The **pdftexcmds** package

Heiko Oberdiek

<heiko.oberdiek at googlemail.com>

2010/04/01 v0.9

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 2
   1.2 Macros 3
     1.2.1 Additional macro: \pdf@isprimitive 4
     1.2.2 Experimental 4

2 Implementation   **5**
   2.1 Reload check and package identification 5
   2.2 Catcodes 6
   2.3 Load package infwarerr 6
   2.4 Without LuaTEX 7
   2.5 \pdf@primitive, \pdf@ifprimitive 8
     2.5.1 Using LuaTEX's tex.enableprimitives 8
     2.5.2 Trying various names to find the primitives 8
     2.5.3 Result 9
   2.6 X ETEX 9
   2.7 \pdf@isprimitive 10
   2.8 Load Lua module 11
   2.9 Lua functions 11
   2.10 Lua module 14

3 Test   **18**
   3.1 Catcode checks for loading 18
   3.2 Test for \pdf@isprimitive 20

4 Installation   **20**
   4.1 Download 20
   4.2 Bundle installation 21
   4.3 Package installation 21
   4.4 Refresh file name databases 21
   4.5 Some details for the interested 21

5 History   **22**
   [2007/11/11 v0.1] 22
   [2007/11/12 v0.2] 22
   [2007/12/12 v0.3] 22
   [2009/04/10 v0.4] 22
   [2009/09/22 v0.5] 22

1 Documentation

Some primitives of pdfTEX are not defined by LuaTEX. This package implements macro based solutions using Lua code for the following missing pdfTEX primitives;

- `\pdfstrcmp`
- `\pdfunescapehex`
- `\pdfescapehex`
- `\pdfescapename`
- `\pdfescapestring`
- `\pdffilesize`
- `\pdffilemoddate`
- `\pdffiledump`
- `\pdfmdfivesum`
- `\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 catched. 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

`\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 pdfTeX. However pdfTeX 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.

`\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 TeX number.

`\pdf@mdfivesum {⟨string⟩}`

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

`\pdf@filemdfivesum {⟨filename⟩}`

Same as `\pdfmdfivesum file{⟨filename⟩}`.

`\pdf@shellescape`

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

`\pdf@system {⟨cmdline⟩}`

It is a wrapper for `\immediate\write18` in pdfTEX or `os.execute` in LuaTEX.

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 XƎTEX 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. XƎTEX calls it `\ifprimitive`. It is a switch that checks if the command `\cmd` has it's primitive meaning.

1.2.1 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 LATEX:

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

1.2.2 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
 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@pdftexcmds.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{pdftexcmds}{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}%
```

5

```
57      \fi
58    }%
59  \fi
60 \expandafter\x\csname ver@pdftexcmds.sty\endcsname
61 \ProvidesPackage{pdftexcmds}%
62   [2010/04/01 v0.9 Utility functions of pdfTeX for LuaTeX (HO)]
```

2.2   Catcodes

```
63 \begingroup
64   \catcode123 1 % {
65   \catcode125 2 % }
66   \def\x{\endgroup
67     \expandafter\edef\csname pdftexcmds@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\pdftexcmds@AtEnd{%
81     \pdftexcmds@AtEnd
82     \catcode#1 \the\catcode#1\relax
83   }%
84   \catcode#1 #2\relax
85 }
86 \TMP@EnsureCode{10}{12}% ^^J
87 \TMP@EnsureCode{33}{12}% !
88 \TMP@EnsureCode{34}{12}% "
89 \TMP@EnsureCode{39}{12}% '
90 \TMP@EnsureCode{40}{12}% (
91 \TMP@EnsureCode{41}{12}% )
92 \TMP@EnsureCode{42}{12}% *
93 \TMP@EnsureCode{43}{12}% +
94 \TMP@EnsureCode{44}{12}% ,
95 \TMP@EnsureCode{45}{12}% -
96 \TMP@EnsureCode{46}{12}% .
97 \TMP@EnsureCode{47}{12}% /
98 \TMP@EnsureCode{58}{12}% :
99 \TMP@EnsureCode{60}{12}% <
100 \TMP@EnsureCode{61}{12}% =
101 \TMP@EnsureCode{62}{12}% >
102 \TMP@EnsureCode{94}{7}% ^ (superscript)
103 \TMP@EnsureCode{95}{12}% _ (other)
104 \TMP@EnsureCode{96}{12}% `
105 \TMP@EnsureCode{126}{12}% ~ (other)
106 \edef\pdftexcmds@AtEnd{%
107   \pdftexcmds@AtEnd
108   \escapechar=\number\escapechar\relax
109 }
110 \escapechar=92 %
```

2.3   Load package **infwarerr**

