

The pdftextcmds package

Heiko Oberdiek
<oberdiek@uni-freiburg.de>

2009/04/10 v0.4

Abstract

LUA_T_EX provides most of the commands of pdf_T_EX 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	1
1.1	General principles	2
1.2	Macros	2
1.2.1	Experimental	3
2	Implementation	4
2.1	Reload check and package identification	4
2.2	Catcodes	5
2.3	Load package infwarerr	6
2.4	Without LUA _T _E X	6
2.5	Load module	7
2.6	Lua functions	7
2.7	Lua module	10
3	Test	14
3.1	Catcode checks for loading	14
4	Installation	16
4.1	Download	16
4.2	Bundle installation	16
4.3	Package installation	16
4.4	Refresh file name databases	17
4.5	Some details for the interested	17
5	History	17
[2007/11/11 v0.1]		17
[2007/11/12 v0.2]		17
[2007/12/12 v0.3]		17
[2009/04/10 v0.4]		17
6	Index	18

1 Documentation

Some primitives of pdf_T_EX are not defined by LUA_T_EX. This package implements macro based solutions using Lua code for the following missing pdf_T_EX 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_TE_X 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_TE_X provides `\pdf⟨cmd⟩`.

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

Expandibility: The macro `\pdf@⟨cmd⟩` is expandable if the corresponding pdf_TE_X primitive has this property. Exact two expansion steps are necessary (first is the macro expansion).

Without Lua_TE_X: The macros `\pdf@⟨cmd⟩` are mapped to the commands of pdf_TE_X 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^AT_EX.

```
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@pdfdoccmds.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{pdfdoccmds}{The package is already loaded}%
26    \aftergroup\endinput
27  \fi
28 \fi
29 \endgroup
```

Package identification:

```
30 \begingroup
31   \catcode35 6 % #
32   \catcode40 12 % (
33   \catcode41 12 % )
34   \catcode44 12 % ,
35   \catcode45 12 % -
36   \catcode46 12 % .
37   \catcode47 12 % /
38   \catcode58 12 % :
39   \catcode64 11 % @
40   \catcode91 12 % [
41   \catcode93 12 % ]
42   \catcode123 1 % {
43   \catcode125 2 % }
```

```

44 \expandafter\ifx\csname ProvidesPackage\endcsname\relax
45   \def\x#1#2#3[#4]{\endgroup
46     \immediate\write-1{Package: #3 #4}%
47     \xdef#1{#4}%
48   }%
49 \else
50   \def\x#1#2[#3]{\endgroup
51     #2[#{#3}]%
52     \ifx#1\@undefined
53       \xdef#1{#3}%
54     \fi
55     \ifx#1\relax
56       \xdef#1{#3}%
57     \fi
58   }%
59 \fi
60 \expandafter\x\csname ver@pdftexcmds.sty\endcsname
61 \ProvidesPackage{pdftexcmds}%
62 [2009/04/10 v0.4 LuaTeX support for pdfTeX utility functions (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)

```

2.3 Load package infwarerr

```

106 \begingroup\expandafter\expandafter\expandafter\endgroup
107 \expandafter\ifx\csname RequirePackage\endcsname\relax
108   \input infwarerr.sty\relax
109   \input ifluatex.sty\relax
110 \else
111   \RequirePackage{infwarerr}[2007/09/09]%
112   \RequirePackage{ifluatex}[2009/04/10]%
113 \fi

```

2.4 Without LuaTeX

```

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

```

```

161 \def\pdf@mdfivesum#\pdfmdfivesum}%
162 \let\pdf@mdfivesumnative\pdf@mdfivesum
163 \def\pdf@filemdfivesum#\pdfmdfivesum file}%
164 \fi
165 \def\pdf@system#{%
166 \immediate\write18%
167 }%
168 \pdfTexcmds@AtEnd
169 \expandafter\endinput
170 \fi

```

\pdfTexcmds@directlua

```

171 \ifnum\luaTeXversion<36 %
172 \def\pdfTexcmds@directlua{\directlua0 }%
173 \else
174 \let\pdfTexcmds@directlua\directlua
175 \fi

```

2.5 Load module

```

176 \begingroup\expandafter\expandafter\expandafter\endgroup
177 \expandafter\ifx\csname RequirePackage\endcsname\relax
178 \input luatex-loader.sty\relax
179 \else
180 \RequirePackage{luatex-loader}[2009/04/10]%
181 \fi
182 \pdfTexcmds@directlua{%
183 require("oberdiek.pdfTexcmds")%
184 }

