2.}%
223   }%
224   \else
225     \let\x\PackageInfo
226     \expandafter\let\csname on@line\endcsname\empty
227   \fi
228   \x{ifpdf}{pdfTeX in PDF mode is \ifpdf\else not \fi detected}%
229 \endgroup

230 \ifpdf@AtEnd%

```

231 `</package>`

3 Test

3.1 Catcode checks for loading

```
232 <*test1>
233 \catcode'\{=1 %
234 \catcode'\}=2 %
235 \catcode'\#=6 %
236 \catcode'\@=11 %
237 \expandafter\ifx\csname count@\endcsname\relax
238 \countdef\count@=255 %
239 \fi
240 \expandafter\ifx\csname @gobble\endcsname\relax
241 \long\def\@gobble#1{}%
242 \fi
243 \expandafter\ifx\csname @firstofone\endcsname\relax
244 \long\def\@firstofone#1{#1}%
245 \fi
246 \expandafter\ifx\csname loop\endcsname\relax
247 \expandafter\@firstofone
248 \else
249 \expandafter\@gobble
250 \fi
251 {%
252 \def\loop#1\repeat{%
253 \def\body{#1}%
254 \iterate
255 }%
256 \def\iterate{%
257 \body
258 \let\next\iterate
259 \else
260 \let\next\relax
261 \fi
262 \next
263 }%
264 \let\repeat=\fi
265 }%
266 \def\RestoreCatcodes{}
267 \count@=0 %
268 \loop
269 \edef\RestoreCatcodes{%
270 \RestoreCatcodes
271 \catcode\the\count@=\the\catcode\count@\relax
272 }%
273 \ifnum\count@<255 %
274 \advance\count@ 1 %
275 \repeat
276
277 \def\RangeCatcodeInvalid#1#2{%
278 \count@=#1\relax
279 \loop
280 \catcode\count@=15 %
281 \ifnum\count@<#2\relax
282 \advance\count@ 1 %
283 \repeat
284 }
285 \def\RangeCatcodeCheck#1#2#3{%
286 \count@=#1\relax
```



```

287 \loop
288   \ifnum#3=\catcode\count@
289   \else
290     \errmessage{%
291       Character \the\count@\space
292       with wrong catcode \the\catcode\count@\space
293       instead of \number#3%
294     }%
295   \fi
296 \ifnum\count@<#2\relax
297   \advance\count@ 1 %
298 \repeat
299 }
300 \def\space{ }
301 \expandafter\ifx\csname LoadCommand\endcsname\relax
302 \def\LoadCommand{\input ifpdf.sty\relax}%
303 \fi
304 \def\Test{%
305   \RangeCatcodeInvalid{0}{47}%
306   \RangeCatcodeInvalid{58}{64}%
307   \RangeCatcodeInvalid{91}{96}%
308   \RangeCatcodeInvalid{123}{255}%
309   \catcode'\@=12 %
310   \catcode'\=0 %
311   \catcode'\%=14 %
312   \LoadCommand
313   \RangeCatcodeCheck{0}{36}{15}%
314   \RangeCatcodeCheck{37}{37}{14}%
315   \RangeCatcodeCheck{38}{47}{15}%
316   \RangeCatcodeCheck{48}{57}{12}%
317   \RangeCatcodeCheck{58}{63}{15}%
318   \RangeCatcodeCheck{64}{64}{12}%
319   \RangeCatcodeCheck{65}{90}{11}%
320   \RangeCatcodeCheck{91}{91}{15}%
321   \RangeCatcodeCheck{92}{92}{0}%
322   \RangeCatcodeCheck{93}{96}{15}%
323   \RangeCatcodeCheck{97}{122}{11}%
324   \RangeCatcodeCheck{123}{255}{15}%
325   \RestoreCatcodes
326 }
327 \Test
328 \csname @@end\endcsname
329 \end
330 </test1>

```

4 Installation

4.1 Download

Package. This package is available on CTAN¹:

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

[CTAN:macros/latex/contrib/oberdiek/ifpdf.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](#)

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

TDS refers to the standard “A Directory Structure for $\text{T}_{\text{E}}\text{X}$ 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 $\text{T}_{\text{E}}\text{X}$:

```
tex ifpdf.dtx
```

TDS. Now the different files must be moved into the different directories in your installation TDS tree (also known as `texmf` tree):

<code>ifpdf.sty</code>	\rightarrow <code>tex/generic/oberdiek/ifpdf.sty</code>
<code>ifpdf.pdf</code>	\rightarrow <code>doc/latex/oberdiek/ifpdf.pdf</code>
<code>test/ifpdf-test1.tex</code>	\rightarrow <code>doc/latex/oberdiek/test/ifpdf-test1.tex</code>
<code>ifpdf.dtx</code>	\rightarrow <code>source/latex/oberdiek/ifpdf.dtx</code>

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 ifpdf.pdf unpack_files output .
```

Unpacking with \LaTeX . The `.dtx` chooses its action depending on the format:

plain $\text{T}_{\text{E}}\text{X}$: 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{ifpdf.dtx}
```

Do not forget to quote the argument according to the demands of your shell.

Generating the documentation. You can use both the `.dtx` or the `.drv` to generate the documentation. The process can be configured by the configuration file `ltxdoc.cfg`. For instance, put this line into this file, if you want to have A4 as paper format:

```
\PassOptionsToClass{a4paper}{article}
```

An example follows how to generate the documentation with pdfL^AT_EX:

