

# The pdf texcmds package

Heiko Oberdiek  
<oberdiek@uni-freiburg.de>

2007/12/12 v0.3

## Abstract

LUA<sub>T</sub><sub>E</sub>X provides most of the commands of pdf<sub>T</sub><sub>E</sub>X 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

<b>1</b>	<b>Documentation</b>	<b>1</b>
1.1	General principles . . . . .	2
1.2	Macros . . . . .	2
1.2.1	Experimental . . . . .	3
<b>2</b>	<b>Implementation</b>	<b>4</b>
2.1	Reload check and package identification . . . . .	4
2.2	Catcodes . . . . .	5
2.3	Load package infwarerr . . . . .	6
2.4	Without LUA <sub>T</sub> <sub>E</sub> X . . . . .	6
2.5	Load module . . . . .	7
2.6	Lua functions . . . . .	7
2.7	Lua module . . . . .	9
<b>3</b>	<b>Test</b>	<b>14</b>
3.1	Catcode checks for loading . . . . .	14
<b>4</b>	<b>Installation</b>	<b>15</b>
4.1	Download . . . . .	15
4.2	Bundle installation . . . . .	16
4.3	Package installation . . . . .	16
4.4	Refresh file name databases . . . . .	16
4.5	Some details for the interested . . . . .	16
<b>5</b>	<b>History</b>	<b>17</b>
	[2007/11/11 v0.1] . . . . .	17
	[2007/11/12 v0.2] . . . . .	17
	[2007/12/12 v0.3] . . . . .	17
<b>6</b>	<b>Index</b>	<b>17</b>

## 1 Documentation

Some primitives of pdf<sub>T</sub><sub>E</sub>X are not defined by LUA<sub>T</sub><sub>E</sub>X. This package implements macro based solutions using Lua code for the following missing pdf<sub>T</sub><sub>E</sub>X 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 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}.
```

LUA<sub>T</sub>E<sub>X</sub> 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 pdf<sub>T</sub>E<sub>X</sub> provides `\pdf⟨cmd⟩`.

**Arguments:** The order of arguments in `\pdf@⟨cmd⟩` is the same as for the corresponding primitive of pdf<sub>T</sub>E<sub>X</sub>. The arguments are ordinary undelimited T<sub>E</sub>X arguments, no *⟨general text⟩* and without additional keywords.

**Expandibility:** The macro `\pdf@⟨cmd⟩` is expandable if the corresponding pdf<sub>T</sub>E<sub>X</sub> primitive has this property. Exact two expansion steps are necessary (first is the macro expansion).

**Without Lua<sub>T</sub>E<sub>X</sub>:** The macros `\pdf@⟨cmd⟩` are mapped to the commands of pdf<sub>T</sub>E<sub>X</sub> if they are available. Otherwise they are undefined.

## 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.

<code>\pdf@escapehex {⟨string⟩}</code> <code>\pdf@escapestring {⟨string⟩}</code> <code>\pdf@escapename {⟨string⟩}</code>
--

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.

<code>\pdf@filesize {⟨filename⟩}</code>
---

Same as `\pdffilesize{⟨filename⟩}`.

<code>\pdf@filemoddate {⟨filename⟩}</code>
--

Same as `\pdffilemoddate{⟨filename⟩}`.

<code>\pdf@filedump {⟨offset⟩} {⟨length⟩} {⟨filename⟩}</code>
---

Same as `\pdffiledump offset ⟨offset⟩ length ⟨length⟩ {⟨filename⟩}`. Both `⟨offset⟩` and `⟨length⟩` must not be empty, but must be a valid TeX number.

<code>\pdf@mdfivesum {⟨string⟩}</code>
--

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

<code>\pdf@filemdfivesum {⟨filename⟩}</code>
--

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

<code>\pdf@shellescape</code>
-------------------------------

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.

<code>\pdf@system {⟨cmdline⟩}</code>
--------------------------------------

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?

### 1.2.1 Experimental

<code>\pdf@unescapehexnative {⟨string⟩}</code> <code>\pdf@escapehexnative {⟨string⟩}</code> <code>\pdf@escapenamenative {⟨string⟩}</code> <code>\pdf@mdfivesumnative {⟨string⟩}</code>
---

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 L<sup>A</sup>T<sub>E</sub>X.

