

# The ifpdf package

Heiko Oberdiek  
<oberdiek@uni-freiburg.de>

2009/04/10 v2.0

## 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<sup>A</sup>T<sub>E</sub>X formats.

## Contents

<b>1</b>	<b>Documentation</b>	<b>2</b>
1.1	Introduction . . . . .	2
1.2	Usage . . . . .	2
1.3	Specification . . . . .	3
<b>2</b>	<b>Implementation</b>	<b>3</b>
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 L <sup>A</sup> T <sub>E</sub> X . . . . .	5
2.5	<code>\ifpdf</code> . . . . .	6
2.6	Test for fool attempts . . . . .	6
2.7	Protocol entry . . . . .	7
<b>3</b>	<b>Test</b>	<b>7</b>
3.1	Catcode checks for loading . . . . .	7
<b>4</b>	<b>Installation</b>	<b>8</b>
4.1	Download . . . . .	8
4.2	Bundle installation . . . . .	9
4.3	Package installation . . . . .	9
4.4	Refresh file name databases . . . . .	9
4.5	Some details for the interested . . . . .	9
<b>5</b>	<b>History</b>	<b>10</b>
	[2001/06/14 v1.0] . . . . .	10
	[2001/07/14 v1.1] . . . . .	10
	[2001/09/26 v1.2] . . . . .	10
	[2005/07/22 v1.3] . . . . .	10
	[2006/02/20 v1.4] . . . . .	10
	[2007/09/09 v1.5] . . . . .	10
	[2007/12/12 v1.6] . . . . .	10
	[2008/12/12 v1.7] . . . . .	11
	[2009/04/10 v2.0] . . . . .	11
<b>6</b>	<b>Index</b>	<b>11</b>

# 1 Documentation

## 1.1 Introduction

It is commonly known that Hàn Thê Thành's pdfT<sub>E</sub>X generates PDF output directly and many people uses pdfT<sub>E</sub>X 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<sub>E</sub>X distributions replace the traditional T<sub>E</sub>X binary with pdfT<sub>E</sub>X. Then, for example, called as `latex` pdfT<sub>E</sub>X works in DVI mode with the L<sup>A</sup>T<sub>E</sub>X format preloaded, called as `pdflatex` pdfT<sub>E</sub>X starts in PDF mode.

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

As regulary reader of T<sub>E</sub>X newsgroups and mailing lists I could observe many problems with this task. Common errors are:

- pdfT<sub>E</sub>X has *two* modes. Using pdfT<sub>E</sub>X 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<sup>A</sup>T<sub>E</sub>X'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<sub>E</sub>X 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<sub>E</sub>X?**

## 1.2 Usage

The package `ifpdf` can be used with both plain-T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X:

**plain-T<sub>E</sub>X:** `\input ifpdf.sty`

**L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>:** `\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 L<sup>A</sup>T<sub>E</sub>X, version 0.36.0 or higher).
- It can be used with many formats including plain-T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X.

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<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   \catcode123 1 % {
9   \catcode125 2 % }
10  \expandafter\let\expandafter\x\csname ver@ifpdf.sty\endcsname
11  \ifx\x\relax % plain-TEX, first loading
12  \else
13    \def\empty{}%
14    \ifx\x\empty % LATEX, first loading,
15    % variable is initialized, but \ProvidesPackage not yet seen
16  \else
17    \catcode35 6 % #
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{ifpdf}{The package is already loaded}%
26     \aftergroup\endinput
27     \fi
28 \fi
29 \endgroup
Package identification:
30 \begingroup
31 \catcode35 6 % #
32 \catcode40 12 % (
33 \catcode41 12 % )
34 \catcode44 12 % ,
35 \catcode45 12 % -
36 \catcode46 12 % .
37 \catcode47 12 % /
38 \catcode58 12 % :
39 \catcode64 11 % @
40 \catcode91 12 % [
41 \catcode93 12 % ]
42 \catcode123 1 % {
43 \catcode125 2 % }
44 \expandafter\ifx\csname ProvidesPackage\endcsname\relax
45 \def\x#1#2#3[#4]{\endgroup
46 \immediate\write-1{Package: #3 #4}%
47 \xdef#1{#4}%
48 }%
49 \else
50 \def\x#1#2[#3]{\endgroup
51 #2[{#3}]%
52 \ifx#1@undefined
53 \xdef#1{#3}%
54 \fi
55 \ifx#1\relax
56 \xdef#1{#3}%
57 \fi
58 }%
59 \fi
60 \expandafter\x\csname ver@ifpdf.sty\endcsname
61 \ProvidesPackage{ifpdf}%
62 [2009/04/10 v2.0 Provides the ifpdf switch (H0)]

```

## 2.2 Catcodes