```
111 \begingroup\expandafter\expandafter\expandafter\endgroup
112 \expandafter\ifx\csname RequirePackage\endcsname\relax
113   \input infwarerr.sty\relax
```

```
114  \input ifluatex.sty\relax
115  \input ltxcmds.sty\relax
116 \else
117  \RequirePackage{infwarerr}[2007/09/09]%
118  \RequirePackage{ifluatex}[2010/03/01]%
119  \RequirePackage{ltxcmds}[2009/12/12]%
120 \fi
```

2.4   Without LuaTeX

```
121 \ifluatex
122 \else
123  \@PackageInfoNoLine{pdftexcmds}{LuaTeX not detected}%
124  \def\pdftexcmds@nopdftex{%
125    \@PackageInfoNoLine{pdftexcmds}{pdfTeX >= 1.30 not detected}%
126    \let\pdftexcmds@nopdftex\relax
127  }%
128  \def\pdftexcmds@temp#1{%
129    \begingroup\expandafter\expandafter\expandafter\endgroup
130    \expandafter\ifx\csname pdf#1\endcsname\relax
131      \pdftexcmds@nopdftex
132    \else
133      \expandafter\def\csname pdf@#1\expandafter\endcsname
134      \expandafter##\expandafter{%
135        \csname pdf#1\endcsname
136      }%
137    \fi
138  }%
139  \pdftexcmds@temp{strcmp}%
140  \pdftexcmds@temp{escapehex}%
141  \let\pdf@escapehexnative\pdf@escapehex
142  \pdftexcmds@temp{unescapehex}%
143  \let\pdf@unescapehexnative\pdf@unescapehex
144  \pdftexcmds@temp{escapestring}%
145  \pdftexcmds@temp{escapename}%
146  \pdftexcmds@temp{filesize}%
147  \pdftexcmds@temp{filemoddate}%
148  \begingroup\expandafter\expandafter\expandafter\endgroup
149  \expandafter\ifx\csname pdfshellescape\endcsname\relax
150    \pdftexcmds@nopdftex
151    \ltx@IfUndefined{pdftexversion}{%
152    }{%
153      \ifnum\pdftexversion>120 % 1.21a supports \ifeof18
154        \ifeof18 %
155          \chardef\pdf@shellescape=0 %
156        \else
157          \chardef\pdf@shellescape=1 %
158        \fi
159      \fi
160    }%
161  \else
162    \def\pdf@shellescape{%
163      \pdfshellescape
164    }%
165  \fi
166  \begingroup\expandafter\expandafter\expandafter\endgroup
167  \expandafter\ifx\csname pdffiledump\endcsname\relax
168    \pdftexcmds@nopdftex
169  \else
170    \def\pdf@filedump#1#2#3{%
171      \pdffiledump offset#1 length#2{#3}%
172    }%
173  \fi
```

```
174   \begingroup\expandafter\expandafter\expandafter\endgroup
175   \expandafter\ifx\csname pdfmdfivesum\endcsname\relax
176     \pdftexcmds@nopdftex
177   \else
178     \def\pdf@mdfivesum#{\pdfmdfivesum}%
179     \let\pdf@mdfivesumnative\pdf@mdfivesum
180     \def\pdf@filemdfivesum#{\pdfmdfivesum file}%
181   \fi
182   \def\pdf@system#{%
183     \immediate\write18%
184   }%
185 \fi
```

2.5 \pdf@primitive, \pdf@ifprimitive

Since version 1.40.0 pdfTEX has \pdfprimitive and \ifpdfprimitive. And \pdfprimitive was fixed in version 1.40.4.

X∃TEX provides them under the name \primitive and \ifprimitive. LuaTEX 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 LuaTEX's tex.enableprimitives

```
186 \ifluatex
```

\pdftexcmds@directlua

```
187   \ifnum\luatexversion<36 %
188     \def\pdftexcmds@directlua{\directlua0 }%
189   \else
190     \let\pdftexcmds@directlua\directlua
191   \fi

