

The pdftexcmds package

Heiko Oberdiek*

<heiko.oberdiek at googlemail.com>

2017/03/19 v0.25

Abstract

LuaTeX provides most of the commands of pdfTeX 1.40. However a number of utility functions are removed. This package tries to fill the gap and implements some of the missing primitive using Lua.

Contents

1	Documentation	2
1.1	General principles	3
1.2	Macros	3
1.2.1	Strings	3
1.2.2	Files	4
1.2.3	Timekeeping	4
1.2.4	Miscellaneous	5
1.2.5	Additional macro: <code>\pdf@isprimitive</code>	6
1.2.6	Experimental	6
2	Implementation	6
2.1	Reload check and package identification	7
2.2	Catcodes	8
2.3	Load packages	9
2.4	Without LuaTeX	9
2.5	<code>\pdf@primitive</code> , <code>\pdf@ifprimitive</code>	10
2.5.1	Using LuaTeX's <code>tex.enableprimitives</code>	10
2.5.2	Trying various names to find the primitives	11
2.5.3	Result	12
2.6	X _g TeX	12
2.7	<code>\pdf@isprimitive</code>	13
2.8	<code>\pdf@draftmode</code>	13
2.9	Load Lua module	15
2.10	Lua functions	16
2.10.1	Helper macros	16
2.10.2	Strings	17
2.10.3	Files	18
2.10.4	Timekeeping	19
2.10.5	Shell escape	20
2.11	Lua module	20
2.11.1	Strings	21
2.11.2	Files	23
2.11.3	Timekeeping	25
2.11.4	Miscellaneous	25

*Please report any issues at <https://github.com/ho-tex/oberdiek/issues>

3	Test	26
3.1	Catcode checks for loading	26
3.2	Test for <code>\pdf@isprimitive</code>	28
3.3	Test for <code>\pdf@shellescape</code>	29
3.4	Test for escape functions	29
4	Installation	32
4.1	Download	32
4.2	Bundle installation	32
4.3	Package installation	33
4.4	Refresh file name databases	33
4.5	Some details for the interested	33
5	Catalogue	34
6	References	34
7	History	34
	[2007/11/11 v0.1]	34
	[2007/11/12 v0.2]	34
	[2007/12/12 v0.3]	34
	[2009/04/10 v0.4]	35
	[2009/09/22 v0.5]	35
	[2009/09/23 v0.6]	35
	[2009/12/12 v0.7]	35
	[2010/03/01 v0.8]	35
	[2010/04/01 v0.9]	35
	[2010/11/04 v0.10]	35
	[2010/11/11 v0.11]	35
	[2011/01/30 v0.12]	35
	[2011/03/04 v0.13]	35
	[2011/04/10 v0.14]	35
	[2011/04/16 v0.15]	35
	[2011/04/22 v0.16]	36
	[2011/06/29 v0.17]	36
	[2011/07/01 v0.18]	36
	[2011/07/28 v0.19]	36
	[2011/11/29 v0.20]	36
	[2016/05/10 v0.21]	36
	[2016/05/21 v0.22]	36
	[2016/10/02 v0.23]	36
	[2017/01/29 v0.24]	36
	[2017/03/19 v0.25]	36
8	Index	36

1 Documentation

Some primitives of pdfTeX [1] are not defined by LuaTeX [2]. This package implements macro based solutions using Lua code for the following missing pdfTeX primitives;

- `\pdfstrcmp`
- `\pdfunescapehex`
- `\pdfescapehex`
- `\pdfescapename`
- `\pdfescapestring`
- `\pdffilesize`
- `\pdffilemoddate`

- `\pdffiledump`
- `\pdfmdfivesum`
- `\pdfresettimer`
- `\pdfelapsedtime`
- `\immediate\write18`

The original names of the primitives cannot be used:

- The syntax for their arguments cannot easily simulated by macros. The primitives using key words such as `file` (`\pdfmdfivesum`) or `offset` and `length` (`\pdffiledump`) and uses *⟨general text⟩* for the other arguments. Using token registers assignments, *⟨general text⟩* could be caught. However, the simulated primitives are expandable and register assignments would destroy this important property. (*⟨general text⟩* allows something like `\expandafter\bgroup ...`.)
- The original primitives can be expanded using one expansion step. The new macros need two expansion steps because of the additional macro expansion. Example:

```

\expandafter\foo\pdffilemoddate{file}
vs.
\expandafter\expandafter\expandafter
\foo\pdf@filemoddate{file}

```

LuaTeX isn't stable yet and thus the status of this package is *experimental*. Feedback is welcome.

1.1 General principles

Naming convention: Usually this package defines a macro `\pdf@⟨cmd⟩` if pdfTeX provides `\pdf⟨cmd⟩`.

Arguments: The order of arguments in `\pdf@⟨cmd⟩` is the same as for the corresponding primitive of pdfTeX. The arguments are ordinary undelimited TeX arguments, no *⟨general text⟩* and without additional keywords.

Expandibility: The macro `\pdf@⟨cmd⟩` is expandable if the corresponding pdfTeX primitive has this property. Exact two expansion steps are necessary (first is the macro expansion) except for `\pdf@primitive` and `\pdf@ifprimitive`. The latter ones are not macros, but have the direct meaning of the primitive.

Without LuaTeX: The macros `\pdf@⟨cmd⟩` are mapped to the commands of pdfTeX if they are available. Otherwise they are undefined.

Availability: The macros that the packages provides are undefined, if the necessary primitives are not found and cannot be implemented by Lua.

1.2 Macros

1.2.1 Strings [1, “7.15 Strings”]

`\pdf@strcmp {⟨stringA⟩} {⟨stringB⟩}`

Same as `\pdfstrcmp{⟨stringA⟩}{⟨stringB⟩}`.

`\pdf@unescapehex {⟨string⟩}`

Same as `\pdfunescapehex{⟨string⟩}`. The argument is a byte string given in hexadecimal notation. The result are character tokens from 0 until 255 with catcode 12 and the space with catcode 10.

`\pdf@escapehex {<string>}`
`\pdf@escapestring {<string>}`
`\pdf@escapename {<string>}`

Same as the primitives of pdf_TE_X. However pdf_TE_X does not know about characters with codes 256 and larger. Thus the string is treated as byte string, characters with more than eight bits are ignored.

1.2.2 Files [1, “7.18 Files”]

`\pdf@filesize {<filename>}`

Same as `\pdffilesize{<filename>}`.

`\pdf@filemoddate {<filename>}`

Same as `\pdffilemoddate{<filename>}`.

`\pdf@filedump {<offset>} {<length>} {<filename>}`

Same as `\pdffiledump offset <offset> length <length> {<filename>}`. Both `<offset>` and `<length>` must not be empty, but must be a valid _TE_X number.

`\pdf@mdfivesum {<string>}`

Same as `\pdfmdfivesum{<string>}`. Keyword `file` is supported by macro `\pdf@filemdfivesum`.

`\pdf@filemdfivesum {<filename>}`

Same as `\pdfmdfivesum file{<filename>}`.

1.2.3 Timekeeping [1, “7.17 Timekeeping”]

The timekeeping macros are based on Andy Thomas’ work [3].

`\pdf@resettimer`

Same as `\pdfresettimer`, it resets the internal timer.

`\pdf@elapsedtime`

Same as `\pdfelapsedtime`. It behaves like a read-only integer. For printing purposes it can be prefixed by `\the` or `\number`. It measures the time in scaled seconds (seconds multiplied with 65536) since the latest call of `\pdf@resettimer` or start of program/package. The resolution, the shortest time interval that can be measured, depends on the program and system.

- pdf_TE_X with `gettimeofday`: $\geq 1/65536$ s
- pdf_TE_X with `ftime`: ≥ 1 ms
- pdf_TE_X with `time`: ≥ 1 s
- Lua_TE_X: ≥ 10 ms
(`os.clock()` returns a float number with two decimal digits in Lua_TE_X beta-0.70.1-2011061416 (rev 4277)).

1.2.4 Miscellaneous [1, “7.21 Miscellaneous”]

`\pdf@draftmode`

If the \TeX compiler knows `\pdfdraftmode` or `\draftmode` (pdf \TeX , Lua \TeX), then `\pdf@draftmode` returns, whether this mode is enabled. The result is an implicit number: one means the draft mode is available and enabled. If the value is zero, then the mode is not active or `\pdfdraftmode` is not available. An explicit number is yielded by `\number\pdf@draftmode`. The macro cannot be used to change the mode, see `\pdf@setdraftmode`.

`\pdf@ifdraftmode {true} {false}`

If `\pdfdraftmode` is available and enabled, *true* is called, otherwise *false* is executed.

`\pdf@setdraftmode {value}`

Macro `\pdf@setdraftmode` expects the number zero or one as *value*. Zero deactivates the mode and one enables the draft mode. The macro does not have an effect, if the feature `\pdfdraftmode` is not available.

`\pdf@shellescape`

Same as `\pdfshellescape`. It is or expands to 1 if external commands can be executed and 0 otherwise. In pdf \TeX external commands must be enabled first by command line option or configuration option. In Lua \TeX option `--safer` disables the execution of external commands.

In Lua \TeX before 0.68.0 `\pdf@shellescape` is not available due to a bug in `os.execute()`. The argumentless form crashes in some circumstances with segmentation fault. (It is fixed in version 0.68.0 or revision 4167 of Lua \TeX . and ported to some version of 0.67.0).

Hints for usage:

- Before its use `\pdf@shellescape` should be tested, whether it is available. Example with package `ltxcmds` (loaded by package `pdftexcmds`):

```
\ltx@ifundefined{pdf@shellescape}{%
  % \pdf@shellescape is undefined
}%{
  % \pdf@shellescape is available
}
```

Use `\ltx@ifundefined` in expandable contexts.