```

63 \begingroup
64 \catcode123 1 % {
65 \catcode125 2 % }
66 \def\x{\endgroup
67 \expandafter\edef\csname ifpdf@AtEnd\endcsname{%
68 \catcode35 \the\catcode35\relax
69 \catcode64 \the\catcode64\relax
70 \catcode123 \the\catcode123\relax
71 \catcode125 \the\catcode125\relax
72 }%
73 }%
74 \x
75 \catcode35 6 % #
76 \catcode64 11 % @
77 \catcode123 1 % {
78 \catcode125 2 % }
79 \def\TMP@EnsureCode#1#2{%

```

```

80 \edef\ifpdf@AtEnd{%
81   \ifpdf@AtEnd
82   \catcode#1 \the\catcode#1\relax
83 }%
84 \catcode#1 #2\relax
85 }
86 \TMP@EnsureCode{10}{12}% ^^J
87 \TMP@EnsureCode{39}{12}% '
88 \TMP@EnsureCode{40}{12}% (
89 \TMP@EnsureCode{41}{12}% )
90 \TMP@EnsureCode{44}{12}% ,
91 \TMP@EnsureCode{45}{12}% -
92 \TMP@EnsureCode{46}{12}% .
93 \TMP@EnsureCode{47}{12}% /
94 \TMP@EnsureCode{58}{12}% :
95 \TMP@EnsureCode{60}{12}% <
96 \TMP@EnsureCode{61}{12}% =
97 \TMP@EnsureCode{94}{7}% ^
98 \TMP@EnsureCode{96}{12}% ‘

```

## 2.3 Check for previously defined \ifpdf

```

99 \begingroup
100 \expandafter\ifx\csname ifpdf\endcsname\relax
101   \else
102     \edef\i/{\expandafter\string\csname ifpdf\endcsname}%
103     \expandafter\ifx\csname PackageError\endcsname\relax
104       \def\x#1#2{%
105         \edef\z{#2}%
106         \expandafter\errhelp\expandafter{\z}%
107         \errmessage{Package ifpdf Error: #1}%
108       }%
109       \def\y{^^J}%
110       \newlinechar=10 %
111     \else
112       \def\x#1#2{%
113         \PackageError{ifpdf}{#1}{#2}%
114       }%
115       \def\y{\MessageBreak}%
116     \fi
117     \x{Name clash, \i/ is already defined}{%
118       Incompatible versions of \i/ can cause problems,\y
119       therefore package loading is aborted.%
120     }%
121   \endgroup
122   \ifpdf@AtEnd
123   \expandafter\endinput
124 \fi
125 \endgroup

```

## 2.4 \pdfoutput and LuaTeX

It might happen, that L<sup>A</sup>T<sub>E</sub>X 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.

```

126 \begingroup
127 \expandafter\ifx\csname pdfoutput\endcsname\relax
128   \else
129     \def\skip#1\relax\endgroup{\csname fi\endcsname\endgroup}%
130     \skip
131   \fi
132   \expandafter\ifx\csname directlua\endcsname\relax

```

```

133     \def\skip#1\endgroup{\csname fi\endcsname\endgroup}%
134     \skip
135   \fi
136   \expandafter\ifx\csname RequirePackage\endcsname\relax
137     \input ifluatex.sty\relax
138   \else
139     \RequirePackage{ifluatex}[2009/04/10]%
140   \fi
141   \ifluatex
142     \ifnum\luatexversion<36 %

```

Unhappily L<sup>A</sup>T<sub>E</sub>X's `\primitive` (derived from pdfT<sub>E</sub>X's `\pdfprimitive`) cannot be used:

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

Setting a value works, but getting fails, because T<sub>E</sub>X does no longer see it as number. It is unexpandable and breaks numerical contexts.

```

143   \else
144     \directlua{tex.enableprimitives('ifpdf', {'pdfoutput'})}%
145     \global\let\pdfoutput\ifpdfpdfoutput
146   \fi
147   \fi
148   \relax
149 \endgroup

```

## 2.5 `\ifpdf`

`\ifpdf` Create and set the switch. `\newif` initializes the switch with `\iffalse`.

```
150 \newif\ifpdf
```

Test `\pdfoutput`. Is it defined and different from `\relax`? Someone could have used L<sup>A</sup>T<sub>E</sub>X internal `\@ifundefined`, or something else involving. Notice, `\csname` is executed inside a group for the test to cancel the side effect of `\csname`.