192   \begingroup
193     \newlinechar=10 %
194     \endlinechar=\newlinechar
195     \pdftexcmds@directlua{%
196       if tex.enableprimitives then
197         tex.enableprimitives('pdf@', {'primitive', 'ifprimitive'})
198         tex.enableprimitives('', {'luaescapestring'})
199       end
200     }%
201   \endgroup %

202 \fi
```

2.5.2 Trying various names to find the primitives

\pdftexcmds@strip@prefix

```
203 \def\pdftexcmds@strip@prefix#1>{}

204 \def\pdftexcmds@temp#1#2#3{%
205   \begingroup\expandafter\expandafter\expandafter\endgroup
206   \expandafter\ifx\csname pdf@#1\endcsname\relax
207     \begingroup
208       \def\x{#3}%
209       \edef\x{\expandafter\pdftexcmds@strip@prefix\meaning\x}%
210       \escapechar=-1 %
211       \edef\y{\expandafter\meaning\csname#2\endcsname}%
212     \expandafter\endgroup
213     \ifx\x\y
```

```
214        \expandafter\let\csname pdf@#1\expandafter\endcsname
215        \csname #2\endcsname
216      \fi
217    \fi
218  }
```

```
219 \pdftexcmds@temp{primitive}{pdfprimitive}{pdfprimitive}% pdfTeX, LuaTeX
220 \pdftexcmds@temp{primitive}{primitive}{primitive}% XeTeX
221 \pdftexcmds@temp{primitive}{luatexprimitive}{pdfprimitive}% LuaTeX
222 \pdftexcmds@temp{primitive}{luatexpdfprimitive}{pdfprimitive}% LuaTeX
```

```
223 \pdftexcmds@temp{ifprimitive}{ifpdfprimitive}{ifpdfprimitive}% pdfTeX, LuaTeX
224 \pdftexcmds@temp{ifprimitive}{ifprimitive}{ifprimitive}% XeTeX
225 \pdftexcmds@temp{ifprimitive}{luatexifprimitive}{ifpdfprimitive}% LuaTeX
226 \pdftexcmds@temp{ifprimitive}{luatexifpdfprimitive}{ifpdfprimitive}% LuaTeX
```

Disable broken \pdfprimitive.

```
227 \begingroup
228   \expandafter\ifx\csname pdf@primitive\endcsname\relax
229   \else
230     \expandafter\ifx\csname pdftexversion\endcsname\relax
231     \else
232       \ifnum\pdftexversion=140 %
233         \expandafter\ifx\csname pdftexrevision\endcsname\relax
234         \else
235           \ifnum\pdftexrevision<4 %
236             \endgroup
237             \let\pdf@primitive\@undefined
238             \@PackageInfoNoLine{pdftexcmds}{%
239               \string\pdf@primitive disabled, because\MessageBreak
240               \string\pdfprimitive\space is broken until pdfTeX 1.40.4%
241             }%
242             \begingroup
243           \fi
244         \fi
245       \fi
246     \fi
247   \fi
248 \endgroup
```

2.5.3   Result

```
249 \begingroup
250   \@PackageInfoNoLine{pdftexcmds}{%
251     \string\pdf@primitive\space is %
252     \expandafter\ifx\csname pdf@primitive\endcsname\relax not \fi
253     available%
254   }%
255   \@PackageInfoNoLine{pdftexcmds}{%
256     \string\pdf@ifprimitive\space is %
257     \expandafter\ifx\csname pdf@ifprimitive\endcsname\relax not \fi
258     available%
259   }%
260 \endgroup
```

2.6   XƎTEX

Look for primitives \shellescape, \strcmp.

```
261 \def\pdftexcmds@temp#1{%
262   \begingroup\expandafter\expandafter\expandafter\endgroup
```

```
263 \expandafter\ifx\csname pdf@#1\endcsname\relax
264     \begingroup
265       \escapechar=-1 %
266       \edef\x{\expandafter\meaning\csname#1\endcsname}%
267       \def\y{#1}%
268       \def\z##1->{}%
269       \edef\y{\expandafter\z\meaning\y}%
270     \expandafter\endgroup
271     \ifx\x\y
272       \expandafter\def\csname pdf@#1\expandafter\endcsname
273       \expandafter{%
274         \csname#1\endcsname
275       }%
276     \fi
277   \fi
278 }%
279 \pdftexcmds@temp{shellescape}%
280 \pdftexcmds@temp{strcmp}%
```

2.7 \pdf@isprimitive