```

2 \begingroup
3   \catcode44 12 % ,
4   \catcode45 12 % -
5   \catcode46 12 % .
6   \catcode58 12 % :
7   \catcode64 11 % @
8   \expandafter\let\expandafter\x\csname ver@pdfdoccmds.sty\endcsname
9   \ifcase 0%
10    \ifx\x\relax % plain
11    \else
12      \ifx\x\empty % LaTeX
13      \else
14        1%
15      \fi
16    \fi
17  \else
18    \catcode35 6 % #
19    \catcode123 1 % {
20    \catcode125 2 % }
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{pdfdoccmds}{The package is already loaded}%
29  \endgroup
30  \expandafter\endinput
31  \fi
32 \endgroup

```

Package identification:

```

33 \begingroup
34   \catcode35 6 % #
35   \catcode40 12 % (
36   \catcode41 12 % )
37   \catcode44 12 % ,
38   \catcode45 12 % -
39   \catcode46 12 % .
40   \catcode47 12 % /
41   \catcode58 12 % :
42   \catcode64 11 % @
43   \catcode123 1 % {

```

```

44 \catcode125 2 % }
45 \expandafter\ifx\csname ProvidesPackage\endcsname\relax
46 \def\x#1#2#3[#4]{\endgroup
47 \immediate\write-1{Package: #3 #4}%
48 \xdef#1{#4}%
49 }%
50 \else
51 \def\x#1#2[#3]{\endgroup
52 #2[#3]}%
53 \ifx#1\relax
54 \xdef#1{#3}%
55 \fi
56 }%
57 \fi
58 \expandafter\x\csname ver@pdftexcmds.sty\endcsname
59 \ProvidesPackage{pdftexcmds}%
60 [2007/12/12 v0.3 LuaTeX support for pdfTeX utility functions (HO)]

```

## 2.2 Catcodes

```

61 \begingroup
62 \catcode123 1 % {
63 \catcode125 2 % }
64 \def\x{\endgroup
65 \expandafter\edef\csname pdftexcmds@AtEnd\endcsname{%
66 \catcode35 \the\catcode35\relax
67 \catcode64 \the\catcode64\relax
68 \catcode123 \the\catcode123\relax
69 \catcode125 \the\catcode125\relax
70 }%
71 }%
72 \x
73 \catcode35 6 % #
74 \catcode64 11 % @
75 \catcode123 1 % {
76 \catcode125 2 % }
77 \def\TMP@EnsureCode#1#2{%
78 \edef\pdftexcmds@AtEnd{%
79 \pdftexcmds@AtEnd
80 \catcode#1 \the\catcode#1\relax
81 }%
82 \catcode#1 #2\relax
83 }
84 \TMP@EnsureCode{10}{12}% ^^J
85 \TMP@EnsureCode{33}{12}% !
86 \TMP@EnsureCode{34}{12}% "
87 \TMP@EnsureCode{39}{12}% '
88 \TMP@EnsureCode{40}{12}% (
89 \TMP@EnsureCode{41}{12}% )
90 \TMP@EnsureCode{42}{12}% *
91 \TMP@EnsureCode{43}{12}% +
92 \TMP@EnsureCode{44}{12}% ,
93 \TMP@EnsureCode{45}{12}% -
94 \TMP@EnsureCode{46}{12}% .
95 \TMP@EnsureCode{47}{12}% /
96 \TMP@EnsureCode{58}{12}% :
97 \TMP@EnsureCode{60}{12}% <
98 \TMP@EnsureCode{61}{12}% =
99 \TMP@EnsureCode{62}{12}% >
100 \TMP@EnsureCode{94}{7}% ^ (superscript)
101 \TMP@EnsureCode{95}{12}% _ (other)
102 \TMP@EnsureCode{126}{12}% ~ (other)

```

## 2.3 Load package infwarerr

```
103 \begingroup\expandafter\expandafter\expandafter\endgroup
104 \expandafter\ifx\csname RequirePackage\endcsname\relax
105   \input infwarerr.sty\relax
106 \else
107   \RequirePackage{infwarerr}[2007/09/09]%
108 \fi
```

## 2.4 Without LuaTeX