- `\pdf@shellescape` might be a numerical constant, expands to the primitive, or expands to a plain number. Therefore use it in contexts where these differences does not matter.
- Use in comparisons, e.g.:

```
\ifnum\pdf@shellescape=0 ...
```

- Print the number: `\number\pdf@shellescape`

`\pdf@system {cmdline}`

It is a wrapper for `\immediate\write18` in pdf \TeX or `os.execute` in Lua \TeX .

In theory `os.execute` returns a status number. But its meaning is quite undefined. Are there some reliable properties? Does it make sense to provide an user interface to this status exit code?

```
\pdf@primitive \cmd
```

Same as `\pdfprimitive` in pdfTeX or LuaTeX. In XeTeX the primitive is called `\primitive`. Despite the current definition of the command `\cmd`, it's meaning as primitive is used.

```
\pdf@ifprimitive \cmd
```

Same as `\ifpdfprimitive` in pdfTeX or LuaTeX. XeTeX calls it `\ifprimitive`. It is a switch that checks if the command `\cmd` has it's primitive meaning.

1.2.5 Additional macro: `\pdf@isprimitive`

```
\pdf@isprimitive \cmd1 \cmd2 {<true>} {<false>}
```

If `\cmd1` has the primitive meaning given by the primitive name of `\cmd2`, then the argument `<true>` is executed, otherwise `<false>`. The macro `\pdf@isprimitive` is expandable. Internally it checks the result of `\meaning` and is therefore available for all TeX variants, even the original TeX. Example with `\makeatletter`:

```
\makeatletter
\pdf@isprimitive{@@input}{input}{%
  \typeout{\string\@@input\space is original\string\input}%
}%
\typeout{Oops, \string\@@input\space is not the %
  original\string\input}%
}
```

1.2.6 Experimental

```
\pdf@unescapehexnative {<string>}
\pdf@escapehexnative {<string>}
\pdf@escapenamenative {<string>}
\pdf@mdfivesumnative {<string>}
```

The variants without `native` in the macro name are supposed to be compatible with pdfTeX. However characters with more than eight bits are not supported and are ignored. If LuaTeX is running, then its UTF-8 coded strings are used. Thus the full unicode character range is supported. However the result differs from pdfTeX for characters with eight or more bits.

```
\pdf@pipe {<cmdline>}
```

It calls `<cmdline>` and returns the output of the external program in the usual manner as byte string (catcode 12, space with catcode 10). The Lua documentation says, that the used `io.popen` may not be available on all platforms. Then macro `\pdf@pipe` is undefined.

2 Implementation

```
1 <*package>
```

2.1 Reload check and package identification

Reload check, especially if the package is not used with \LaTeX .

```
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@pdftexcmds.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{pdftexcmds}{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@pdftexcmds.sty\endcsname
67 \ProvidesPackage{pdftexcmds}%
68 [2017/03/19 v0.25 Utility functions of pdfTeX for LuaTeX (HO)]%

```

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 pdftexcmds@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 \endlinechar=13 %
90 \catcode35=6 % #
91 \catcode64=11 % @
92 \catcode123=1 % {
93 \catcode125=2 % }
94 \def\TMP@EnsureCode#1#2{%
95   \edef\pdftexcmds@AtEnd{%
96     \pdftexcmds@AtEnd
97     \catcode#1=\the\catcode#1\relax
98   }%
99   \catcode#1=#2\relax
100 }
101 \TMP@EnsureCode{0}{12}%
102 \TMP@EnsureCode{1}{12}%
103 \TMP@EnsureCode{2}{12}%
104 \TMP@EnsureCode{10}{12}% ^^J
105 \TMP@EnsureCode{33}{12}% !
106 \TMP@EnsureCode{34}{12}% "
107 \TMP@EnsureCode{38}{4}% &
108 \TMP@EnsureCode{39}{12}% '
109 \TMP@EnsureCode{40}{12}% (
110 \TMP@EnsureCode{41}{12}% )
111 \TMP@EnsureCode{42}{12}% *
112 \TMP@EnsureCode{43}{12}% +
113 \TMP@EnsureCode{44}{12}% ,
114 \TMP@EnsureCode{45}{12}% -
115 \TMP@EnsureCode{46}{12}% .
116 \TMP@EnsureCode{47}{12}% /
117 \TMP@EnsureCode{58}{12}% :
118 \TMP@EnsureCode{60}{12}% <

```

```

119 \TMP@EnsureCode{62}{12}% >
120 \TMP@EnsureCode{91}{12}% [
121 \TMP@EnsureCode{93}{12}% ]
122 \TMP@EnsureCode{94}{7}% ^ (superscript)
123 \TMP@EnsureCode{95}{12}% _ (other)
124 \TMP@EnsureCode{96}{12}% `
125 \TMP@EnsureCode{126}{12}% ~ (other)
126 \edef\pdftexcmds@AtEnd{%
127   \pdftexcmds@AtEnd
128   \escapechar=\number\escapechar\relax
129   \noexpand\endinput
130 }
131 \escapechar=92 %

```

2.3 Load packages

```

132 \begingroup\expandafter\expandafter\expandafter\endgroup
133 \expandafter\ifx\csname RequirePackage\endcsname\relax
134   \def\TMP@RequirePackage#1[#2]{%
135     \begingroup\expandafter\expandafter\expandafter\endgroup
136     \expandafter\ifx\csname ver@#1.sty\endcsname\relax
137       \input #1.sty\relax
138     \fi
139   }%
140 \TMP@RequirePackage{infwarerr}[2007/09/09]%
141 \TMP@RequirePackage{ifluatex}[2010/03/01]%
142 \TMP@RequirePackage{ltxcmds}[2010/12/02]%
143 \TMP@RequirePackage{ifpdf}[2010/09/13]%
144 \else
145   \RequirePackage{infwarerr}[2007/09/09]%
146   \RequirePackage{ifluatex}[2010/03/01]%
147   \RequirePackage{ltxcmds}[2010/12/02]%
148   \RequirePackage{ifpdf}[2010/09/13]%
149 \fi

```

2.4 Without LuaTeX

```

150 \ifluatex
151 \else
152   \@PackageInfoNoLine{pdftexcmds}{LuaTeX not detected}%
153   \def\pdftexcmds@nopdftex{%
154     \@PackageInfoNoLine{pdftexcmds}{pdfTeX >= 1.30 not detected}%
155     \let\pdftexcmds@nopdftex\relax
156   }%
157   \def\pdftexcmds@temp#1{%
158     \begingroup\expandafter\expandafter\expandafter\endgroup
159     \expandafter\ifx\csname pdf#1\endcsname\relax
160       \pdftexcmds@nopdftex
161     \else
162       \expandafter\def\csname pdf@#1\endcsname
163       \expandafter##\expandafter{%
164         \csname pdf#1\endcsname
165       }%
166     \fi
167   }%
168   \pdftexcmds@temp{strcmp}%
169   \pdftexcmds@temp{escapehex}%
170   \let\pdf@escapehexnative\pdf@escapehex
171   \pdftexcmds@temp{unescapehex}%
172   \let\pdf@unescapehexnative\pdf@unescapehex
173   \pdftexcmds@temp{escapestring}%
174   \pdftexcmds@temp{escapename}%
175   \pdftexcmds@temp{filesize}%
176   \pdftexcmds@temp{filemoddate}%

```

```

177 \begingroup\expandafter\expandafter\expandafter\endgroup
178 \expandafter\ifx\csname pdfshellescape\endcsname\relax
179 \pdfcmds@nopdf
180 \ltx@ifundefined{pdfversion}{%
181 }{%
182 \ifnum\pdfversion>120 % 1.21a supports \ifeof18
183 \ifeof18 %
184 \chardef\pdf@shellescape=0 %
185 \else
186 \chardef\pdf@shellescape=1 %
187 \fi
188 \fi
189 }%
190 \else
191 \def\pdf@shellescape{%
192 \pdfshellescape
193 }%
194 \fi
195 \begingroup\expandafter\expandafter\expandafter\endgroup
196 \expandafter\ifx\csname pdffiledump\endcsname\relax
197 \pdfcmds@nopdf
198 \else
199 \def\pdf@filedump#1#2#3{%
200 \pdffiledump offset#1 length#2{#3}%
201 }%
202 \fi
203 \begingroup\expandafter\expandafter\expandafter\endgroup
204 \expandafter\ifx\csname pdfmdfivesum\endcsname\relax
205 \pdfcmds@nopdf
206 \else
207 \def\pdf@mdfivesum#{\pdfmdfivesum}%
208 \let\pdf@mdfivesumnative\pdf@mdfivesum
209 \def\pdf@filemdfivesum#{\pdfmdfivesum file}%
210 \fi
211 \def\pdf@system#{%
212 \immediate\write18%
213 }%
214 \def\pdfcmds@temp#1{%
215 \begingroup\expandafter\expandafter\expandafter\endgroup
216 \expandafter\ifx\csname pdf#1\endcsname\relax
217 \pdfcmds@nopdf
218 \else
219 \expandafter\let\csname pdf@#1\endcsname
220 \csname pdf#1\endcsname
221 \fi
222 }%
223 \pdfcmds@temp{resettimer}%
224 \pdfcmds@temp{elapsedtime}%
225 \fi

```

2.5 `\pdf@primitive`, `\pdf@ifprimitive`

Since version 1.40.0 pdf_{TEX} has `\pdfprimitive` and `\ifpdfprimitive`. And `\pdfprimitive` was fixed in version 1.40.4.

X_YTEX provides them under the name `\primitive` and `\ifprimitive`. Lua_{TEX} knows both name variants, but they have possibly to be enabled first (`tex.enableprimitives`).

Depending on the format TeX Live uses a prefix `luatex`.

Caution: `\let` must be used for the definition of the macros, especially because of `\ifpdfprimitive`.

2.5.1 Using Lua_{TEX}'s `tex.enableprimitives`

```

226 \ifluatex

\pdftexcmds@directlua

227 \ifnum\luatexversion<36 %
228 \def\pdftexcmds@directlua{\directlua0 }%
229 \else
230 \let\pdftexcmds@directlua\directlua
231 \fi