```
281 \def\pdf@isprimitive{%
282   \begingroup\expandafter\expandafter\expandafter\endgroup
283   \expandafter\ifx\csname pdf@strcmp\endcsname\relax
284     \long\def\pdf@isprimitive##1{%
285       \expandafter\pdftexcmds@isprimitive\expandafter{\meaning##1}%
286     }%
287     \long\def\pdftexcmds@isprimitive##1##2{%
288       \expandafter\pdftexcmds@@isprimitive\expandafter{\string##2}{##1}%
289     }%
290     \def\pdftexcmds@@isprimitive##1##2{%
291       \ifnum0\pdftexcmds@equal##1\delimiter##2\delimiter=1 %
292         \expandafter\ltx@firstoftwo
293       \else
294         \expandafter\ltx@secondoftwo
295       \fi
296     }%
297     \def\pdftexcmds@equal##1##2\delimiter##3##4\delimiter{%
298       \ifx##1##3%
299         \ifx\relax##2##4\relax
300           1%
301         \else
302           \ifx\relax##2\relax
303           \else
304             \ifx\relax##4\relax
305             \else
306               \pdftexcmds@equalcont{##2}{##4}%
307             \fi
308           \fi
309         \fi
310       \fi
311     }%
312     \def\pdftexcmds@equalcont##1{%
313       \def\pdftexcmds@equalcont####1####2##1##1##1##1{%
314         ##1##1##1##1%
315         \pdftexcmds@equal####1\delimiter####2\delimiter
316       }%
317     }%
318     \expandafter\pdftexcmds@equalcont\csname fi\endcsname
319   \else
320     \long\def\pdf@isprimitive##1##2{%
321       \ifnum\pdf@strcmp{\meaning##1}{\string##2}=0 %
```

10

```
322        \expandafter\ltx@firstoftwo
323      \else
324        \expandafter\ltx@secondoftwo
325      \fi
326    }%
327  \fi
328 }
329 \ifluatex
330 \else
331   \pdf@isprimitive
332   \pdftexcmds@AtEnd
333   \expandafter\endinput
334 \fi
```

2.8 Load Lua module

```
335 \begingroup\expandafter\expandafter\expandafter\endgroup
336 \expandafter\ifx\csname RequirePackage\endcsname\relax
337   \input luatex-loader.sty\relax
338 \else
339   \RequirePackage{luatex-loader}[2009/04/10]%
340 \fi
341 \pdftexcmds@directlua{%
342   require("oberdiek.pdftexcmds")%
343 }
```

2.9 Lua functions

\pdftexcmds@toks

```
344 \begingroup\expandafter\expandafter\expandafter\endgroup
345 \expandafter\ifx\csname newtoks\endcsname\relax
346   \toksdef\pdftexcmds@toks=0 %
347 \else
348   \csname newtoks\endcsname\pdftexcmds@toks
349 \fi
350 \ifnum\luatexversion<36 %
351 \else
352   \catcode`\0=9 %
353 \fi
```

\pdf@strcmp

```
354 \long\def\pdf@strcmp#1#2{%
355   \directlua0{%
356     oberdiek.pdftexcmds.strcmp("\luaescapestring{#1}",%
357         "\luaescapestring{#2}")%
358   }%
359 }%
360 \pdf@isprimitive
```

\pdf@escapehex

```
361 \long\def\pdf@escapehex#1{%
362   \directlua0{%
363     oberdiek.pdftexcmds.escapehex("\luaescapestring{#1}", "byte")%
364   }%
365 }%
```

\pdf@escapehexnative

```
366 \long\def\pdf@escapehexnative#1{%
367   \directlua0{%
368     oberdiek.pdftexcmds.escapehex("\luaescapestring{#1}")%
369   }%
370 }%
```

11

`\pdf@unescapehex`

```
371 \def\pdf@unescapehex#1{%
372   \the\expandafter\pdftexcmds@toks
373   \directlua0{%
374     oberdiek.pdftexcmds.toks="pdftexcmds@toks"%
375     oberdiek.pdftexcmds.unescapehex("\luaescapestring{#1}", "byte")%
376   }%
377 }%
```

`\pdf@unescapehexnative`