```

2.6 Lua functions

\pdfTexcmds@toks

```

185 \begingroup\expandafter\expandafter\expandafter\endgroup
186 \expandafter\ifx\csname newtoks\endcsname\relax
187 \toksdef\pdfTexcmds@toks=0 %
188 \else
189 \csname newtoks\endcsname\pdfTexcmds@toks
190 \fi

191 \ifnum\luaTeXversion<36 %
192 \else
193 \catcode'\0=9 %
194 \fi

```

\pdf@stricmp

```

195 \long\def\pdf@stricmp#1#2{%
196 \directlua0{%
197 oberdiek.pdfTexcmds.stricmp("\luaescapestring{#1}",%
198 "\luaescapestring{#2}")%
199 }%
200 }%

```

\pdf@escapehex

```

201 \long\def\pdf@escapehex#1{%
202 \directlua0{%
203 oberdiek.pdfTexcmds.escapehex("\luaescapestring{#1}", "byte")%
204 }%
205 }%

```

\pdf@escapehexnative

```

206 \long\def\pdf@escapehexnative#1{%

```

```

207 \directlua0{%
208     oberdiek.pdf texcmds.escapehex("\luaescapestring{#1}")%
209 }%
210 }%

\pdf@unescapehex

211 \def\pdf@unescapehex#1{%
212     \the\expandafter\pdf texcmds@toks
213     \directlua0{%
214         oberdiek.pdf texcmds.toks="pdf texcmds@toks"%
215         oberdiek.pdf texcmds.unescapehex("\luaescapestring{#1}", "byte")%
216     }%
217 }%

\pdf@unescapehexnative

218 \def\pdf@unescapehexnative#1{%
219     \the\expandafter\pdf texcmds@toks
220     \directlua0{%
221         oberdiek.pdf texcmds.toks="pdf texcmds@toks"%
222         oberdiek.pdf texcmds.unescapehex("\luaescapestring{#1}")%
223     }%
224 }%

\pdf@escapestring

225 \long\def\pdf@escapestring#1{%
226     \directlua0{%
227         oberdiek.pdf texcmds.escapestring("\luaescapestring{#1}", "byte")%
228     }%
229 }

\pdf@escapename

230 \long\def\pdf@escapename#1{%
231     \directlua0{%
232         oberdiek.pdf texcmds.escapename("\luaescapestring{#1}", "byte")%
233     }%
234 }

\pdf@escapenamenative

235 \long\def\pdf@escapenamenative#1{%
236     \directlua0{%
237         oberdiek.pdf texcmds.escapename("\luaescapestring{#1}")%
238     }%
239 }

\pdf@filesize

240 \def\pdf@filesize#1{%
241     \directlua0{%
242         oberdiek.pdf texcmds.filesize("\luaescapestring{#1}")%
243     }%
244 }

\pdf@filemoddate

245 \def\pdf@filemoddate#1{%
246     \directlua0{%
247         oberdiek.pdf texcmds.filemoddate("\luaescapestring{#1}")%
248     }%
249 }

\pdf@filedump

250 \def\pdf@filedump#1#2#3{%
251     \directlua0{%

```



```

252     oberdiek.pdfcmds.filedump("\luaescapestring{\number#1}",%
253     "\luaescapestring{\number#2}",%
254     "\luaescapestring{\number#3}")%
255 }%
256 }%

\pdf@mdfivesum

257 \long\def\pdf@mdfivesum#1{%
258   \directlua0{%
259     oberdiek.pdfcmds.mdfivesum("\luaescapestring{#1}", "byte")%
260   }%
261 }%

\pdf@mdfivesumnative

262 \long\def\pdf@mdfivesumnative#1{%
263   \directlua0{%
264     oberdiek.pdfcmds.mdfivesum("\luaescapestring{#1}")%
265   }%
266 }%

\pdf@filemdfivesum

267 \def\pdf@filemdfivesum#1{%
268   \directlua0{%
269     oberdiek.pdfcmds.filemdfivesum("\luaescapestring{#1}")%
270   }%
271 }%

\pdf@shellescape

272 \def\pdf@shellescape{%
273   \directlua0{%
274     oberdiek.pdfcmds.shellescape()%
275   }%
276 }

\pdf@system

277 \def\pdf@system#1{%
278   \directlua0{%
279     oberdiek.pdfcmds.system("\luaescapestring{#1}")%
280   }%
281 }

\pdf@lastsystemstatus

282 \def\pdf@lastsystemstatus{%
283   \directlua0{%
284     oberdiek.pdfcmds.lastsystemstatus()%
285   }%
286 }

\pdf@lastsystemexit

287 \def\pdf@lastsystemexit{%
288   \directlua0{%
289     oberdiek.pdfcmds.lastsystemexit()%
290   }%
291 }

292 \catcode'\0=12 %

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

293 \ifnum0%
294   \pdfcmds@directlua{%
295     if io.popen then %