```
109 \begingroup\expandafter\expandafter\expandafter\endgroup
110 \expandafter\ifx\csname directlua\endcsname\relax
111   \@PackageInfo{pdftexcmds}{LuaTeX not detected}%
112   \def\pdftexcmds@nopdftex{%
113     \@PackageInfoNoLine{pdftexcmds}{pdfTeX >= 1.30 not detected}%
114     \let\pdftexcmds@nopdftex\relax
115   }%
116   \def\pdftexcmds@temp#1{%
117     \begingroup\expandafter\expandafter\expandafter\endgroup
118     \expandafter\ifx\csname pdf#1\endcsname\relax
119       \pdftexcmds@nopdftex
120     \else
121       \expandafter\def\csname pdf@#1\endcsname\expandafter\endcsname
122       \expandafter##\expandafter{%
123         \csname pdf#1\endcsname
124       }%
125     \fi
126   }%
127   \pdftexcmds@temp{strcmp}%
128   \pdftexcmds@temp{escapehex}%
129   \let\pdf@escapehexnative\pdf@escapehex
130   \pdftexcmds@temp{unescapehex}%
131   \let\pdf@unescapehexnative\pdf@unescapehex
132   \pdftexcmds@temp{escapestring}%
133   \pdftexcmds@temp{escapename}%
134   \pdftexcmds@temp{filesize}%
135   \pdftexcmds@temp{filemoddate}%
136   \begingroup\expandafter\expandafter\expandafter\endgroup
137   \expandafter\ifx\csname pdfshellescape\endcsname\relax
138     \pdftexcmds@nopdftex
139   \else
140     \def\pdf@shellescape{%
141       \pdfshellescape
142     }%
143   \fi
144   \begingroup\expandafter\expandafter\expandafter\endgroup
145   \expandafter\ifx\csname pdffiledump\endcsname\relax
146     \pdftexcmds@nopdftex
147   \else
148     \def\pdf@filedump#1#2#3{%
149       \pdffiledump offset#1 length#2{#3}%
150     }%
151   \fi
152   \begingroup\expandafter\expandafter\expandafter\endgroup
153   \expandafter\ifx\csname pdfmdfivesum\endcsname\relax
154     \pdftexcmds@nopdftex
155   \else
156     \def\pdf@mdfivesum#1{\pdfmdfivesum}%
157     \let\pdf@mdfivesumnative\pdf@mdfivesum
158     \def\pdf@filemdfivesum#1{\pdfmdfivesum file}%
159   \fi
160   \def\pdf@system#1{%
161     \immediate\write18%
```

```

162 }%
163 \pdftexcmds@AtEnd
164 \expandafter\endinput
165 \fi

```

## 2.5 Load module

```

166 \begingroup\expandafter\expandafter\expandafter\endgroup
167 \expandafter\ifx\csname RequirePackage\endcsname\relax
168 \input luatex-loader.sty\relax
169 \else
170 \RequirePackage{luatex-loader}[2007/12/12]%
171 \fi
172 \directlua0{%
173 require("oberdiek.pdftexcmds")%
174 }

```

## 2.6 Lua functions

\pdftexcmds@toks

```

175 \begingroup\expandafter\expandafter\expandafter\endgroup
176 \expandafter\ifx\csname newtoks\endcsname\relax
177 \toksdef\pdftexcmds@toks=0 %
178 \else
179 \csname newtoks\endcsname\pdftexcmds@toks
180 \fi

```

\pdf@strcmp

```

181 \long\def\pdf@strcmp#1#2{%
182 \directlua0{%
183 oberdiek.pdftexcmds.strptime("\luaescapestring{#1}",%
184 "\luaescapestring{#2}")%
185 }%
186 }%

```

\pdf@escapehex

```

187 \long\def\pdf@escapehex#1{%
188 \directlua0{%
189 oberdiek.pdftexcmds.escapehex("\luaescapestring{#1}", "byte")%
190 }%
191 }%

```

\pdf@escapehexnative

```

192 \long\def\pdf@escapehexnative#1{%
193 \directlua0{%
194 oberdiek.pdftexcmds.escapehex("\luaescapestring{#1}")%
195 }%
196 }%

```

\pdf@unescapehex