```
378 \def\pdf@unescapehexnative#1{%
379   \the\expandafter\pdftexcmds@toks
380   \directlua0{%
381     oberdiek.pdftexcmds.toks="pdftexcmds@toks"%
382     oberdiek.pdftexcmds.unescapehex("\luaescapestring{#1}")%
383   }%
384 }%
```

`\pdf@escapestring`

```
385 \long\def\pdf@escapestring#1{%
386   \directlua0{%
387     oberdiek.pdftexcmds.escapestring("\luaescapestring{#1}", "byte")%
388   }%
389 }
```

`\pdf@escapename`

```
390 \long\def\pdf@escapename#1{%
391   \directlua0{%
392     oberdiek.pdftexcmds.escapename("\luaescapestring{#1}", "byte")%
393   }%
394 }
```

`\pdf@escapenamenative`

```
395 \long\def\pdf@escapenamenative#1{%
396   \directlua0{%
397     oberdiek.pdftexcmds.escapename("\luaescapestring{#1}")%
398   }%
399 }
```

`\pdf@filesize`

```
400 \def\pdf@filesize#1{%
401   \directlua0{%
402     oberdiek.pdftexcmds.filesize("\luaescapestring{#1}")%
403   }%
404 }
```

`\pdf@filemoddate`

```
405 \def\pdf@filemoddate#1{%
406   \directlua0{%
407     oberdiek.pdftexcmds.filemoddate("\luaescapestring{#1}")%
408   }%
409 }
```

`\pdf@filedump`

```
410 \def\pdf@filedump#1#2#3{%
411   \directlua0{%
412     oberdiek.pdftexcmds.filedump("\luaescapestring{\number#1}",%
413         "\luaescapestring{\number#2}",%
414         "\luaescapestring{#3}")%
415   }%
416 }%
```

```
417 \long\def\pdf@mdfivesum#1{%
418   \directlua0{%
419     oberdiek.pdftexcmds.mdfivesum("\luaescapestring{#1}", "byte")%
420   }%
421 }%
```

```
422 \long\def\pdf@mdfivesumnative#1{%
423   \directlua0{%
424     oberdiek.pdftexcmds.mdfivesum("\luaescapestring{#1}")%
425   }%
426 }%
```

```
427 \def\pdf@filemdfivesum#1{%
428   \directlua0{%
429     oberdiek.pdftexcmds.filemdfivesum("\luaescapestring{#1}")%
430   }%
431 }%
```

```
432 \def\pdf@shellescape{%
433   \directlua0{%
434     oberdiek.pdftexcmds.shellescape()%
435   }%
436 }
```

```
437 \def\pdf@system#1{%
438   \directlua0{%
439     oberdiek.pdftexcmds.system("\luaescapestring{#1}")%
440   }%
441 }
```

```
442 \def\pdf@lastsystemstatus{%
443   \directlua0{%
444     oberdiek.pdftexcmds.lastsystemstatus()%
445   }%
446 }
```

```
447 \def\pdf@lastsystemexit{%
448   \directlua0{%
449     oberdiek.pdftexcmds.lastsystemexit()%
450   }%
451 }
```

```
452 \catcode`\0=12 %
```

Check availability of `io.popen` first.

```
453 \ifnum0%
454   \pdftexcmds@directlua{%
455     if io.popen then %
456       tex.write("1")%
457     end%
458   }%
459   =1 %
460   \def\pdf@pipe#1{%
461     \the\expandafter\pdftexcmds@toks
```

13

```
462    \pdftexcmds@directlua{%
463      oberdiek.pdftexcmds.toks="pdftexcmds@toks"%
464      oberdiek.pdftexcmds.pipe("\luaescapestring{#1}")%
465    }%
466  }%
467 \fi