```

151 \begingroup\expandafter\expandafter\expandafter\endgroup
152 \expandafter\ifx\csname pdfoutput\endcsname\relax
153 \else
154   \ifnum\pdfoutput<1 %

```

`\pdfoutput=0` or negative, so not generating pdf.

```

155   \else
156     \pdftrue
157   \fi
158 \fi

```

## 2.6 Test for fool attempts

```

159 \begingroup
160   \expandafter\ifx\csname pdfoutput\endcsname\relax
161   \else
162     \escapechar=-1 %
163     \edef\m{\meaning\pdfoutput}%
164     \edef\p{%
165       \string p\string d\string f%
166       \string o\string u\string t\string p\string u\string t%
167     }%
168     \ifx\m\p
169     \else
170       \expandafter\ifx\csname PackageWarningNoLine\endcsname\relax
171         \def\PackageWarningNoLine#1#2{%
172           \immediate\write16{%
173             Package ‘#1’ Warning: #2.%
174           }%

```

```

175     }%
176     \fi
177     \PackageWarningNoLine{ifpdf}{%
178         Someone has redefined \string\pdfoutput%
179     }%
180     \fi
181 \fi
182 \endgroup

```

## 2.7 Protocol entry

Log comment:

```

183 \begingroup
184 \expandafter\ifx\csname PackageInfo\endcsname\relax
185     \def\x#1#2{%
186         \immediate\write-1{Package #1 Info: #2.}%
187     }%
188 \else
189     \let\x\PackageInfo
190     \expandafter\let\csname on@line\endcsname\empty
191 \fi
192 \x{ifpdf}{pdfTeX in pdf mode \ifpdf\else not \fi detected}%
193 \endgroup
194 \ifpdf@AtEnd
195 </package>

```

## 3 Test

### 3.1 Catcode checks for loading

```

196 <*test1>
197 \catcode'\{=1 %
198 \catcode'\}=2 %
199 \catcode'\#=6 %
200 \catcode'\@=11 %
201 \expandafter\ifx\csname count@\endcsname\relax
202     \countdef\count@=255 %
203 \fi
204 \expandafter\ifx\csname @gobble\endcsname\relax
205     \long\def\@gobble#1{}%
206 \fi
207 \expandafter\ifx\csname @firstofone\endcsname\relax
208     \long\def\@firstofone#1{#1}%
209 \fi
210 \expandafter\ifx\csname loop\endcsname\relax
211     \expandafter\@firstofone
212 \else
213     \expandafter\@gobble
214 \fi
215 {%
216     \def\loop#1\repeat{%
217         \def\body{#1}%
218         \iterate
219     }%
220     \def\iterate{%
221         \body
222         \let\next\iterate
223     \else
224         \let\next\relax
225     \fi
226     \next

```

```

227 }%
228 \let\repeat=\fi
229 }%
230 \def\RestoreCatcodes{}
231 \count@=0 %
232 \loop
233 \edef\RestoreCatcodes{%
234 \RestoreCatcodes
235 \catcode\the\count@=\the\catcode\count@\relax
236 }%
237 \ifnum\count@<255 %
238 \advance\count@ 1 %
239 \repeat
240
241 \def\RangeCatcodeInvalid#1#2{%
242 \count@=#1\relax
243 \loop
244 \catcode\count@=15 %
245 \ifnum\count@<#2\relax
246 \advance\count@ 1 %
247 \repeat
248 }
249 \expandafter\ifx\csname LoadCommand\endcsname\relax
250 \def\LoadCommand{\input ifpdf.sty\relax}%
251 \fi
252 \def\Test{%
253 \RangeCatcodeInvalid{0}{47}%
254 \RangeCatcodeInvalid{58}{64}%
255 \RangeCatcodeInvalid{91}{96}%
256 \RangeCatcodeInvalid{123}{255}%
257 \catcode'\@=12 %
258 \catcode'\=0 %
259 \catcode'\{=1 %
260 \catcode'\}=2 %
261 \catcode'\#=6 %
262 \catcode'\[=12 %
263 \catcode'\]=12 %
264 \catcode'\%=14 %
265 \catcode'\ =10 %
266 \catcode13=5 %
267 \LoadCommand
268 \RestoreCatcodes
269 }
270 \Test
271 \csname @@end\endcsname
272 \end
273 </test1>

```

## 4 Installation

### 4.1 Download

**Package.** This package is available on CTAN<sup>1</sup>:

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

[CTAN:macros/latex/contrib/oberdiek/ifpdf.pdf](http://ctan.org/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.

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



`CTAN:install/macros/latex/contrib/oberdiek.tds.zip`

*TDS* refers to the standard “A Directory Structure for  $\text{\TeX}$  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- $\text{\TeX}$ :