```

197 \def\pdf@unescapehex#1{%
198 \the\expandafter\pdftexcmds@toks
199 \directlua0{%
200 oberdiek.pdftexcmds.toks="pdftexcmds@toks"%
201 oberdiek.pdftexcmds.unescapehex("\luaescapestring{#1}", "byte")%
202 }%
203 }%

```

\pdf@unescapehexnative

```

204 \def\pdf@unescapehexnative#1{%
205 \the\expandafter\pdftexcmds@toks
206 \directlua0{%

```

```

207     oberdiek.pdfcmds.toks="pdfcmds@toks"%
208     oberdiek.pdfcmds.unescapehex("\luaescapestring{#1}")%
209 }%
210 }%

\pdf@escapestring
211 \long\def\pdf@escapestring#1{%
212   \directlua0{%
213     oberdiek.pdfcmds.escapestring("\luaescapestring{#1}", "byte")%
214   }%
215 }

\pdf@escapename
216 \long\def\pdf@escapename#1{%
217   \directlua0{%
218     oberdiek.pdfcmds.escapename("\luaescapestring{#1}", "byte")%
219   }%
220 }

\pdf@escapenamenative
221 \long\def\pdf@escapenamenative#1{%
222   \directlua0{%
223     oberdiek.pdfcmds.escapename("\luaescapestring{#1}")%
224   }%
225 }

\pdf@filesize
226 \def\pdf@filesize#1{%
227   \directlua0{%
228     oberdiek.pdfcmds.filesize("\luaescapestring{#1}")%
229   }%
230 }

\pdf@filemoddate
231 \def\pdf@filemoddate#1{%
232   \directlua0{%
233     oberdiek.pdfcmds.filemoddate("\luaescapestring{#1}")%
234   }%
235 }

\pdf@filedump
236 \def\pdf@filedump#1#2#3{%
237   \directlua0{%
238     oberdiek.pdfcmds.filedump("\luaescapestring{\number#1}",%
239       "\luaescapestring{\number#2}",%
240       "\luaescapestring{\number#3}")%
241   }%
242 }%

\pdf@mdfivesum
243 \long\def\pdf@mdfivesum#1{%
244   \directlua0{%
245     oberdiek.pdfcmds.mdfivesum("\luaescapestring{#1}", "byte")%
246   }%
247 }%

\pdf@mdfivesumnative
248 \long\def\pdf@mdfivesumnative#1{%
249   \directlua0{%
250     oberdiek.pdfcmds.mdfivesum("\luaescapestring{#1}")%
251   }%
252 }%

```



```

\pdf@filemdfivesum
253 \def\pdf@filemdfivesum#1{%
254   \directlua0{%
255     oberdiek.pdfdoccmds.filemdfivesum("\luaescapestring{#1}")%
256   }%
257 }%

\pdf@shellescape
258 \def\pdf@shellescape{%
259   \directlua0{%
260     oberdiek.pdfdoccmds.shellescape()%
261   }%
262 }

\pdf@system
263 \def\pdf@system#1{%
264   \directlua0{%
265     oberdiek.pdfdoccmds.system("\luaescapestring{#1}")%
266   }%
267 }

\pdf@lastsystemstatus
268 \def\pdf@lastsystemstatus{%
269   \directlua0{%
270     oberdiek.pdfdoccmds.lastsystemstatus()%
271   }%
272 }

\pdf@lastsystemexit
273 \def\pdf@lastsystemexit{%
274   \directlua0{%
275     oberdiek.pdfdoccmds.lastsystemexit()%
276   }%
277 }

\pdf@pipe Check availability of io.popen first.
278 \ifnum0%
279   \directlua0{%
280     if io.popen then %
281       tex.write("1")%
282     end%
283   }%
284   =1 %
285   \def\pdf@pipe#1{%
286     \the\expandafter\pdfdoccmds@toks
287     \directlua0{%
288       oberdiek.pdfdoccmds.toks="pdfdoccmds@toks"%
289       oberdiek.pdfdoccmds.pipe("\luaescapestring{#1}")%
290     }%
291   }%
292 \fi