468 \pdftexcmds@AtEnd
469 ⟨/package⟩
```

2.10   Lua module

470 ⟨*lua⟩

```
471 module("oberdiek.pdftexcmds", package.seeall)
472 local systemexitstatus
473 function strcmp(A, B)
474   if A == B then
475     tex.write("0")
476   elseif A < B then
477     tex.write("-1")
478   else
479     tex.write("1")
480   end
481 end
482 local function utf8_to_byte(str)
483   local i = 0
484   local n = string.len(str)
485   local t = {}
486   while i < n do
487     i = i + 1
488     local a = string.byte(str, i)
489     if a < 128 then
490       table.insert(t, string.char(a))
491     else
492       if a >= 192 and i < n then
493         i = i + 1
494         local b = string.byte(str, i)
495         if b < 128 or b >= 192 then
496           i = i - 1
497         elseif a == 194 then
498           table.insert(t, string.char(b))
499         elseif a == 195 then
500           table.insert(t, string.char(b + 64))
501         end
502       end
503     end
504   end
505   return table.concat(t)
506 end
507 function escapehex(str, mode)
508   if mode == "byte" then
509     str = utf8_to_byte(str)
510   end
511   tex.write((string.gsub(str, ".",
512     function (ch)
513       return string.format("%02X", string.byte(ch))
514     end
515   )))
516 end
```

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

```
517 function unescapehex(str, mode)
```

14

```
518    local a = 0
519    local first = true
520    local result = {}
521    for i = 1, string.len(str), 1 do
522      local ch = string.byte(str, i)
523      if ch >= 48 and ch <= 57 then
524        ch = ch - 48
525      elseif ch >= 65 and ch <= 70 then
526        ch = ch - 55
527      elseif ch >= 97 and ch <= 102 then
528        ch = ch - 87
529      else
530        ch = nil
531      end
532      if ch then
533        if first then
534          a = ch * 16
535          first = false
536        else
537          table.insert(result, a + ch)
538          first = true
539        end
540      end
541    end
542    if not first then
543      table.insert(result, a)
544    end
545    if mode == "byte" then
546      local utf8 = {}
547      for i, a in ipairs(result) do
548        if a < 128 then
549          table.insert(utf8, a)
550        else
551          if a < 192 then
552            table.insert(utf8, 194)
553            a = a - 128
554          else
555            table.insert(utf8, 195)
556            a = a - 192
557          end
558          table.insert(utf8, a + 128)
559        end
560      end
561      result = utf8
562    end
563    tex.settoks(toks, string.char(unpack(result)))
564 end
```

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

```
565 function escapestring(str, mode)
566   if mode == "byte" then
567     str = utf8_to_byte(str)
568   end
569   tex.write((string.gsub(str, ".",
570     function (ch)
571       local b = string.byte(ch)
572       if b < 33 or b > 126 then
573         return string.format("\\%.3o", b)
574       end
575       if b == 40 or b == 41 or b == 92 then
576         return "\\" .. ch
577       end
```

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

```
578        return nil
579      end
580   )))
581 end
```

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

```
582 function escapename(str, mode)
583   if mode == "byte" then
584     str = utf8_to_byte(str)
585   end
586   tex.write((string.gsub(str, ".",
587     function (ch)
588       local b = string.byte(ch)
589       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.

```
590         return ""
591       end
592       if b <= 32 or b >= 127
593          or b == 35 or b == 37 or b == 40 or b == 41
594          or b == 47 or b == 60 or b == 62 or b == 91
595          or b == 93 or b == 123 or b == 125 then
596         return string.format("#%.2X", b)
597       else
```

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

```
598         return nil
599       end
600     end
601   )))
602 end
603 function filesize(filename)
604   local foundfile = kpse.find_file(filename, "tex", true)
605   if foundfile then
606     local size = lfs.attributes(foundfile, "size")
607     if size then
608       tex.write(size)
609     end
610   end
611 end
```