```

```

296         tex.write("1")%
297     end%
298 }%
299 =1 %
300 \def\pdf@pipe#1{%
301     \the\expandafter\pdftexcmds@toks
302     \pdftexcmds@directlua{%
303         oberdiek.pdftexcmds.toks="pdftexcmds@toks"%
304         oberdiek.pdftexcmds.pipe("\luaescapestring{#1}")%
305     }%
306 }%
307 \fi

308 \pdftexcmds@AtEnd
309 \end{package}

```

2.7 Lua module

```

310 (*lua)

311 module("oberdiek.pdftexcmds", package.seeall)
312 local systemexitstatus
313 function strcmp(A, B)
314     if A == B then
315         tex.write("0")
316     elseif A < B then
317         tex.write("-1")
318     else
319         tex.write("1")
320     end
321 end
322 local function utf8_to_byte(str)
323     local i = 0
324     local n = string.len(str)
325     local t = {}
326     while i < n do
327         i = i + 1
328         local a = string.byte(str, i)
329         if a < 128 then
330             table.insert(t, string.char(a))
331         else
332             if a >= 192 and i < n then
333                 i = i + 1
334                 local b = string.byte(str, i)
335                 if b < 128 or b >= 192 then
336                     i = i - 1
337                 elseif a == 194 then
338                     table.insert(t, string.char(b))
339                 elseif a == 195 then
340                     table.insert(t, string.char(b + 64))
341                 end
342             end
343         end
344     end
345     return table.concat(t)
346 end
347 function escapehex(str, mode)
348     if mode == "byte" then
349         str = utf8_to_byte(str)
350     end
351     tex.write((string.gsub(str, ".",
352         function (ch)
353             return string.format("%02X", string.byte(ch))

```

```

354     end
355   )))
356 end

```

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

```

357 function unescapehex(str, mode)
358   local a = 0
359   local first = true
360   local result = {}
361   for i = 1, string.len(str), 1 do
362     local ch = string.byte(str, i)
363     if ch >= 48 and ch <= 57 then
364       ch = ch - 48
365     elseif ch >= 65 and ch <= 70 then
366       ch = ch - 55
367     elseif ch >= 97 and ch <= 102 then
368       ch = ch - 87
369     else
370       ch = nil
371     end
372     if ch then
373       if first then
374         a = ch * 16
375         first = false
376       else
377         table.insert(result, a + ch)
378         first = true
379       end
380     end
381   end
382   if not first then
383     table.insert(result, a)
384   end
385   if mode == "byte" then
386     local utf8 = {}
387     for i, a in ipairs(result) do
388       if a < 128 then
389         table.insert(utf8, a)
390       else
391         if a < 192 then
392           table.insert(utf8, 194)
393           a = a - 128
394         else
395           table.insert(utf8, 195)
396           a = a - 192
397         end
398         table.insert(utf8, a + 128)
399       end
400     end
401     result = utf8
402   end
403   tex.settoks(toks, string.char(unpack(result)))
404 end

```

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

```

405 function escapestring(str, mode)
406   if mode == "byte" then
407     str = utf8_to_byte(str)
408   end
409   tex.write((string.gsub(str, ".",
410     function (ch)
411       local b = string.byte(ch)
412       if b < 33 or b > 126 then

```

```

413         return string.format("\\%.3o", b)
414     end
415     if b == 40 or b == 41 or b == 92 then
416         return "\\" .. ch
417     end

```

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

```

418         return nil
419     end
420 )))
421 end

```

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

```

422 function escapename(str, mode)
423     if mode == "byte" then
424         str = utf8_to_byte(str)
425     end
426     tex.write((string.gsub(str, ".",
427         function (ch)
428             local b = string.byte(ch)
429             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.