293 \pdfdoccmds@AtEnd
294 \end{package}

```

## 2.7 Lua module

```

295 (*lua)
296 module("oberdiek.pdfdoccmds", package.seeall)
297 local systemexitstatus
298 function strcmp(A, B)

```

```

299  if A == B then
300      tex.write("0")
301  elseif A < B then
302      tex.write("-1")
303  else
304      tex.write("1")
305  end
306 end
307 local function utf8_to_byte(str)
308     local i = 0
309     local n = string.len(str)
310     local t = {}
311     while i < n do
312         i = i + 1
313         local a = string.byte(str, i)
314         if a < 128 then
315             table.insert(t, string.char(a))
316         else
317             if a >= 192 and i < n then
318                 i = i + 1
319                 local b = string.byte(str, i)
320                 if b < 128 or b >= 192 then
321                     i = i - 1
322                 elseif a == 194 then
323                     table.insert(t, string.char(b))
324                 elseif a == 195 then
325                     table.insert(t, string.char(b + 64))
326                 end
327             end
328         end
329     end
330     return table.concat(t)
331 end
332 function escapehex(str, mode)
333     if mode == "byte" then
334         str = utf8_to_byte(str)
335     end
336     tex.write((string.gsub(str, ".",
337         function (ch)
338             return string.format("%02X", string.byte(ch))
339         end
340     )))
341 end

```

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

```

342 function unescapehex(str, mode)
343     local a = 0
344     local first = true
345     local result = {}
346     for i = 1, string.len(str), 1 do
347         local ch = string.byte(str, i)
348         if ch >= 48 and ch <= 57 then
349             ch = ch - 48
350         elseif ch >= 65 and ch <= 70 then
351             ch = ch - 55
352         elseif ch >= 97 and ch <= 102 then
353             ch = ch - 87
354         else
355             ch = nil
356         end
357         if ch then
358             if first then

```

```

359         a = ch * 16
360         first = false
361     else
362         table.insert(result, a + ch)
363         first = true
364     end
365 end
366 end
367 if not first then
368     table.insert(result, a)
369 end
370 if mode == "byte" then
371     local utf8 = {}
372     for i, a in ipairs(result) do
373         if a < 128 then
374             table.insert(utf8, a)
375         else
376             if a < 192 then
377                 table.insert(utf8, 194)
378                 a = a - 128
379             else
380                 table.insert(utf8, 195)
381                 a = a - 192
382             end
383             table.insert(utf8, a + 128)
384         end
385     end
386     result = utf8
387 end
388 tex.settoks(toks, string.char(unpack(result)))
389 end

```

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

```

390 function escapestring(str, mode)
391     if mode == "byte" then
392         str = utf8_to_byte(str)
393     end
394     tex.write((string.gsub(str, ".",
395         function (ch)
396             local b = string.byte(ch)
397             if b < 33 or b > 126 then
398                 return string.format("\\%.3o", b)
399             end
400             if b == 40 or b == 41 or b == 92 then
401                 return "\\" .. ch
402             end

```

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

```

403         return nil
404     end
405 )))
406 end

```

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

```

407 function escapename(str, mode)
408     if mode == "byte" then
409         str = utf8_to_byte(str)
410     end
411     tex.write((string.gsub(str, ".",
412         function (ch)
413             local b = string.byte(ch)
414             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.

```

415         return ""
416     end
417     if b <= 32 or b >= 127
418         or b == 35 or b == 37 or b == 40 or b == 41
419         or b == 47 or b == 60 or b == 62 or b == 91
420         or b == 93 or b == 123 or b == 125 then
421         return string.format("#%.2X", b)
422     else
423         return nil
424     end
425 end
426 )))
427 end
428 function filesize(filename)
429     local foundfile = kpse.find_file(filename, "tex", true)
430     if foundfile then
431         local size = lfs.attributes(foundfile, "size")
432         if size then
433             tex.write(size)
434         end
435     end
436 end