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

```
612 function filemoddate(filename)
613   local foundfile = kpse.find_file(filename, "tex", true)
614   if foundfile then
615     local date = lfs.attributes(foundfile, "modification")
616     if date then
617       local d = os.date("*t", date)
618       if d.sec >= 60 then
619         d.sec = 59
620       end
621       local u = os.date("!*t", date)
622       local off = 60 * (d.hour - u.hour) + d.min - u.min
623       if d.year ~= u.year then
624         if d.year > u.year then
625           off = off + 1440
626         else
627           off = off - 1440
628         end
629       elseif d.yday ~= u.yday then
630         if d.yday > u.yday then
631           off = off + 1440
632         else
633           off = off - 1440
```

```
634          end
635        end
636        local timezone
637        if off == 0 then
638          timezone = "Z"
639        else
640          local hours = math.floor(off / 60)
641          local mins = math.abs(off - hours * 60)
642          timezone = string.format("%+03d'%02d'", hours, mins)
643        end
644        tex.write(string.format("D:%04d%02d%02d%02d%02d%02d%s",
645            d.year, d.month, d.day, d.hour, d.min, d.sec, timezone))
646      end
647    end
648 end
649 function filedump(offset, length, filename)
650    length = tonumber(length)
651    if length and length > 0 then
652      local foundfile = kpse.find_file(filename, "tex", true)
653      if foundfile then
654        offset = tonumber(offset)
655        if not offset then
656          offset = 0
657        end
658        local filehandle = io.open(foundfile, "r")
659        if filehandle then
660          if offset > 0 then
661            filehandle:seek("set", offset)
662          end
663          local dump = filehandle:read(length)
664          escapehex(dump)
665        end
666      end
667    end
668 end
669 function mdfivesum(str, mode)
670    if mode == "byte" then
671      str = utf8_to_byte(str)
672    end
673    escapehex(md5.sum(str))
674 end
675 function filemdfivesum(filename)
676    local foundfile = kpse.find_file(filename, "tex", true)
677    if foundfile then
678      local filehandle = io.open(foundfile, "r")
679      if filehandle then
680        local contents = filehandle:read("*a")
681        escapehex(md5.sum(contents))
682      end
683    end
684 end
685 function shellescape()
686    if os.execute then
687      tex.write("1")
688    else
689      tex.write("0")
690    end
691 end
692 function system(cmdline)
693    systemexitstatus = nil
694    texio.write_nl("log", "system(" .. cmdline .. ") ")
695    if os.execute then
```

```
696    texio.write("log", "executed.")
697    systemexitstatus = os.execute(cmdline)
698  else
699    texio.write("log", "disabled.")
700  end
701 end
702 function lastsystemstatus()
703  local result = tonumber(systemexitstatus)
704  if result then
705    local x = math.floor(result / 256)
706    tex.write(result - 256 * math.floor(result / 256))
707  end
708 end
709 function lastsystemexit()
710  local result = tonumber(systemexitstatus)
711  if result then
712    tex.write(math.floor(result / 256))
713  end
714 end
715 function pipe(cmdline)
716  local result
717  systemexitstatus = nil
718  texio.write_nl("log", "pipe(" .. cmdline ..") ")
719  if io.popen then
720    texio.write("log", "executed.")
721    local handle = io.popen(cmdline, "r")
722    if handle then
723      result = handle:read("*a")
724      handle:close()
725    end
726  else
727    texio.write("log", "disabled.")
728  end
729  if result then
730    tex.settoks(toks, result)
731  else
732    tex.settoks(toks, "")
733  end
734 end
735 ⟨/lua⟩
```

3   Test

3.1   Catcode checks for loading

```
736 ⟨*test1⟩
```

```
737 \catcode`\{=1 %
738 \catcode`\}=2 %
739 \catcode`\#=6 %
740 \catcode`\@=11 %
741 \expandafter\ifx\csname count@\endcsname\relax
742  \countdef\count@=255 %
743 \fi
744 \expandafter\ifx\csname @gobble\endcsname\relax
745  \long\def\@gobble#1{}%
746 \fi
747 \expandafter\ifx\csname @firstofone\endcsname\relax
748  \long\def\@firstofone#1{#1}%
749 \fi
750 \expandafter\ifx\csname loop\endcsname\relax
751  \expandafter\@firstofone
752 \else
```

```
753    \expandafter\@gobble
754 \fi
755 {%
756    \def\loop#1\repeat{%
757       \def\body{#1}%
758       \iterate
759    }%
760    \def\iterate{%
761       \body
762          \let\next\iterate
763       \else
764          \let\next\relax
765       \fi
766       \next
767    }%
768    \let\repeat=\fi
769 }%
770 \def\RestoreCatcodes{}
771 \count@=0 %
772 \loop
773    \edef\RestoreCatcodes{%
774       \RestoreCatcodes
775       \catcode\the\count@=\the\catcode\count@\relax
776    }%
777 \ifnum\count@<255 %
778    \advance\count@ 1 %
779 \repeat
780
781 \def\RangeCatcodeInvalid#1#2{%
782    \count@=#1\relax
783    \loop
784       \catcode\count@=15 %
785    \ifnum\count@<#2\relax
786       \advance\count@ 1 %
787    \repeat
788 }
789 \expandafter\ifx\csname LoadCommand\endcsname\relax
790    \def\LoadCommand{\input pdftexcmds.sty\relax}%
791 \fi
792 \def\Test{%
793    \RangeCatcodeInvalid{0}{47}%
794    \RangeCatcodeInvalid{58}{64}%
795    \RangeCatcodeInvalid{91}{96}%
796    \RangeCatcodeInvalid{123}{255}%
797    \catcode`\@=12 %
798    \catcode`\\=0 %
799    \catcode`\{=1 %
800    \catcode`\}=2 %
801    \catcode`\#=6 %
802    \catcode`\[=12 %
803    \catcode`\]=12 %
804    \catcode`\%=14 %
805    \catcode`\ =10 %
806    \catcode13=5 %
807    \LoadCommand
808    \RestoreCatcodes
809 }
810 \Test
811 \csname @@end\endcsname
812 \end
813 ⟨/test1⟩
```