```

430         return ""
431     end
432     if b <= 32 or b >= 127
433         or b == 35 or b == 37 or b == 40 or b == 41
434         or b == 47 or b == 60 or b == 62 or b == 91
435         or b == 93 or b == 123 or b == 125 then
436         return string.format("#%.2X", b)
437     else

```

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

```

438         return nil
439     end
440 end
441 )))
442 end
443 function filesize(filename)
444     local foundfile = kpse.find_file(filename, "tex", true)
445     if foundfile then
446         local size = lfs.attributes(foundfile, "size")
447         if size then
448             tex.write(size)
449         end
450     end
451 end

```

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

```

452 function filemoddate(filename)
453     local foundfile = kpse.find_file(filename, "tex", true)
454     if foundfile then
455         local date = lfs.attributes(foundfile, "modification")
456         if date then
457             local d = os.date("*t", date)
458             if d.sec >= 60 then
459                 d.sec = 59
460             end
461             local u = os.date("!*t", date)
462             local off = 60 * (d.hour - u.hour) + d.min - u.min
463             if d.year ~= u.year then
464                 if d.year > u.year then
465                     off = off + 1440
466                 else
467                     off = off - 1440

```

```

468         end
469     elseif d.yday ~= u.yday then
470         if d.yday > u.yday then
471             off = off + 1440
472         else
473             off = off - 1440
474         end
475     end
476     local timezone
477     if off == 0 then
478         timezone = "Z"
479     else
480         local hours = math.floor(off / 60)
481         local mins = math.abs(off - hours * 60)
482         timezone = string.format("%+03d'%02d'", hours, mins)
483     end
484     tex.write(string.format("D:%04d%02d%02d%02d%02d%02d%s",
485         d.year, d.month, d.day, d.hour, d.min, d.sec, timezone))
486 end
487 end
488 end
489 function filedump(offset, length, filename)
490     length = tonumber(length)
491     if length and length > 0 then
492         local foundfile = kpse.find_file(filename, "tex", true)
493         if foundfile then
494             offset = tonumber(offset)
495             if not offset then
496                 offset = 0
497             end
498             local filehandle = io.open(foundfile, "r")
499             if filehandle then
500                 if offset > 0 then
501                     filehandle:seek("set", offset)
502                 end
503                 local dump = filehandle:read(length)
504                 escapehex(dump)
505             end
506         end
507     end
508 end
509 function md5sum(str, mode)
510     if mode == "byte" then
511         str = utf8_to_byte(str)
512     end
513     escapehex(md5.sum(str))
514 end
515 function filemd5sum(filename)
516     local foundfile = kpse.find_file(filename, "tex", true)
517     if foundfile then
518         local filehandle = io.open(foundfile, "r")
519         if filehandle then
520             local contents = filehandle:read("*a")
521             escapehex(md5.sum(contents))
522         end
523     end
524 end
525 function shellescape()
526     if os.execute then
527         tex.write("1")
528     else
529         tex.write("0")

```

```

530 end
531 end
532 function system(cmdline)
533   systemexitstatus = nil
534   texio.write_nl("log", "system(" .. cmdline .. ") ")
535   if os.execute then
536     texio.write("log", "executed.")
537     systemexitstatus = os.execute(cmdline)
538   else
539     texio.write("log", "disabled.")
540   end
541 end
542 function lastssystemstatus()
543   local result = tonumber(systemexitstatus)
544   if result then
545     local x = math.floor(result / 256)
546     tex.write(result - 256 * math.floor(result / 256))
547   end
548 end
549 function lastssystemexit()
550   local result = tonumber(systemexitstatus)
551   if result then
552     tex.write(math.floor(result / 256))
553   end
554 end
555 function pipe(cmdline)
556   local result
557   systemexitstatus = nil
558   texio.write_nl("log", "pipe(" .. cmdline .. ") ")
559   if io.popen then
560     texio.write("log", "executed.")
561     local handle = io.popen(cmdline, "r")
562     if handle then
563       result = handle:read("*a")
564       handle:close()
565     end
566   else
567     texio.write("log", "disabled.")
568   end
569   if result then
570     tex.settoks(toks, result)
571   else
572     tex.settoks(toks, "")
573   end
574 end
575  $\langle$ /lua $\rangle$ 

```

3 Test

3.1 Catcode checks for loading

```

576  $\langle$ *test1 $\rangle$ 
577 \catcode'\{=1 %
578 \catcode'\}=2 %
579 \catcode'\#=6 %
580 \catcode'\@=11 %
581 \expandafter\ifx\csname count@\endcsname\relax
582   \countdef\count@=255 %
583 \fi
584 \expandafter\ifx\csname @gobble\endcsname\relax
585   \long\def\@gobble#1{}%
586 \fi

```