```

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

```

437 function filemoddate(filename)
438     local foundfile = kpse.find_file(filename, "tex", true)
439     if foundfile then
440         local date = lfs.attributes(foundfile, "modification")
441         if date then
442             local d = os.date("*t", date)
443             if d.sec >= 60 then
444                 d.sec = 59
445             end
446             local u = os.date("!*t", date)
447             local off = 60 * (d.hour - u.hour) + d.min - u.min
448             if d.year ~= u.year then
449                 if d.year > u.year then
450                     off = off + 1440
451                 else
452                     off = off - 1440
453                 end
454             elseif d.yday ~= u.yday then
455                 if d.yday > u.yday then
456                     off = off + 1440
457                 else
458                     off = off - 1440
459                 end
460             end
461             local timezone
462             if off == 0 then
463                 timezone = "Z"
464             else
465                 local hours = math.floor(off / 60)
466                 local mins = math.abs(off - hours * 60)
467                 timezone = string.format("%+03d'%02d'", hours, mins)
468             end
469             tex.write(string.format("D:%04d%02d%02d%02d%02d%s",
470                 d.year, d.month, d.day, d.hour, d.min, d.sec, timezone))
471         end
472     end
473 end
474 function filedump(offset, length, filename)

```

```

475 length = tonumber(length)
476 if length and length > 0 then
477     local foundfile = kpse.find_file(filename, "tex", true)
478     if foundfile then
479         offset = tonumber(offset)
480         if not offset then
481             offset = 0
482         end
483         local filehandle = io.open(foundfile, "r")
484         if filehandle then
485             if offset > 0 then
486                 filehandle:seek("set", offset)
487             end
488             local dump = filehandle:read(length)
489             escapehex(dump)
490         end
491     end
492 end
493 end
494 function md5sum(str, mode)
495     if mode == "byte" then
496         str = utf8_to_byte(str)
497     end
498     escapehex(md5.sum(str))
499 end
500 function filemd5sum(filename)
501     local foundfile = kpse.find_file(filename, "tex", true)
502     if foundfile then
503         local filehandle = io.open(foundfile, "r")
504         if filehandle then
505             local contents = filehandle:read("*a")
506             escapehex(md5.sum(contents))
507         end
508     end
509 end
510 function shellescape()
511     if os.execute then
512         tex.write("1")
513     else
514         tex.write("0")
515     end
516 end
517 function system(cmdline)
518     systemexitstatus = nil
519     texio.write_nl("log", "system(" .. cmdline .. ") ")
520     if os.execute then
521         texio.write("log", "executed.")
522         systemexitstatus = os.execute(cmdline)
523     else
524         texio.write("log", "disabled.")
525     end
526 end
527 function lastsystemstatus()
528     local result = tonumber(systemexitstatus)
529     if result then
530         local x = math.floor(result / 256)
531         tex.write(result - 256 * math.floor(result / 256))
532     end
533 end
534 function lastsystemexit()
535     local result = tonumber(systemexitstatus)
536     if result then

```

```

537     tex.write(math.floor(result / 256))
538 end
539 end
540 function pipe(cmdline)
541     local result
542     systemexitstatus = nil
543     texio.write_nl("log", "pipe(" .. cmdline .. ") ")
544     if io.popen then
545         texio.write("log", "executed.")
546         local handle = io.popen(cmdline, "r")
547         if handle then
548             result = handle:read("*a")
549             handle:close()
550         end
551     else
552         texio.write("log", "disabled.")
553     end
554     if result then
555         tex.settoks(toks, result)
556     else
557         tex.settoks(toks, "")
558     end
559 end
560  $\langle$ /lua $\rangle$ 

```

### 3 Test

#### 3.1 Catcode checks for loading

```

561  $\langle$ *test1 $\rangle$ 
562 \catcode'\{=1 %
563 \catcode'\}=2 %
564 \catcode'\#=6 %
565 \catcode'\@=11 %
566 \expandafter\ifx\csname count@\endcsname\relax
567     \countdef\count@=255 %
568 \fi
569 \expandafter\ifx\csname @gobble\endcsname\relax
570     \long\def\@gobble#1{}%
571 \fi
572 \expandafter\ifx\csname @firstofone\endcsname\relax
573     \long\def\@firstofone#1{#1}%
574 \fi
575 \expandafter\ifx\csname loop\endcsname\relax
576     \expandafter\@firstofone
577 \else
578     \expandafter\@gobble
579 \fi
580 {%
581     \def\loop#1\repeat{%
582         \def\body{#1}%
583         \iterate
584     }%
585     \def\iterate{%
586         \body
587         \let\next\iterate
588     \else
589         \let\next\relax
590     \fi
591     \next
592 }%
593 \let\repeat=\fi