232 \begingroup
233 \newlinechar=10 %
234 \endlinechar=\newlinechar
235 \pdftexcmds@directlua{%
236 if tex.enableprimitives then
237 tex.enableprimitives(
238 'pdf@',
239 {'primitive', 'ifprimitive', 'pdfdraftmode', 'draftmode'}
240 )
241 tex.enableprimitives('', {'luaescapestring'})
242 end
243 }%
244 \endgroup %

245 \fi

```

2.5.2 Trying various names to find the primitives

```

\pdftexcmds@strip@prefix

246 \def\pdftexcmds@strip@prefix#1>{}

247 \def\pdftexcmds@temp#1#2#3{%
248 \begingroup\expandafter\expandafter\expandafter\endgroup
249 \expandafter\ifx\csname pdf@#1\endcsname\relax
250 \begingroup
251 \def\x{#3}%
252 \edef\x{\expandafter\pdftexcmds@strip@prefix\meaning\x}%
253 \escapechar=-1 %
254 \edef\y{\expandafter\meaning\csname#2\endcsname}%
255 \expandafter\endgroup
256 \ifx\x\y
257 \expandafter\let\csname pdf@#1\expandafter\endcsname
258 \csname #2\endcsname
259 \fi
260 \fi
261 }

\pdf@primitive

262 \pdftexcmds@temp{primitive}{pdfprimitive}{pdfprimitive}% pdfTeX, oldLuaTeX
263 \pdftexcmds@temp{primitive}{primitive}{primitive}% XeTeX, luatex
264 \pdftexcmds@temp{primitive}{luatexpprimitive}{pdfprimitive}% oldLuaTeX
265 \pdftexcmds@temp{primitive}{luatexpdfprimitive}{pdfprimitive}% oldLuaTeX

\pdf@ifprimitive

266 \pdftexcmds@temp{ifprimitive}{ifpdfprimitive}{ifpdfprimitive}% pdfTeX, oldLu-
aTeX
267 \pdftexcmds@temp{ifprimitive}{ifprimitive}{ifprimitive}% XeTeX, luatex
268 \pdftexcmds@temp{ifprimitive}{luatexifprimitive}{ifpdfprimitive}% oldLuaTeX
269 \pdftexcmds@temp{ifprimitive}{luatexifpdfprimitive}{ifpdfprimitive}% oldLuaTeX

Disable broken \pdfprimitive.

270 \ifluatex\else
271 \begingroup
272 \expandafter\ifx\csname pdf@primitive\endcsname\relax

```

```

273 \else
274 \expandafter\ifx\csname pdftexversion\endcsname\relax
275 \else
276 \ifnum\pdftexversion=140 %
277 \expandafter\ifx\csname pdftexrevision\endcsname\relax
278 \else
279 \ifnum\pdftexrevision<4 %
280 \endgroup
281 \let\pdf@primitive\@undefined
282 \@PackageInfoNoLine{pdftexcmds}{%
283 \string\pdf@primitive\space disabled, %
284 because\MessageBreak
285 \string\pdfprimitive\space is broken until pdfTeX 1.40.4%
286 }%
287 \begingroup
288 \fi
289 \fi
290 \fi
291 \fi
292 \fi
293 \endgroup
294 \fi

```

2.5.3 Result

```

295 \begingroup
296 \@PackageInfoNoLine{pdftexcmds}{%
297 \string\pdf@primitive\space is %
298 \expandafter\ifx\csname pdf@primitive\endcsname\relax not \fi
299 available%
300 }%
301 \@PackageInfoNoLine{pdftexcmds}{%
302 \string\pdf@ifprimitive\space is %
303 \expandafter\ifx\csname pdf@ifprimitive\endcsname\relax not \fi
304 available%
305 }%
306 \endgroup

```

2.6 X_gTeX

Look for primitives `\shellescape`, `\stricmp`.

```

307 \def\pdftexcmds@temp#1{%
308 \begingroup\expandafter\expandafter\expandafter\endgroup
309 \expandafter\ifx\csname pdf@#1\endcsname\relax
310 \begingroup
311 \escapechar=-1 %
312 \edef\x{\expandafter\meaning\csname#1\endcsname}%
313 \def\y{#1}%
314 \def\z##1->{%
315 \edef\y{\expandafter\z\meaning\y}%
316 \expandafter\endgroup
317 \ifx\x\y
318 \expandafter\def\csname pdf@#1\endcsname
319 \expandafter{%
320 \csname#1\endcsname
321 }%
322 \fi
323 \fi
324 }%
325 \pdftexcmds@temp{shellescape}%
326 \pdftexcmds@temp{stricmp}%

```



```

386 \ifpdf
387   \let\pdfdocmds@temp\ltx@one
388   \@PackageInfoNoLine{pdfdocmds}{\ltx@backslashchar pdfdraftmode found}%
389   \else
390     \@PackageInfoNoLine{pdfdocmds}{%
391       \ltx@backslashchar pdfdraftmode is ignored in DVI mode%
392     }%
393   \fi
394 }
395 \ifcase\pdfdocmds@temp

\pdf@draftmode

396 \let\pdf@draftmode\ltx@zero

\pdf@ifdraftmode

397 \let\pdf@ifdraftmode\ltx@secondoftwo

\pdfdocmds@setdraftmode

398 \def\pdfdocmds@setdraftmode#1{%
399 \else

\pdfdocmds@draftmode

400 \let\pdfdocmds@draftmode\pdfdraftmode

\pdf@ifdraftmode

401 \def\pdf@ifdraftmode{%
402   \ifnum\pdfdocmds@draftmode=\ltx@one
403     \expandafter\ltx@firstoftwo
404   \else
405     \expandafter\ltx@secondoftwo
406   \fi
407 }%

\pdf@draftmode

408 \def\pdf@draftmode{%
409   \ifnum\pdfdocmds@draftmode=\ltx@one
410     \expandafter\ltx@one
411   \else
412     \expandafter\ltx@zero
413   \fi
414 }%

\pdfdocmds@setdraftmode

415 \def\pdfdocmds@setdraftmode#1{%
416   \pdfdocmds@draftmode=#1\relax
417 }%

418 \fi

\pdf@setdraftmode

419 \def\pdf@setdraftmode#1{%
420   \begingroup
421     \count\ltx@cclv=#1\relax
422   \edef\x{\endgroup
423     \noexpand\pdfdocmds@@setdraftmode{\the\count\ltx@cclv}}%
424   }%
425   \x
426 }

```

`\pdfTexcmds@setdraftmode`

```
427 \def\pdfTexcmds@setdraftmode#1{%
428   \ifcase#1 %
429     \pdfTexcmds@setdraftmode{#1}%
430   \or
431     \pdfTexcmds@setdraftmode{#1}%
432   \else
433     \@PackageWarning{pdfTexcmds}{%
434       \string\pdf@setdraftmode: Ignoring\MessageBreak
435       invalid value `#1'%
436     }%
437   \fi
438 }
```

2.9 Load Lua module

```
439 \ifluatex
440 \else
441   \expandafter\pdfTexcmds@AtEnd
442 \fi%

443 \ifnum\luatexversion<80
444   \begingroup\expandafter\expandafter\expandafter\endgroup
445   \expandafter\ifx\csname RequirePackage\endcsname\relax
446     \def\TMP@RequirePackage#1[#2]{%
447       \begingroup\expandafter\expandafter\expandafter\endgroup
448       \expandafter\ifx\csname ver@#1.sty\endcsname\relax
449         \input #1.sty\relax
450       \fi
451     }%
452   \TMP@RequirePackage{luatex-loader}[2009/04/10]%
453 \else
454   \RequirePackage{luatex-loader}[2009/04/10]%
455 \fi
456 \fi
457 \pdfTexcmds@directlua{%
458   require("pdfTexcmds")%
459 }
460 \ifnum\luatexversion>37 %
461   \ifnum0%
462     \pdfTexcmds@directlua{%
463       if status.ini_version then %
464         tex.write("1")%
465       end%
466     }>0 %
467   \everyjob\expandafter{%
468     \the\everyjob
469     \pdfTexcmds@directlua{%
470       require("pdfTexcmds")%
471     }%
472   }%
473 \fi
474 \fi
475 \begingroup
476 \def\x{2017/03/19 v0.25}%
477 \ltx@onelevel@sanitize\x
478 \edef\y{%
479   \pdfTexcmds@directlua{%
480     if oberdiek.pdfTexcmds.getVersion then %
481       oberdiek.pdfTexcmds.getVersion()%
482     end%
483   }%
484 }
```

```

485 \ifx\x\y
486 \else
487 \@PackageError{pdftexcmds}{%
488   Wrong version of lua module.\MessageBreak
489   Package version: \x\MessageBreak
490   Lua module: \y
491 } \@ehc
492 \fi
493 \endgroup

```

2.10 Lua functions

2.10.1 Helper macros

`\pdftexcmds@toks`

```

494 \begingroup\expandafter\expandafter\expandafter\endgroup
495 \expandafter\ifx\csname newtoks\endcsname\relax
496 \toksdef\pdftexcmds@toks=0 %
497 \else
498 \csname newtoks\endcsname\pdftexcmds@toks
499 \fi

```

`\pdftexcmds@Patch`

```

500 \def\pdftexcmds@Patch{0}
501 \ifnum\luatexversion>40 %
502 \ifnum\luatexversion<66 %
503 \def\pdftexcmds@Patch{1}%
504 \fi
505 \fi

506 \ifcase\pdftexcmds@Patch
507 \catcode`\&=14 %
508 \else
509 \catcode`\&=9 %

```

`\pdftexcmds@PatchDecode`

```

510 \def\pdftexcmds@PatchDecode#1\@nil{%
511 \pdftexcmds@DecodeA#1^^A^^A\@nil{}%
512 }%

```

`\pdftexcmds@DecodeA`