3.2 Test for \pdf@isprimitive

```
814 ⟨*test2⟩
815 \catcode`\{=1 %
816 \catcode`\}=2 %
817 \catcode`\#=6 %
818 \catcode`\@=11 %
819 \input pdftexcmds.sty\relax
820 \def\msg#1{%
821   \begingroup
822     \escapechar=92 %
823     \immediate\write16{#1}%
824   \endgroup
825 }
826 \long\def\test#1#2#3#4{%
827   \begingroup
828     #4%
829     \def\str{%
830       Test \string\pdf@isprimitive
831       {\string #1}{\string #2}{...}: %
832     }%
833     \pdf@isprimitive{#1}{#2}{%
834       \ifx#3Y%
835         \msg{\str true ==> OK.}%
836       \else
837         \errmessage{\str false ==> FAILED}%
838       \fi
839     }{%
840       \ifx#3Y%
841         \errmessage{\str true ==> FAILED}%
842       \else
843         \msg{\str false ==> OK.}%
844       \fi
845     }%
846   \endgroup
847 }
848 \test\relax\relax Y{}
849 \test\foobar\relax Y{\let\foobar\relax}
850 \test\foobar\relax N{}
851 \test\hbox\hbox Y{}
852 \test\foobar@hbox\hbox Y{\let\foobar@hbox\hbox}
853 \test\if\if Y{}
854 \test\if\ifx N{}
855 \test\ifx\if N{}
856 \test\par\par Y{}
857 \test\hbox\par N{}
858 \test\par\hbox N{}
859 \csname @@end\endcsname\end
860 ⟨/test2⟩
```

4 Installation

4.1 Download

Package. This package is available on CTAN[1]:

CTAN:macros/latex/contrib/oberdiek/pdftexcmds.dtx The source file.

CTAN:macros/latex/contrib/oberdiek/pdftexcmds.pdf Documentation.

---

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

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

TDS refers to the standard "A Directory Structure for TeX Files" (CTAN:tds/tds.pdf). Directories with `texmf` in their name are usually organized this way.

4.2   Bundle installation

Unpacking. Unpack the `oberdiek.tds.zip` in the TDS tree (also known as `texmf` tree) of your choice. Example (linux):

```
unzip oberdiek.tds.zip -d ~/texmf
```

Script installation. Check the directory `TDS:scripts/oberdiek/` for scripts that need further installation steps. Package `attachfile2` comes with the Perl script `pdfatfi.pl` that should be installed in such a way that it can be called as `pdfatfi`. Example (linux):

```
chmod +x scripts/oberdiek/pdfatfi.pl
cp scripts/oberdiek/pdfatfi.pl /usr/local/bin/
```

4.3   Package installation

Unpacking. The `.dtx` file is a self-extracting `docstrip` archive. The files are extracted by running the `.dtx` through plain TeX:

```
tex 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
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 (teTeX, mikTeX, . . .) relies on file name databases, you must refresh these. For example, teTeX users run `texhash` or `mktexlsr`.

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 pdftexcmds.pdf unpack_files output .
```

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

plain TEX: Run docstrip and extract the files.

LATEX: Generate the documentation.

If you insist on using LATEX for docstrip (really, docstrip does not need LATEX), then inform the autodetect routine about your intention:

```
latex \let\install=y\input{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 pdfLATEX:

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

5 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.
- XƎTEX's variants are detected for \pdf@shellescape, \pdf@strcmp, \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).

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.

23