```

```

594 }%
595 \def\RestoreCatcodes{}
596 \count@=0 %
597 \loop
598   \edef\RestoreCatcodes{%
599     \RestoreCatcodes
600     \catcode\the\count@=\the\catcode\count@\relax
601   }%
602 \ifnum\count@<255 %
603   \advance\count@ 1 %
604 \repeat
605
606 \def\RangeCatcodeInvalid#1#2{%
607   \count@=#1\relax
608   \loop
609     \catcode\count@=15 %
610     \ifnum\count@<#2\relax
611       \advance\count@ 1 %
612     \repeat
613 }
614 \expandafter\ifx\csname LoadCommand\endcsname\relax
615   \def\LoadCommand{\input pdftexcmds.sty\relax}%
616 \fi
617 \def\Test{%
618   \RangeCatcodeInvalid{0}{47}%
619   \RangeCatcodeInvalid{58}{64}%
620   \RangeCatcodeInvalid{91}{96}%
621   \RangeCatcodeInvalid{123}{255}%
622   \catcode'\@=12 %
623   \catcode'\=0 %
624   \catcode'\{=1 %
625   \catcode'\}=2 %
626   \catcode'\#=6 %
627   \catcode'\[=12 %
628   \catcode'\]=12 %
629   \catcode'\%=14 %
630   \catcode'\ =10 %
631   \catcode13=5 %
632   \LoadCommand
633   \RestoreCatcodes
634 }
635 \Test
636 \csname @@end\endcsname
637 \end
638 </test1>

```

## 4 Installation

### 4.1 Download

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

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

[CTAN:macros/latex/contrib/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](#)

---

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

*TDS* refers to the standard “A Directory Structure for  $\text{\TeX}$  Files” ([CTAN:tds/tds.pdf](#)). Directories with `texmf` in their name are usually organized this way.

## 4.2 Bundle installation

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

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

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

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

## 4.3 Package installation

**Unpacking.** The `.dtx` file is a self-extracting `docstrip` archive. The files are extracted by running the `.dtx` through plain- $\text{\TeX}$ :

```
tex pdftexcmds.dtx
```

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

<code>pdftexcmds.sty</code>	$\rightarrow$ <code>tex/generic/oberdiek/pdftexcmds.sty</code>
<code>oberdiek.pdftexcmds.lua</code>	$\rightarrow$ <code>scripts/oberdiek/oberdiek.pdftexcmds.lua</code>
<code>pdftexcmds.lua</code>	$\rightarrow$ <code>scripts/oberdiek/pdftexcmds.lua</code>
<code>pdftexcmds.pdf</code>	$\rightarrow$ <code>doc/latex/oberdiek/pdftexcmds.pdf</code>
<code>pdftexcmds.dtx</code>	$\rightarrow$ <code>source/latex/oberdiek/pdftexcmds.dtx</code>

If you have a `docstrip.cfg` that configures and enables `docstrip`’s TDS installing feature, then some files can already be in the right place, see the documentation of `docstrip`.

## 4.4 Refresh file name databases

If your  $\text{\TeX}$  distribution (te $\text{\TeX}$ , mi $\text{\TeX}$ , ...) relies on file name databases, you must refresh these. For example, te $\text{\TeX}$  users run `texhash` or `mktextlsr`.

## 4.5 Some details for the interested

**Attached source.** The PDF documentation on CTAN also includes the `.dtx` source file. It can be extracted by AcrobatReader 6 or higher. Another option is `pdftk`, e.g. unpack the file into the current directory:

```
pdftk pdftexcmds.pdf unpack_files output .
```

**Unpacking with  $\text{\LaTeX}$ .** The `.dtx` chooses its action depending on the format:

**plain- $\text{\TeX}$ :** Run `docstrip` and extract the files.

**$\text{\LaTeX}$ :** Generate the documentation.

If you insist on using  $\text{\LaTeX}$  for `docstrip` (really, `docstrip` does not need  $\text{\LaTeX}$ ), then inform the autodetect routine about your intention:

```
latex \let\install=y\input{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 pdfL<sup>A</sup>T<sub>E</sub>X:

```
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.