```

513 \def\pdftexcmds@DecodeA#1^^A^^A#2\@nil#3{%
514 \ifx\relax#2\relax
515 \ltx@ReturnAfterElseFi{%
516 \pdftexcmds@DecodeB#3#1^^A^^B\@nil{}%
517 }%
518 \else
519 \ltx@ReturnAfterFi{%
520 \pdftexcmds@DecodeA#2\@nil{#3#1^^@}%
521 }%
522 \fi
523 }%

```

`\pdftexcmds@DecodeB`

```

524 \def\pdftexcmds@DecodeB#1^^A^^B#2\@nil#3{%
525 \ifx\relax#2\relax%
526 \ltx@ReturnAfterElseFi{%
527 \ltx@zero
528 #3#1%
529 }%
530 \else
531 \ltx@ReturnAfterFi{%
532 \pdftexcmds@DecodeB#2\@nil{#3#1^^A}%

```

```

533 }%
534 \fi
535 }%

536 \fi

537 \ifnum\luatexversion<36 %
538 \else
539 \catcode`\0=9 %
540 \fi

```

2.10.2 Strings [1, “7.15 Strings”]

`\pdf@strcmp`

```

541 \long\def\pdf@strcmp#1#2{%
542 \directlua0{%
543 oberdiek.pdfdocmds.strcmp("\luaescapestring{#1}",%
544 "\luaescapestring{#2}")%
545 }%
546 }%

547 \pdf@isprimitive

```

`\pdf@escapehex`

```

548 \long\def\pdf@escapehex#1{%
549 \directlua0{%
550 oberdiek.pdfdocmds.escapehex("\luaescapestring{#1}", "byte")%
551 }%
552 }%

```

`\pdf@escapehexnative`

```

553 \long\def\pdf@escapehexnative#1{%
554 \directlua0{%
555 oberdiek.pdfdocmds.escapehex("\luaescapestring{#1}")%
556 }%
557 }%

```

`\pdf@unescapehex`

```

558 \def\pdf@unescapehex#1{%
559 & \romannumeral\expandafter\pdfdocmds@PatchDecode
560 \the\expandafter\pdfdocmds@toks
561 \directlua0{%
562 oberdiek.pdfdocmds.toks="pdfdocmds@toks"%
563 oberdiek.pdfdocmds.unescapehex("\luaescapestring{#1}", "byte", \pdfdoc-
564 cmds@Patch)%
565 }%
566 & \@nil
567 }%

```

`\pdf@unescapehexnative`

```

567 \def\pdf@unescapehexnative#1{%
568 & \romannumeral\expandafter\pdfdocmds@PatchDecode
569 \the\expandafter\pdfdocmds@toks
570 \directlua0{%
571 oberdiek.pdfdocmds.toks="pdfdocmds@toks"%
572 oberdiek.pdfdocmds.unescapehex("\luaescapestring{#1}", \pdfdocmds@Patch)%
573 }%
574 & \@nil
575 }%

```

`\pdf@escapestring`

```

576 \long\def\pdf@escapestring#1{%
577 \directlua0{%

```

```

578 oberdiek.pdfcmds.escapestring("\luaescapestring{#1}", "byte")%
579 }%
580 }

```

`\pdf@escapename`

```

581 \long\def\pdf@escapename#1{%
582 \directlua0{%
583 oberdiek.pdfcmds.escapename("\luaescapestring{#1}", "byte")%
584 }%
585 }

```

`\pdf@escapenamename`

```

586 \long\def\pdf@escapenamename#1{%
587 \directlua0{%
588 oberdiek.pdfcmds.escapename("\luaescapestring{#1}")%
589 }%
590 }

```

2.10.3 Files [1, “7.18 Files”]

`\pdf@filesize`

```

591 \def\pdf@filesize#1{%
592 \directlua0{%
593 oberdiek.pdfcmds.filesize("\luaescapestring{#1}")%
594 }%
595 }

```

`\pdf@filemoddate`

```

596 \def\pdf@filemoddate#1{%
597 \directlua0{%
598 oberdiek.pdfcmds.filemoddate("\luaescapestring{#1}")%
599 }%
600 }

```

`\pdf@filedump`

```

601 \def\pdf@filedump#1#2#3{%
602 \directlua0{%
603 oberdiek.pdfcmds.filedump("\luaescapestring{\number#1}",%
604 "\luaescapestring{\number#2}",%
605 "\luaescapestring{#3}")%
606 }%
607 }%

```

`\pdf@mdfivesum`

```

608 \long\def\pdf@mdfivesum#1{%
609 \directlua0{%
610 oberdiek.pdfcmds.mdfivesum("\luaescapestring{#1}", "byte")%
611 }%
612 }%

```

`\pdf@mdfivesumname`

```

613 \long\def\pdf@mdfivesumname#1{%
614 \directlua0{%
615 oberdiek.pdfcmds.mdfivesum("\luaescapestring{#1}")%
616 }%
617 }%

```

`\pdf@filemdfivesum`

```

618 \def\pdf@filemdfivesum#1{%
619 \directlua0{%
620 oberdiek.pdfcmds.filemdfivesum("\luaescapestring{#1}")%
621 }%
622 }%

```

2.10.4 Timekeeping [1, “7.17 Timekeeping”]

`\protected`

```
623 \let\pdf texcmds@temp=Y%
624 \begingroup\expandafter\expandafter\expandafter\endgroup
625 \expandafter\ifx\csname protected\endcsname\relax
626 \pdf texcmds@directlua0{%
627   if tex.enableprimitives then %
628     tex.enableprimitives(', {'protected'})%
629   end%
630 }%
631 \fi
632 \begingroup\expandafter\expandafter\expandafter\endgroup
633 \expandafter\ifx\csname protected\endcsname\relax
634 \let\pdf texcmds@temp=N%
635 \fi
```

`\numexpr`

```
636 \begingroup\expandafter\expandafter\expandafter\endgroup
637 \expandafter\ifx\csname numexpr\endcsname\relax
638 \pdf texcmds@directlua0{%
639   if tex.enableprimitives then %
640     tex.enableprimitives(', {'numexpr'})%
641   end%
642 }%
643 \fi
644 \begingroup\expandafter\expandafter\expandafter\endgroup
645 \expandafter\ifx\csname numexpr\endcsname\relax
646 \let\pdf texcmds@temp=N%
647 \fi

648 \ifx\pdf texcmds@temp N%
649 \@PackageWarningNoLine{pdf texcmds}{%
650   Definitions of \ltx@backslashchar pdf@resettimer and%
651   \MessageBreak
652   \ltx@backslashchar pdf@elapsedtime are skipped, because%
653   \MessageBreak
654   e-TeX's \ltx@backslashchar protected or %
655   \ltx@backslashchar numexpr are missing%
656 }%
657 \else
```

`\pdf@resettimer`

```
658 \protected\def\pdf@resettimer{%
659   \pdf texcmds@directlua0{%
660     oberdiek.pdf texcmds.resettimer()%
661   }%
662 }%
```

`\pdf@elapsedtime`

```
663 \protected\def\pdf@elapsedtime{%
664   \numexpr
665   \pdf texcmds@directlua0{%
666     oberdiek.pdf texcmds.elapsedtime()%
667   }%
668   \relax
669 }%

670 \fi
```

2.10.5 Shell escape

`\pdf@shellescape`

```
671 \ifnum\luatexversion<68 %
672 \else
673 \protected\edef\pdf@shellescape{%
674 \numexpr\directlua{tex.sprint(status.shell_escape)}\relax}
675 \fi
```

`\pdf@system`

```
676 \def\pdf@system#1{%
677 \directlua0{%
678 oberdiek.pdfcmds.system("\luaescapestring{#1}")%
679 }%
680 }
```

`\pdf@lastsystemstatus`

```
681 \def\pdf@lastsystemstatus{%
682 \directlua0{%
683 oberdiek.pdfcmds.lastsystemstatus()%
684 }%
685 }
```

`\pdf@lastsystemexit`

```
686 \def\pdf@lastsystemexit{%
687 \directlua0{%
688 oberdiek.pdfcmds.lastsystemexit()%
689 }%
690 }
```

```
691 \catcode`\0=12 %
```

`\pdf@pipe` Check availability of `io.popen` first.

```
692 \ifnum0%
693 \pdfcmds@directlua{%
694 if io.popen then %
695 tex.write("1")%
696 end%
697 }%
698 =1 %
699 \def\pdf@pipe#1{%
700 & \romannumeral\expandafter\pdfcmds@PatchDecode
701 \the\expandafter\pdfcmds@toks
702 \pdfcmds@directlua{%
703 oberdiek.pdfcmds.toks="pdfcmds@toks"%
704 oberdiek.pdfcmds.pipe("\luaescapestring{#1}", \pdfcmds@Patch)%
705 }%
706 & \@nil
707 }%
708 \fi

709 \pdfcmds@AtEnd%
710 </package>
```

2.11 Lua module

```
711 <*lua>
712 module("oberdiek.pdfcmds", package.seeall)
713 local systemexitstatus
714 function getversion()
715 tex.write("2017/03/19 v0.25")
716 end
```

2.11.1 Strings [1, “7.15 Strings”]

```
717 function strcmp(A, B)
718   if A == B then
719     tex.write("0")
720   elseif A < B then
721     tex.write("-1")
722   else
723     tex.write("1")
724   end
725 end
726 local function utf8_to_byte(str)
727   local i = 0
728   local n = string.len(str)
729   local t = {}
730   while i < n do
731     i = i + 1
732     local a = string.byte(str, i)
733     if a < 128 then
734       table.insert(t, string.char(a))
735     else
736       if a >= 192 and i < n then
737         i = i + 1
738         local b = string.byte(str, i)
739         if b < 128 or b >= 192 then
740           i = i - 1
741         elseif a == 194 then
742           table.insert(t, string.char(b))
743         elseif a == 195 then
744           table.insert(t, string.char(b + 64))
745         end
746       end
747     end
748   end
749   return table.concat(t)
750 end
751 function escapehex(str, mode)
752   if mode == "byte" then
753     str = utf8_to_byte(str)
754   end
755   tex.write((string.gsub(str, ".",
756     function (ch)
757       return string.format("%02X", string.byte(ch))
758     end
759   )))
760 end
```

See procedure `unescapehex` in file `utils.c` of `pdfTeX`. Caution: `tex.write` ignores leading spaces.