```
pdflatex ifpdf.dtx
makeindex -s gind.ist ifpdf.idx
pdflatex ifpdf.dtx
makeindex -s gind.ist ifpdf.idx
pdflatex ifpdf.dtx
```

5 History

[2001/06/14 v1.0]

- First public version.

[2001/07/14 v1.1]

- Documentation addition: global options

[2001/09/26 v1.2]

- Documentation typo corrected.
- Version number corrected.
- Line number in log entry removed.

[2005/07/22 v1.3]

- Some source code comments from Robin Fairbairns added.
- Bug fix for negative values of `\pdfoutput` (Oleg Katsitadze)
- LPPL 1.3
- Installation section with locations added.

[2006/02/20 v1.4]

- DTX framework.
- More robust check in case of undefined `\pdfoutput`.
- Extended documentation.

[2007/09/09 v1.5]

- Catcode settings added.

[2007/12/12 v1.6]

- Minor update.

[2008/12/12 v1.7]

- Fix in documentation for `\boolean` (found by S. Venkataraman).
- Code is not changed.

[2009/04/10 v2.0]

- Support for LuaTeX 0.40 added.
- Checks, whether `\pdfoutput` was changed.

[2010/01/28 v2.1]

- Compatibility to iniTeX added.

[2010/09/13 v2.2]

- Code with `\escapechar` rewritten because of LuaTeX bug: `\escapechar` does not work.

[2011/01/30 v2.3]

- Already loaded package files are not input in plain TeX.

6 Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; plain numbers refer to the code lines where the entry is used.

Symbols			
<code>\#</code>	235	<code>\csname</code>	14, 21, 50, 66, 76, 117, 119, 120, 144, 148, 153, 156, 175, 178, 179, 183, 184, 188, 191, 199, 206, 220, 226, 237, 240, 243, 246, 301, 328
<code>\%</code>	311		
<code>\@</code>	236, 309		
<code>\@firstofone</code>	244, 247		
<code>\@gobble</code>	241, 249		
<code>\@undefined</code>	58		
<code>\%</code>	310		
<code>\{</code>	233		
<code>\}</code>	234		
A		D	
<code>\advance</code>	274, 282, 297	<code>\directlua</code>	168
<code>\aftergroup</code>	29		
B		E	
<code>\body</code>	253, 257	<code>\empty</code>	17, 18, 226
C		<code>\end</code>	329
<code>\catcode</code>	2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 69, 70, 72, 73, 74, 78, 79, 80, 81, 82, 83, 84, 87, 88, 90, 91, 92, 93, 97, 99, 233, 234, 235, 236, 271, 280, 288, 292, 309, 310, 311	<code>\endcsname</code>	14, 21, 50, 66, 76, 117, 119, 120, 144, 148, 153, 156, 175, 178, 179, 183, 184, 188, 191, 199, 206, 220, 226, 237, 240, 243, 246, 301, 328
<code>\count@</code>	238, 267, 271, 273, 274, 278, 280, 281, 282, 286, 288, 291, 292, 296, 297	<code>\endinput</code>	29, 115
<code>\countdef</code>	238	<code>\endlinechar</code>	4, 35, 71, 77, 89
		<code>\errhelp</code>	123
		<code>\errmessage</code>	124, 290
		<code>\escapechar</code>	201
		I	
		<code>\i</code>	119, 134, 135
		<code>\ifluatex</code>	164
		<code>\ifnum</code>	165, 193, 273, 281, 288, 296
		<code>\ifpdf</code>	2, 174, 228
		<code>\ifpdf@AtEnd</code>	95, 96, 115, 139, 230
		<code>\ifpdfpdfoutput</code>	169

<code>\ifx</code>	15, 18, 21, 50, 58, 61, 117, 120, 144, 148, 153, 156, 175, 191, 199, 204, 206, 220, 237, 240, 243, 246, 301
<code>\immediate</code>	23, 52, 208, 222
<code>\input</code>	157, 302
<code>\iterate</code>	254, 256, 258
L	
<code>\LoadCommand</code>	302, 312
<code>\loop</code>	252, 268, 279, 287
<code>\luatexversion</code>	165
M	
<code>\m</code>	202, 204
<code>\meaning</code>	202
<code>\MessageBreak</code>	132
N	
<code>\newlinechar</code>	127
<code>\next</code>	258, 260, 262
<code>\number</code>	293
P	
<code>\p</code>	203, 204
<code>\PackageError</code>	130
<code>\PackageInfo</code>	26, 225
<code>\PackageWarningNoLine</code>	207, 213
<code>\pdffalse</code>	176, 186
<code>\pdfoutput</code>	169, 193, 202, 203, 214
<code>\pdftrue</code>	181, 195
<code>\ProvidesPackage</code>	19, 67
R	
<code>\RangeCatcodeCheck</code> 285, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324
<code>\RangeCatcodeInvalid</code> 277, 305, 306, 307, 308
<code>\repeat</code>	252, 264, 275, 283, 298
<code>\RequirePackage</code>	162
<code>\RestoreCatcodes</code>	266, 269, 270, 325
S	
<code>\skip</code>	143, 146, 149
<code>\space</code>	291, 292, 300
T	
<code>\Test</code>	304, 327
<code>\the</code>	77, 78, 79, 80, 81, 82, 83, 84, 97, 271, 291, 292
<code>\TMP@EnsureCode</code>	94, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114
<code>\TMP@RequirePackage</code>	154, 160
W	
<code>\write</code>	23, 52, 208, 222
X	
<code>\x</code>	14, 15, 18, 22, 26, 28, 51, 56, 66, 75, 87, 121, 129, 134, 221, 225, 228
Y	
<code>\y</code>	126, 132, 135
Z	
<code>\z</code>	122, 123