## 6 Index

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

Symbols		43, 44, 62, 63, 66, 67, 68, 69, 73,
\#	564, 626	74, 75, 76, 80, 82, 562, 563, 564,
\%	629	565, 600, 609, 622, 623, 624,
\@	565, 622	625, 626, 627, 628, 629, 630, 631
\@PackageInfo	111	\count@ 567, 596,
\@PackageInfoNoLine	113	600, 602, 603, 607, 609, 610, 611
\@firstofone	573, 576	\countdef 567
\@gobble	570, 578	\csname 8, 21,
\[	627	45, 58, 65, 104, 110, 118, 121,
\]	398, 401, 623	123, 137, 145, 153, 167, 176,
\{	562, 624	179, 566, 569, 572, 575, 614, 636
\}	563, 625	
\]	628	
		<b>D</b>
\_	630	\directlua 172, 182,
		188, 193, 199, 206, 212, 217,
		222, 227, 232, 237, 244, 249,
		254, 259, 264, 269, 274, 279, 287
<b>A</b>		
\advance	603, 611	<b>E</b>
<b>B</b>		\empty 12
\body	582, 586	\end 637
<b>C</b>		\endcsname 8, 21,
\catcode	3, 4, 5, 6, 7, 18, 19, 20,	45, 58, 65, 104, 110, 118, 121,
	34, 35, 36, 37, 38, 39, 40, 41, 42,	123, 137, 145, 153, 167, 176,
		179, 566, 569, 572, 575, 614, 636
		\endinput 30, 164

<b>I</b>			
\ifcase	9	\pdf@pipe	4, 278
\ifnum	278, 602, 610	\pdf@shellescape	3, 140, 258
\ifx	10, 12, 21, 45, 53, 104, 110, 118, 137, 145, 153, 167, 176, 566, 569, 572, 575, 614	\pdf@strcmp	2, 181
\immediate	23, 47, 161	\pdf@system	3, 160, 263
\input	105, 168, 615	\pdf@unescapehex	2, 131, 197
\iterate	583, 585, 587	\pdf@unescapehexnative	3, 131, 204
<b>L</b>		\pdffiledump	149
\LoadCommand	615, 632	\pdfmdfivesum	156, 158
\loop	581, 597, 608	\pdfshellescape	141
\luaescapestring	183, 184, 189, 194, 201, 208, 213, 218, 223, 228, 233, 238, 239, 240, 245, 250, 255, 265, 289	\pdftexcmds@AtEnd	78, 79, 163, 293
<b>N</b>		\pdftexcmds@nopdfTeX	112, 114, 119, 138, 146, 154
\next	587, 589, 591	\pdftexcmds@temp	116, 127, 128, 130, 132, 133, 134, 135
\number	238, 239	\pdftexcmds@toks	175, 198, 205, 286
<b>P</b>		\ProvidesPackage	59
\PackageInfo	26	<b>R</b>	
\pdf@escapehex	3, 129, 187	\RangeCatcodeInvalid	606, 618, 619, 620, 621
\pdf@escapehexnative	129, 192	\repeat	581, 593, 604, 612
\pdf@escapename	216	\RequirePackage	107, 170
\pdf@escapenamenative	221	\RestoreCatcodes	595, 598, 599, 633
\pdf@escapestring	211	<b>T</b>	
\pdf@filedump	3, 148, 236	\Test	617, 635
\pdf@filemdfivesum	3, 158, 253	\the	66, 67, 68, 69, 80, 198, 205, 286, 600
\pdf@filemoddate	3, 231	\TMP@EnsureCode	77, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102
\pdf@filesize	3, 226	\toksdef	177
\pdf@lastsystemexit	273	<b>W</b>	
\pdf@lastsystemstatus	268	\write	23, 47, 161
\pdf@mdfivesum	3, 156, 157, 243	<b>X</b>	
\pdf@mdfivesumnative	157, 248	\x	8, 10, 12, 22, 26, 28, 46, 51, 58, 64, 72