```
761 function unescapehex(str, mode, patch)
762   local a = 0
763   local first = true
764   local result = {}
765   for i = 1, string.len(str), 1 do
766     local ch = string.byte(str, i)
767     if ch >= 48 and ch <= 57 then
768       ch = ch - 48
769     elseif ch >= 65 and ch <= 70 then
770       ch = ch - 55
771     elseif ch >= 97 and ch <= 102 then
772       ch = ch - 87
773     else
774       ch = nil
```

```

775 end
776 if ch then
777   if first then
778     a = ch * 16
779     first = false
780   else
781     table.insert(result, a + ch)
782     first = true
783   end
784 end
785 end
786 if not first then
787   table.insert(result, a)
788 end
789 if patch == 1 then
790   local temp = {}
791   for i, a in ipairs(result) do
792     if a == 0 then
793       table.insert(temp, 1)
794       table.insert(temp, 1)
795     else
796       if a == 1 then
797         table.insert(temp, 1)
798         table.insert(temp, 2)
799       else
800         table.insert(temp, a)
801       end
802     end
803   end
804   result = temp
805 end
806 if mode == "byte" then
807   local utf8 = {}
808   for i, a in ipairs(result) do
809     if a < 128 then
810       table.insert(utf8, a)
811     else
812       if a < 192 then
813         table.insert(utf8, 194)
814         a = a - 128
815       else
816         table.insert(utf8, 195)
817         a = a - 192
818       end
819       table.insert(utf8, a + 128)
820     end
821   end
822   result = utf8
823 end

```

this next line added for current luatex; this is the only change in the file. eroux,
28apr13. (v 0.21)

```

824 local unpack = _G["unpack"] or table.unpack
825 tex.settoks(toks, string.char(unpack(result)))
826 end

```

See procedure `escapestring` in file `utils.c` of `pdfTeX`.

```

827 function escapestring(str, mode)
828   if mode == "byte" then
829     str = utf8_to_byte(str)
830   end
831   tex.write((string.gsub(str, ".",
832     function (ch)
833       local b = string.byte(ch)

```

```

834     if b < 33 or b > 126 then
835         return string.format("\\%.3o", b)
836     end
837     if b == 40 or b == 41 or b == 92 then
838         return "\\\" .. ch
839     end

```

Lua 5.1 returns the match in case of return value nil.

```

840     return nil
841 end
842 )))
843 end

```

See procedure `escapename` in file `utils.c` of pdfTeX.

```

844 function escapename(str, mode)
845     if mode == "byte" then
846         str = utf8_to_byte(str)
847     end
848     tex.write((string.gsub(str, ".",
849         function (ch)
850             local b = string.byte(ch)
851             if b == 0 then

```

In Lua 5.0 nil could be used for the empty string, But nil returns the match in Lua 5.1, thus we use the empty string explicitly.

```

852         return ""
853     end
854     if b <= 32 or b >= 127
855         or b == 35 or b == 37 or b == 40 or b == 41
856         or b == 47 or b == 60 or b == 62 or b == 91
857         or b == 93 or b == 123 or b == 125 then
858         return string.format("#%.2X", b)
859     else

```

Lua 5.1 returns the match in case of return value nil.

```

860     return nil
861 end
862 end
863 )))
864 end

```

2.11.2 Files [1, “7.18 Files”]

```

865 function filesize(filename)
866     local foundfile = kpse.find_file(filename, "tex", true)
867     if foundfile then
868         local size = lfs.attributes(foundfile, "size")
869         if size then
870             tex.write(size)
871         end
872     end
873 end

```

See procedure `makepdftime` in file `utils.c` of pdfTeX.

```

874 function filemoddate(filename)
875     local foundfile = kpse.find_file(filename, "tex", true)
876     if foundfile then
877         local date = lfs.attributes(foundfile, "modification")
878         if date then
879             local d = os.date("!*t", date)
880             if d.sec >= 60 then
881                 d.sec = 59
882             end
883             local u = os.date("!*t", date)
884             local off = 60 * (d.hour - u.hour) + d.min - u.min
885             if d.year ~= u.year then
886                 if d.year > u.year then

```

```

887     off = off + 1440
888     else
889     off = off - 1440
890     end
891     elseif d.yday ~= u.yday then
892     if d.yday > u.yday then
893     off = off + 1440
894     else
895     off = off - 1440
896     end
897     end
898     local timezone
899     if off == 0 then
900     timezone = "Z"
901     else
902     local hours = math.floor(off / 60)
903     local mins = math.abs(off - hours * 60)
904     timezone = string.format("%+03d'%02d'", hours, mins)
905     end
906     tex.write(string.format("D:%04d%02d%02d%02d%02d%02d%s",
907     d.year, d.month, d.day, d.hour, d.min, d.sec, timezone))
908     end
909 end
910 end
911 function filedump(offset, length, filename)
912 length = tonumber(length)
913 if length and length > 0 then
914 local foundfile = kpse.find_file(filename, "tex", true)
915 if foundfile then
916 offset = tonumber(offset)
917 if not offset then
918 offset = 0
919 end
920 local filehandle = io.open(foundfile, "r")
921 if filehandle then
922 if offset > 0 then
923 filehandle:seek("set", offset)
924 end
925 local dump = filehandle:read(length)
926 escapehex(dump)
927 filehandle:close()
928 end
929 end
930 end
931 end
932 function mdfivesum(str, mode)
933 if mode == "byte" then
934 str = utf8_to_byte(str)
935 end
936 escapehex(md5.sum(str))
937 end
938 function filemdfivesum(filename)
939 local foundfile = kpse.find_file(filename, "tex", true)
940 if foundfile then
941 local filehandle = io.open(foundfile, "r")
942 if filehandle then
943 local contents = filehandle:read("*a")
944 escapehex(md5.sum(contents))
945 filehandle:close()
946 end
947 end
948 end

```

2.11.3 Timekeeping [1, “7.17 Timekeeping”]

The functions for timekeeping are based on Andy Thomas’ work [3]. Changes:

- Overflow check is added.
- `string.format` is used to avoid exponential number representation for sure.
- `tex.write` is used instead of `tex.print` to get tokens with catcode 12 and without appended `\endlinechar`.

```
949 local basetime = 0
950 function resettimer()
951   basetime = os.clock()
952 end
953 function elapsedtime()
954   local val = (os.clock() - basetime) * 65536 + .5
955   if val > 2147483647 then
956     val = 2147483647
957   end
958   tex.write(string.format("%d", val))
959 end
```

2.11.4 Miscellaneous [1, “7.21 Miscellaneous”]

```
960 function shellescape()
961   if os.execute then
962     if status
963       and status.luatex_version
964       and status.luatex_version >= 68 then
965       tex.write(os.execute())
966     else
967       local result = os.execute()
968       if result == 0 then
969         tex.write("0")
970       else
971         if result == nil then
972           tex.write("0")
973         else
974           tex.write("1")
975         end
976       end
977     end
978   else
979     tex.write("0")
980   end
981 end
982 function system(cmdline)
983   systemexitstatus = nil
984   texio.write_nl("log", "system(" .. cmdline .. ") ")
985   if os.execute then
986     texio.write("log", "executed.")
987     systemexitstatus = os.execute(cmdline)
988   else
989     texio.write("log", "disabled.")
990   end
991 end
992 function lastsystemstatus()
993   local result = tonumber(systemexitstatus)
994   if result then
995     local x = math.floor(result / 256)
996     tex.write(result - 256 * math.floor(result / 256))
997   end
998 end
```

```

999 function lastsysteemexit()
1000 local result = tonumber(systemexitstatus)
1001 if result then
1002   tex.write(math.floor(result / 256))
1003 end
1004 end
1005 function pipe(cmdline, patch)
1006 local result
1007 systeemexitstatus = nil
1008 texio.write_nl("log", "pipe(" .. cmdline .. ") ")
1009 if io.popen then
1010   texio.write("log", "executed.")
1011   local handle = io.popen(cmdline, "r")
1012   if handle then
1013     result = handle.read("*a")
1014     handle:close()
1015   end
1016 else
1017   texio.write("log", "disabled.")
1018 end
1019 if result then
1020   if patch == 1 then
1021     local temp = {}
1022     for i, a in ipairs(result) do
1023       if a == 0 then
1024         table.insert(temp, 1)
1025         table.insert(temp, 1)
1026       else
1027         if a == 1 then
1028           table.insert(temp, 1)
1029           table.insert(temp, 2)
1030         else
1031           table.insert(temp, a)
1032         end
1033       end
1034     end
1035     result = temp
1036   end
1037   tex.settoks(toks, result)
1038 else
1039   tex.settoks(toks, "")
1040 end
1041 end
1042  $\langle$ /lua $\rangle$ 

```

3 Test

3.1 Catcode checks for loading

```

1043  $\langle$ *test1 $\rangle$ 
1044 \catcode`\{=1 %
1045 \catcode`\}=2 %
1046 \catcode`\#=6 %
1047 \catcode`\@=11 %
1048 \expandafter\ifx\csname count@\endcsname\relax
1049 \countdef\count@=255 %
1050 \fi
1051 \expandafter\ifx\csname @gobble\endcsname\relax
1052 \long\def\@gobble#1{#1}%
1053 \fi
1054 \expandafter\ifx\csname @firstofone\endcsname\relax
1055 \long\def\@firstofone#1{#1}%

```