```
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  $\text{\TeX}$  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 ifpdf.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{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<sup>A</sup>T<sub>E</sub>X:

```
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 L<sup>A</sup>T<sub>E</sub>X 0.40 added.
- Checks, whether `\pdfoutput` was changed.

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

Symbols			
<code>\#</code> .....	199, 261	<code>\endcsname</code> .....	10, 18, 44, 60, 67, 100, 102, 103, 127, 129, 132, 133, 136, 152, 160, 170, 184, 190, 201, 204, 207, 210, 249, 271
<code>\%</code> .....	264	<code>\endinput</code> .....	26, 123
<code>\@</code> .....	200, 257	<code>\errhelp</code> .....	106
<code>\@firstofone</code> .....	208, 211	<code>\errmessage</code> .....	107
<code>\@gobble</code> .....	205, 213	<code>\escapechar</code> .....	162
<code>\@undefined</code> .....	52	<b>I</b>	
<code>\[</code> .....	262	<code>\i</code> .....	102, 117, 118
<code>\]</code> .....	178, 258	<code>\ifluatex</code> .....	141
<code>\{</code> .....	197, 259	<code>\ifnum</code> .....	142, 154, 237, 245
<code>\}</code> .....	198, 260	<code>\ifpdf</code> .....	2, <u>150</u> , 192
<code>\]</code> .....	263	<code>\ifpdf@AtEnd</code> .....	80, 81, 122, 194
<code>\_</code> .....	265	<code>\ifpdfpdfoutput</code> .....	145
<b>A</b>		<code>\ifx</code> .	11, 14, 18, 44, 52, 55, 100, 103, 127, 132, 136, 152, 160, 168, 170, 184, 201, 204, 207, 210, 249
<code>\advance</code> .....	238, 246	<code>\immediate</code> .....	20, 46, 172, 186
<code>\aftergroup</code> .....	26	<code>\input</code> .....	137, 250
<b>B</b>		<code>\iterate</code> .....	218, 220, 222
<code>\body</code> .....	217, 221	<b>L</b>	
<b>C</b>		<code>\LoadCommand</code> .....	250, 267
<code>\catcode</code> 3, 4, 5, 6, 7, 8, 9, 17, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 64, 65, 68, 69, 70, 71, 75, 76, 77, 78, 82, 84, 197, 198, 199, 200, 235, 244, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266		<code>\loop</code> .....	216, 232, 243
<code>\count@</code> .....	202, 231, 235, 237, 238, 242, 244, 245, 246	<code>\luatexversion</code> .....	142
<code>\countdef</code> .....	202	<b>M</b>	
<code>\csname</code> .....	10, 18, 44, 60, 67, 100, 102, 103, 127, 129, 132, 133, 136, 152, 160, 170, 184, 190, 201, 204, 207, 210, 249, 271	<code>\m</code> .....	163, 168
<b>D</b>		<code>\meaning</code> .....	163
<code>\directlua</code> .....	144	<code>\MessageBreak</code> .....	115
<b>E</b>		<b>N</b>	
<code>\empty</code> .....	13, 14, 190	<code>\newif</code> .....	150
<code>\end</code> .....	272	<code>\newlinechar</code> .....	110
<b>P</b>		<code>\next</code> .....	222, 224, 226
		<code>\p</code> .....	164, 168
		<code>\PackageError</code> .....	113
		<code>\PackageInfo</code> .....	23, 189
		<code>\PackageWarningNoLine</code> .....	171, 177
		<code>\pdfoutput</code> .....	145, 154, 163

\pdftrue .....	156	\TMP@EnsureCode ...	79, 86, 87, 88,
\ProvidesPackage .....	15, 61		89, 90, 91, 92, 93, 94, 95, 96, 97, 98
<b>R</b>		<b>W</b>	
\RangeCatcodeInvalid .....		\write .....	20, 46, 172, 186
	241, 253, 254, 255, 256		
\repeat .....	216, 228, 239, 247	<b>X</b>	
\RequirePackage .....	139	\x ..	10, 11, 14, 19, 23, 25, 45, 50, 60,
\RestoreCatcodes ..	230, 233, 234, 268		66, 74, 104, 112, 117, 185, 189, 192
<b>S</b>		<b>Y</b>	
\skip .....	129, 130, 133, 134	\y .....	109, 115, 118
<b>T</b>		<b>Z</b>	
\Test .....	252, 270	\z .....	105, 106
\the .....	68, 69, 70, 71, 82, 235		