```

1056 \fi
1057 \expandafter\ifx\csname loop\endcsname\relax
1058 \expandafter\@firstofone
1059 \else
1060 \expandafter\@gobble
1061 \fi
1062 {%
1063 \def\loop#1\repeat{%
1064 \def\body{#1}%
1065 \iterate
1066 }%
1067 \def\iterate{%
1068 \body
1069 \let\next\iterate
1070 \else
1071 \let\next\relax
1072 \fi
1073 \next
1074 }%
1075 \let\repeat=\fi
1076 }%
1077 \def\RestoreCatcodes{}
1078 \count@=0 %
1079 \loop
1080 \edef\RestoreCatcodes{%
1081 \RestoreCatcodes
1082 \catcode\the\count@=\the\catcode\count@\relax
1083 }%
1084 \ifnum\count@<255 %
1085 \advance\count@ 1 %
1086 \repeat
1087
1088 \def\RangeCatcodeInvalid#1#2{%
1089 \count@=#1\relax
1090 \loop
1091 \catcode\count@=15 %
1092 \ifnum\count@<#2\relax
1093 \advance\count@ 1 %
1094 \repeat
1095 }
1096 \def\RangeCatcodeCheck#1#2#3{%
1097 \count@=#1\relax
1098 \loop
1099 \ifnum#3=\catcode\count@
1100 \else
1101 \errmessage{%
1102 Character \the\count@\space
1103 with wrong catcode \the\catcode\count@\space
1104 instead of \number#3%
1105 }%
1106 \fi
1107 \ifnum\count@<#2\relax
1108 \advance\count@ 1 %
1109 \repeat
1110 }
1111 \def\space{ }
1112 \expandafter\ifx\csname LoadCommand\endcsname\relax
1113 \def\LoadCommand{\input pdftexcmds.sty\relax}%
1114 \fi
1115 \def\Test{%
1116 \RangeCatcodeInvalid{0}{47}%
1117 \RangeCatcodeInvalid{58}{64}%

```

```

1118 \RangeCatcodeInvalid{91}{96}%
1119 \RangeCatcodeInvalid{123}{255}%
1120 \catcode`\@=12 %
1121 \catcode`\=0 %
1122 \catcode`\%=14 %
1123 \LoadCommand
1124 \RangeCatcodeCheck{0}{36}{15}%
1125 \RangeCatcodeCheck{37}{37}{14}%
1126 \RangeCatcodeCheck{38}{47}{15}%
1127 \RangeCatcodeCheck{48}{57}{12}%
1128 \RangeCatcodeCheck{58}{63}{15}%
1129 \RangeCatcodeCheck{64}{64}{12}%
1130 \RangeCatcodeCheck{65}{90}{11}%
1131 \RangeCatcodeCheck{91}{91}{15}%
1132 \RangeCatcodeCheck{92}{92}{0}%
1133 \RangeCatcodeCheck{93}{96}{15}%
1134 \RangeCatcodeCheck{97}{122}{11}%
1135 \RangeCatcodeCheck{123}{255}{15}%
1136 \RestoreCatcodes
1137 }
1138 \Test
1139 \csname @@end\endcsname
1140 \end
1141 </test1>

```

3.2 Test for \pdf@isprimitive

```

1142 <*test2>
1143 \catcode`\{=1 %
1144 \catcode`\}=2 %
1145 \catcode`\#=6 %
1146 \catcode`\@=11 %
1147 \input pdftexcmds.sty\relax
1148 \def\msg#1{%
1149   \begingroup
1150     \escapechar=92 %
1151     \immediate\write16{#1}%
1152   \endgroup
1153 }
1154 \long\def\test#1#2#3#4{%
1155   \begingroup
1156     #4%
1157   \def\str{%
1158     Test \string\pdf@isprimitive
1159     {\string #1}{\string #2}{...}: %
1160   }%
1161   \pdf@isprimitive{#1}{#2}{%
1162     \ifx#3Y%
1163       \msg{\str true ==> OK.}%
1164     \else
1165       \errmessage{\str false ==> FAILED}%
1166     \fi
1167   }{%
1168     \ifx#3Y%
1169       \errmessage{\str true ==> FAILED}%
1170     \else
1171       \msg{\str false ==> OK.}%
1172     \fi
1173   }%
1174 \endgroup
1175 }
1176 \test\relax\relax Y{}
1177 \test\foobar\relax Y{\let\foobar\relax}

```

```

1178 \test\foobar\relax N{}
1179 \test\hbox\hbox Y{}
1180 \test\foobar@hbox\hbox Y{\let\foobar@hbox\hbox}
1181 \test\if\if Y{}
1182 \test\if\ifx N{}
1183 \test\ifx\if N{}
1184 \test\par\par Y{}
1185 \test\hbox\par N{}
1186 \test\par\hbox N{}
1187 \csname @@end\endcsname\end
1188 </test2>

```

3.3 Test for \pdf@shellescape

```

1189 <*test-shell>
1190 \catcode`\{=1 %
1191 \catcode`\}=2 %
1192 \catcode`\#=6 %
1193 \catcode`\@=11 %
1194 \input pdftexcmds.sty\relax
1195 \def\msg#{\immediate\write16}
1196 \def\MaybeEnd{}
1197 \ifx\luatexversion\UnDeFiNeD
1198 \else
1199 \ifnum\luatexversion<68 %
1200 \ifx\pdf@shellescape\@undefined
1201 \msg{SHELL=U}%
1202 \msg{OK (LuaTeX < 0.68)}%
1203 \else
1204 \msg{SHELL=defined}%
1205 \errmessage{Failed (LuaTeX < 0.68)}%
1206 \fi
1207 \def\MaybeEnd{\csname @@end\endcsname\end}%
1208 \fi
1209 \fi
1210 \MaybeEnd
1211 \ifx\pdf@shellescape\@undefined
1212 \msg{SHELL=U}%
1213 \else
1214 \msg{SHELL=\number\pdf@shellescape}%
1215 \fi
1216 \ifx\expected\@undefined
1217 \else
1218 \ifx\expected\relax
1219 \msg{EXPECTED=U}%
1220 \ifx\pdf@shellescape\@undefined
1221 \msg{OK}%
1222 \else
1223 \errmessage{Failed}%
1224 \fi
1225 \else
1226 \msg{EXPECTED=\number\expected}%
1227 \ifnum\pdf@shellescape=\expected\relax
1228 \msg{OK}%
1229 \else
1230 \errmessage{Failed}%
1231 \fi
1232 \fi
1233 \fi
1234 \csname @@end\endcsname\end
1235 </test-shell>

```

3.4 Test for escape functions

```

1236 (*test-escape)
1237 \catcode`\{=1 %
1238 \catcode`\}=2 %
1239 \catcode`\#=6 %
1240 \catcode`\^=7 %
1241 \catcode`\@=11 %
1242 \errorcontextlines=1000 %
1243 \input pdftexcmds.sty\relax
1244 \def\msg#1{%
1245   \begingroup
1246     \escapechar=92 %
1247     \immediate\write16{#1}%
1248   \endgroup
1249 }

1250 \begingroup
1251   \catcode`\@=11 %
1252   \countdef\count@=255 %
1253   \def\space{ }%
1254   \long\def\@whilenum#1\do #2{%
1255     \ifnum #1\relax
1256       #2\relax
1257     \@iwhilenum{#1\relax#2\relax}%
1258   \fi
1259 }%
1260 \long\def\@iwhilenum#1{%
1261   \ifnum #1%
1262     \expandafter\@iwhilenum
1263   \else
1264     \expandafter\ltx@gobble
1265   \fi
1266   {#1}%
1267 }%
1268 \gdef\AllBytes{}%
1269 \count@=0 %
1270 \catcode0=12 %
1271 \@whilenum\count@<256 \do{%
1272   \lccode0=\count@
1273   \ifnum\count@=32 %
1274     \xdef\AllBytes{\AllBytes\space}%
1275   \else
1276     \lowercase{%
1277       \xdef\AllBytes{\AllBytes^^@}%
1278     }%
1279   \fi
1280   \advance\count@ by 1 %
1281 }%
1282 \endgroup

1283 \def\AllBytesHex{%
1284 000102030405060708090A0B0C0D0E0F%
1285 101112131415161718191A1B1C1D1E1F%
1286 202122232425262728292A2B2C2D2E2F%
1287 303132333435363738393A3B3C3D3E3F%
1288 404142434445464748494A4B4C4D4E4F%
1289 505152535455565758595A5B5C5D5E5F%
1290 606162636465666768696A6B6C6D6E6F%
1291 707172737475767778797A7B7C7D7E7F%
1292 808182838485868788898A8B8C8D8E8F%
1293 909192939495969798999A9B9C9D9E9F%
1294 A0A1A2A3A4A5A6A7A8A9AAABACADAEAF%
1295 B0B1B2B3B4B5B6B7B8B9BABBBCBDBEBF%
1296 C0C1C2C3C4C5C6C7C8C9CACBCCCDCECF%
1297 D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF%

```

```

1298 E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEF%
1299 F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF%
1300 }
1301 \ltx@onelevel@sanitize\AllBytesHex
1302 \expandafter\lowercase\expandafter{%
1303 \expandafter\def\expandafter\AllBytesHexLC
1304 \expandafter{\AllBytesHex}%
1305 }
1306 \begingroup
1307 \catcode`\#=12 %
1308 \xdef\AllBytesName{%
1309 #01#02#03#04#05#06#07#08#09#0A#0B#0C#0D#0E#0F%
1310 #10#11#12#13#14#15#16#17#18#19#1A#1B#1C#1D#1E#1F%
1311 #20!"#23$#25&'#28#29*+,-.#2F%
1312 0123456789:;#3C=#3E?%
1313 @ABCDEFGHIJKLMNO%
1314 PQRSTUVWXYZ#5B\ltx@backslashchar#5D^_%
1315 `abcdefghijklmnop%
1316 pqrstuvwxyz#7B|#7D\string~#7F%
1317 #80#81#82#83#84#85#86#87#88#89#8A#8B#8C#8D#8E#8F%
1318 #90#91#92#93#94#95#96#97#98#99#9A#9B#9C#9D#9E#9F%
1319 #A0#A1#A2#A3#A4#A5#A6#A7#A8#A9#AA#AB#AC#AD#AE#AF%
1320 #B0#B1#B2#B3#B4#B5#B6#B7#B8#B9#BA#BB#BC#BD#BE#BF%
1321 #C0#C1#C2#C3#C4#C5#C6#C7#C8#C9#CA#CB#CC#CD#CE#CF%
1322 #D0#D1#D2#D3#D4#D5#D6#D7#D8#D9#DA#DB#DC#DD#DE#DF%
1323 #E0#E1#E2#E3#E4#E5#E6#E7#E8#E9#EA#EB#EC#ED#EE#EF%
1324 #F0#F1#F2#F3#F4#F5#F6#F7#F8#F9#FA#FB#FC#FD#FE#FF%
1325 }%
1326 \endgroup
1327 \ltx@onelevel@sanitize\AllBytesName
1328 \edef\AllBytesFromName{\expandafter\ltx@gobble\AllBytes}
1329 \begingroup
1330 \def\{|}%
1331 \edef%\{\ltx@percentchar}%
1332 \catcode`\|=0 %
1333 \catcode`\#=12 %
1334 \catcode`\~ =12 %
1335 \catcode`\ =12 %
1336 |xdef|AllBytesString{%
1337 \000\001\002\003\004\005\006\007\010\011\012\013\014\015\016\017%
1338 \020\021\022\023\024\025\026\027\030\031\032\033\034\035\036\037%
1339 \040!"#$%&'(\)*+,-./%
1340 0123456789:;<=>?%
1341 @ABCDEFGHIJKLMNO%
1342 PQRSTUVWXYZ[\]^_ %
1343 `abcdefghijklmnop%
1344 pqrstuvwxyz{|}~\177%
1345 \200\201\202\203\204\205\206\207\210\211\212\213\214\215\216\217%
1346 \220\221\222\223\224\225\226\227\230\231\232\233\234\235\236\237%
1347 \240\241\242\243\244\245\246\247\250\251\252\253\254\255\256\257%
1348 \260\261\262\263\264\265\266\267\270\271\272\273\274\275\276\277%
1349 \300\301\302\303\304\305\306\307\310\311\312\313\314\315\316\317%
1350 \320\321\322\323\324\325\326\327\330\331\332\333\334\335\336\337%
1351 \340\341\342\343\344\345\346\347\350\351\352\353\354\355\356\357%
1352 \360\361\362\363\364\365\366\367\370\371\372\373\374\375\376\377%
1353 }%
1354 |endgroup
1355 \ltx@onelevel@sanitize\AllBytesString
1356 \def\Test#1#2#3{%
1357 \begingroup
1358 \expandafter\expandafter\expandafter\def
1359 \expandafter\expandafter\expandafter\TestResult

```

```

1360 \expandafter\expandafter\expandafter{%
1361   #1{#2}%
1362 }%
1363 \ifx\TestResult#3%
1364 \else
1365   \newlinechar=10 %
1366   \msg{Expect:^^J#3}%
1367   \msg{Result:^^J\TestResult}%
1368   \errmessage{\string#2 -\string#1-> \string#3}%
1369 \fi
1370 \endgroup
1371 }
1372 \def\test#1#2#3{%
1373 \edef\TestFrom{#2}%
1374 \edef\TestExpect{#3}%
1375 \ltx@onelevel@sanitize\TestExpect
1376 \Test#1\TestFrom\TestExpect
1377 }
1378 \test\pdf@unescapehex{74657374}{test}
1379 \begingroup
1380 \catcode0=12 %
1381 \catcode1=12 %
1382 \test\pdf@unescapehex{740074017400740174}{t^^@t^^At^^@t^^At}%
1383 \endgroup
1384 \Test\pdf@escapehex\AllBytes\AllBytesHex
1385 \Test\pdf@unescapehex\AllBytesHex\AllBytes
1386 \Test\pdf@escapename\AllBytes\AllBytesName
1387 \Test\pdf@escapestring\AllBytes\AllBytesString
1388 \csname @@end\endcsname\end
1389 </test-escape>

```

4 Installation

4.1 Download

Package. This package is available on CTAN¹:

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

[CTAN:macros/latex/contrib/oberdiek/pdftexcmds.pdf](http://ctan.org/pkg/oberdiek/pdftexcmds.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](http://ctan.org/pkg/oberdiek.tds.zip)

TDS refers to the standard “A Directory Structure for T_EX Files” ([CTAN:tds/tds.pdf](http://ctan.org/pkg/tds)). 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
```

¹<http://ctan.org/pkg/pdftexcmds>

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 pdftexcmds.dtx
```

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

```
pdftexcmds.sty          → tex/generic/oberdiek/pdftexcmds.sty
oberdiek.pdftexcmds.lua → scripts/oberdiek/oberdiek.pdftexcmds.lua
pdftexcmds.lua         → scripts/oberdiek/pdftexcmds.lua
pdftexcmds.pdf         → doc/latex/oberdiek/pdftexcmds.pdf
test/pdftexcmds-test1.tex → doc/latex/oberdiek/test/pdftexcmds-test1.tex
test/pdftexcmds-test2.tex → doc/latex/oberdiek/test/pdftexcmds-test2.tex
test/pdftexcmds-test-shell.tex → doc/latex/oberdiek/test/pdftexcmds-test-shell.tex
test/pdftexcmds-test-escape.tex → doc/latex/oberdiek/test/pdftexcmds-test-escape.tex
pdftexcmds.dtx         → source/latex/oberdiek/pdftexcmds.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

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{pdftexcmds.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 pdftexcmds.dtx
bibtex pdftexcmds.aux
makeindex -s gind.ist pdftexcmds.idx
pdflatex pdftexcmds.dtx
makeindex -s gind.ist pdftexcmds.idx
pdflatex pdftexcmds.dtx
```

5 Catalogue

The following XML file can be used as source for the [T_EX Catalogue](#). The elements `caption` and `description` are imported from the original XML file from the Catalogue. The name of the XML file in the Catalogue is `pdftexcmds.xml`.

```
1390 <*catalogue>
1391 <?xml version='1.0' encoding='us-ascii'?>
1392 <!DOCTYPE entry SYSTEM 'catalogue.dtd'>
1393 <entry datestamp='$Date$' modifier='$Author$' id='pdftexcmds'>
1394   <name>pdftexcmds</name>
1395   <caption>LuaTeX support for pdfTeX utility functions.</caption>
1396   <authorref id='auth:oberdiek' />
1397   <copyright owner='Heiko Oberdiek' year='2007,2009-2011' />
1398   <license type='lppl1.3' />
1399   <version number='0.20' />
1400   <description>
1401     LuaTeX provides most of the commands of
1402     <xref refid='pdftex'>pdfTeX</xref> 1.40. However, a number of
1403     utility functions are not available. This package tries to fill
1404     the gap and implements some of the missing primitives using Lua.
1405     <p />
1406     The package is part of the <xref refid='oberdiek'>oberdiek</xref>
1407     bundle.
1408   </description>
1409   <documentation details='Package documentation'
1410     href='ctan:/macros/latex/contrib/oberdiek/pdftexcmds.pdf' />
1411   <ctan file='true' path='/macros/latex/contrib/oberdiek/pdftexcmds.dtx' />
1412   <miktex location='oberdiek' />
1413   <texlive location='oberdiek' />
1414   <install path='/macros/latex/contrib/oberdiek/oberdiek.tds.zip' />
1415 </entry>
1416 </catalogue>
```

6 References

- [1] Hàn Thê Thành et al. *The pdfT_EX user manual*. Version 655 (1.40.11). 2010-11-23. URL: <http://mirror.ctan.org/systems/pdftex/manual/pdftex-a.pdf> (visited on 2011-11-29).
- [2] LuaT_EX development team. *LuaT_EX Reference*. Version beta 0.71.0. 2011-10-11. URL: <http://www.luatex.org/svn/trunk/manual/luatexref-t.pdf> (visited on 2011-11-29).
- [3] Andy Thomas. *Analog of \pdfelapsedtime for LuaT_EX and X_YT_EX*. URL: <http://tex.stackexchange.com/a/32531> (visited on 2011-11-29).

7 History

[2007/11/11 v0.1]

- First version.

[2007/11/12 v0.2]

- Short description fixed.

[2007/12/12 v0.3]

- Organization of Lua code as module.

[2009/04/10 v0.4]

- Adaptation for syntax change of `\directlua` in LuaTeX 0.36.

[2009/09/22 v0.5]

- `\pdf@primitive`, `\pdf@ifprimitive` added.
- X_YTeX's variants are detected for `\pdf@shellescape`, `\pdf@stricmp`, `\pdf@primitive`, `\pdf@ifprimitive`.

[2009/09/23 v0.6]

- Macro `\pdf@isprimitive` added.

[2009/12/12 v0.7]

- Short info shortened.

[2010/03/01 v0.8]

- Required date for package `ifluatex` updated.

[2010/04/01 v0.9]

- Use `\ifeof18` for defining `\pdf@shellescape` between pdfTeX 1.21a (inclusive) and 1.30.0 (exclusive).

[2010/11/04 v0.10]

- `\pdf@draftmode`, `\pdf@ifdraftmode` and `\pdf@setdraftmode` added.

[2010/11/11 v0.11]

- Missing `\RequirePackage` for package `ifpdf` added.

[2011/01/30 v0.12]

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

[2011/03/04 v0.13]

- Improved Lua function `shellescape` that also uses the result of `os.execute()` (thanks to Philipp Stephani).

[2011/04/10 v0.14]

- Version check of loaded module added.
- Patch for bug in LuaTeX between 0.40.6 and 0.65 that is fixed in revision 4096.

[2011/04/16 v0.15]

- LuaTeX: `\pdf@shellescape` is only supported for version 0.70.0 and higher due to a bug, `os.execute()` crashes in some circumstances. Fixed in LuaTeX beta-0.70.0, revision 4167.

[2011/04/22 v0.16]

- Previous fix was not working due to a wrong catcode of digit zero (due to easily support the old `\directlua0`). The version border is lowered to 0.68, because some beta-0.67.0 seems also to work.

[2011/06/29 v0.17]

- Documentation addition to `\pdf@shellescape`.

[2011/07/01 v0.18]

- Add Lua module loading in `\everyjob` for `iniTeX` (LuaTeX only).

[2011/07/28 v0.19]

- Missing space in an info message added (Martin Münch).

[2011/11/29 v0.20]

- `\pdf@resettimer` and `\pdf@elapsedtime` added (thanks Andy Thomas).

[2016/05/10 v0.21]

- local `unpack` added (thanks Élie Roux).

[2016/05/21 v0.22]

- adjust `\textbackslash` usage in bib file for biber bug.

[2016/10/02 v0.23]

- add `file.close` to lua filehandles (github pull request).

[2017/01/29 v0.24]

- Avoid loading `luatex-loader` for current `luatex`. (Use `pdftexcmds.lua` not `oberdiek.pdfcmds.lua` to simplify file search with standard require)

[2017/03/19 v0.25]

- New `\pdf@shellescape` for LuaTeX, see github issue 20.

8 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>\@PackageWarning</code>	433
<code>\#</code>	<code>\@PackageWarningNoLine</code>	649
<code>\%</code>	<code>\@ehc</code>	491
<code>\&</code>	<code>\@firstofone</code>	1055, 1058
<code>\(</code>	<code>\@gobble</code>	1052, 1060
<code>\)</code>	<code>\@iwhilenum</code>	1257, 1260, 1262
<code>\@</code>	<code>\@nil</code>	510, 511, 513,
<code>\@PackageError</code>	516, 520, 524, 532, 565, 574, 706	
<code>\@PackageInfoNoLine</code>	<code>\@undefined</code>	58,
154, 282, 296, 301, 384, 388, 390	281, 376, 1200, 1211, 1216, 1220	

<code>\LoadCommand</code>	1113, 1123	<code>\pdf@resettimer</code>	4, 658
<code>\loop</code>	1063, 1079, 1090, 1098	<code>\pdf@setdraftmode</code>	5, 419, 434
<code>\lowercase</code>	1276, 1302	<code>\pdf@shellescape</code>	5, 184, 186, 191, 671, 1200, 1211, 1214, 1220, 1227
<code>\ltx@backslashchar</code>	384, 388, 391, 650, 652, 654, 655, 1314	<code>\pdf@strcmp</code>	3, 367, 541
<code>\ltx@cclv</code>	421, 423	<code>\pdf@system</code>	5, 211, 676
<code>\ltx@firstoftwo</code>	338, 368, 403	<code>\pdf@unescapehex</code>	3, 172, 558, 1378, 1382, 1385
<code>\ltx@gobble</code>	1264, 1328	<code>\pdf@unescapehexnative</code>	6, 172, 567
<code>\ltx@ifundefined</code>	180, 383	<code>\pdfdraftmode</code>	376, 377, 400
<code>\ltx@one</code>	387, 402, 409, 410	<code>\pdffiledump</code>	200
<code>\ltx@onelevel@sanitize</code>	477, 1301, 1327, 1355, 1375	<code>\pdfmdfivesum</code>	207, 209
<code>\ltx@percentchar</code>	1331	<code>\pdfprimitive</code>	285
<code>\ltx@ReturnAfterElseFi</code>	515, 526	<code>\pdfshellescape</code>	192
<code>\ltx@ReturnAfterFi</code>	519, 531	<code>\pdftexcmds@Cisprimitive</code>	334, 336
<code>\ltx@secondoftwo</code>	340, 370, 397, 405	<code>\pdftexcmds@Csetdraftmode</code>	423, 427
<code>\ltx@zero</code>	382, 396, 412, 527	<code>\pdftexcmds@AtEnd</code>	95, 96, 126, 127, 441, 709
<code>\luaescapestring</code>	543, 544, 550, 555, 563, 572, 578, 583, 588, 593, 598, 603, 604, 605, 610, 615, 620, 678, 704	<code>\pdftexcmds@DecodeA</code>	511, 513
<code>\luatexversion</code>	227, 443, 460, 501, 502, 537, 671, 1197, 1199	<code>\pdftexcmds@DecodeB</code>	516, 524
M		<code>\pdftexcmds@directlua</code>	227, 235, 457, 462, 469, 479, 626, 638, 659, 665, 693, 702
<code>\MaybeEnd</code>	1196, 1207, 1210	<code>\pdftexcmds@draftmode</code>	400, 402, 409, 416
<code>\meaning</code>	252, 254, 312, 315, 331, 367	<code>\pdftexcmds@equal</code>	337, 343, 361
<code>\MessageBreak</code>	284, 434, 488, 489, 651, 653	<code>\pdftexcmds@equalcont</code>	352, 358, 359, 364
<code>\msg</code>	1148, 1163, 1171, 1195, 1201, 1202, 1204, 1212, 1214, 1219, 1221, 1226, 1228, 1244, 1366, 1367	<code>\pdftexcmds@Cisprimitive</code>	331, 333
N		<code>\pdftexcmds@nopdfTeX</code>	153, 155, 160, 179, 197, 205, 217
<code>\newlinechar</code>	233, 234, 1365	<code>\pdftexcmds@Patch</code>	500, 506, 563, 572, 704
<code>\next</code>	1069, 1071, 1073	<code>\pdftexcmds@PatchDecode</code>	510, 559, 568, 700
<code>\number</code>	128, 603, 604, 1104, 1214, 1226	<code>\pdftexcmds@setdraftmode</code>	398, 415, 429, 431
<code>\numexpr</code>	636, 664, 674	<code>\pdftexcmds@strip@prefix</code>	246, 252
P		<code>\pdftexcmds@temp</code>	157, 168, 169, 171, 173, 174, 175, 176, 214, 223, 224, 247, 262, 263, 264, 265, 266, 267, 268, 269, 307, 325, 326, 382, 387, 395, 623, 634, 646, 648
<code>\PackageInfo</code>	26	<code>\pdftexcmds@toks</code>	494, 560, 569, 701
<code>\par</code>	1184, 1185, 1186	<code>\pdftexrevision</code>	279
<code>\pdf@draftmode</code>	5, 396, 408	<code>\pdftexversion</code>	182, 276
<code>\pdf@elapsedtime</code>	4, 663	<code>\protected</code>	623, 658, 663, 673
<code>\pdf@escapehex</code>	4, 170, 548, 1384	<code>\ProvidesPackage</code>	19, 67
<code>\pdf@escapehexnative</code>	170, 553	R	
<code>\pdf@escapename</code>	581, 1386	<code>\RangeCatcodeCheck</code>	1096, 1124, 1125, 1126, 1127, 1128, 1129, 1130, 1131, 1132, 1133, 1134, 1135
<code>\pdf@escapenamenative</code>	586	<code>\RangeCatcodeInvalid</code>	1088, 1116, 1117, 1118, 1119
<code>\pdf@escapestring</code>	576, 1387	<code>\repeat</code>	1063, 1075, 1086, 1094, 1109
<code>\pdf@filedump</code>	4, 199, 601	<code>\RequirePackage</code>	145, 146, 147, 148, 454
<code>\pdf@filemdfivesum</code>	4, 209, 618	<code>\RestoreCatcodes</code>	1077, 1080, 1081, 1136
<code>\pdf@filemoddate</code>	4, 596	<code>\romannumeral</code>	559, 568, 700
<code>\pdf@filesize</code>	4, 591	S	
<code>\pdf@ifdraftmode</code>	5, 397, 401	<code>\space</code>	283, 285, 297, 302, 1102, 1103, 1111, 1253, 1274
<code>\pdf@ifprimitive</code>	6, 266, 302		
<code>\pdf@Cisprimitive</code>	6, 327, 330, 366, 380, 547, 1158, 1161		
<code>\pdf@lastsystemexit</code>	686		
<code>\pdf@lastsystemstatus</code>	681		
<code>\pdf@mdfivesum</code>	4, 207, 208, 608		
<code>\pdf@mdfivesumnative</code>	208, 613		
<code>\pdf@pipe</code>	6, 692		
<code>\pdf@primitive</code>	6, 262, 281, 283, 297		

<code>\str</code>	1157, 1163, 1165, 1169, 1171	<code>\TMP@RequirePackage</code>	
			. 134, 140, 141, 142, 143, 446, 452
		<code>\toksdef</code>	496
T			
<code>\Test</code>	1115, 1138, 1356, 1376, 1384, 1385, 1386, 1387	U	
<code>\test</code>	1154, 1176, 1177, 1178, 1179, 1180, 1181, 1182, 1183, 1184, 1185, 1186, 1372, 1378, 1382	<code>\UnDeFiNeD</code>	1197
<code>\TestExpect</code>	1374, 1375, 1376	W	
<code>\TestFrom</code>	1373, 1376	<code>\write</code>	23, 52, 212, 1151, 1195, 1247
<code>\TestResult</code>	1359, 1363, 1367	X	
<code>\the</code>	77, 78, 79, 80, 81, 82, 83, 84, 97, 423, 468, 560, 569, 701, 1082, 1102, 1103	<code>\x</code>	14, 15, 18, 22, 26, 28, 51, 56, 66, 75, 87, 251, 252, 256, 312, 317, 422, 425, 476, 477, 485, 489
<code>\TMP@EnsureCode</code>	94, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125	Y	
		<code>\y</code>	254, 256, 313, 315, 317, 478, 485, 490
		Z	
		<code>\z</code>	314, 315