```

587 \expandafter\ifx\csname @firstofone\endcsname\relax
588   \long\def\@firstofone#1{#1}%
589 \fi
590 \expandafter\ifx\csname loop\endcsname\relax
591   \expandafter\@firstofone
592 \else
593   \expandafter\@gobble
594 \fi
595 {%
596   \def\loop#1\repeat{%
597     \def\body{#1}%
598     \iterate
599   }%
600   \def\iterate{%
601     \body
602     \let\next\iterate
603   \else
604     \let\next\relax
605   \fi
606   \next
607 }%
608 \let\repeat=\fi
609 }%
610 \def\RestoreCatcodes{}
611 \count@=0 %
612 \loop
613   \edef\RestoreCatcodes{%
614     \RestoreCatcodes
615     \catcode\the\count@=\the\catcode\count@\relax
616   }%
617 \ifnum\count@<255 %
618   \advance\count@ 1 %
619 \repeat
620
621 \def\RangeCatcodeInvalid#1#2{%
622   \count@=#1\relax
623   \loop
624     \catcode\count@=15 %
625     \ifnum\count@<#2\relax
626       \advance\count@ 1 %
627     \repeat
628 }
629 \expandafter\ifx\csname LoadCommand\endcsname\relax
630   \def\LoadCommand{\input pdftexcmds.sty\relax}%
631 \fi
632 \def\Test{%
633   \RangeCatcodeInvalid{0}{47}%
634   \RangeCatcodeInvalid{58}{64}%
635   \RangeCatcodeInvalid{91}{96}%
636   \RangeCatcodeInvalid{123}{255}%
637   \catcode'\@=12 %
638   \catcode'\=0 %
639   \catcode'\{=1 %
640   \catcode'\}=2 %
641   \catcode'\#=6 %
642   \catcode'\[=12 %
643   \catcode'\]=12 %
644   \catcode'\%=14 %
645   \catcode'\ =10 %
646   \catcode13=5 %
647   \LoadCommand
648   \RestoreCatcodes

```

```

649 }
650 \Test
651 \csname @@end\endcsname
652 \end
653 </test1>

```

4 Installation

4.1 Download

Package. This package is available on CTAN¹:

[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](#)

TDS refers to the standard “A Directory Structure for T_EX 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-T_EX:

```
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>	→ <code>tex/generic/oberdiek/pdftexcmds.sty</code>
<code>oberdiek.pdftexcmds.lua</code>	→ <code>scripts/oberdiek/oberdiek.pdftexcmds.lua</code>
<code>pdftexcmds.lua</code>	→ <code>scripts/oberdiek/pdftexcmds.lua</code>
<code>pdftexcmds.pdf</code>	→ <code>doc/latex/oberdiek/pdftexcmds.pdf</code>
<code>pdftexcmds.dtx</code>	→ <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.

¹<http://ftp.ctan.org/tex-archive/>

4.4 Refresh file name databases

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

4.5 Some details for the interested

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

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

Unpacking with $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$. The `.dtx` chooses its action depending on the format:

plain- $\text{T}_{\text{E}}\text{X}$: Run `docstrip` and extract the files.

$\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$: Generate the documentation.

If you insist on using $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ for `docstrip` (really, `docstrip` does not need $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$), 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 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$:

```
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.

[2009/04/10 v0.4]

- Adaptation for syntax change of `\directlua` in $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 0.36.

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			
\#	579, 641	\endinput	26, 169
\%	644	I	
\@	580, 637	\ifluatex	114
\@PackageInfo	116	\ifnum	171, 191, 293, 617, 625
\@PackageInfoNoLine	118	\ifx	11, 14, 18, 44, 52, 55, 107, 123, 142, 150, 158, 177, 186, 581, 584, 587, 590, 629
\@firstofone	588, 591	\immediate	20, 46, 166
\@gobble	585, 593	\input	108, 109, 178, 630
\@undefined	52	\iterate	598, 600, 602
\[642	L	
\\	413, 416, 638	\LoadCommand	630, 647
\{	577, 639	\loop	596, 612, 623
\}	578, 640	\luaescapestring	197, 198, 203, 208, 215, 222, 227, 232, 237, 242, 247, 252, 253, 254, 259, 264, 269, 279, 304
\]	643	\luatexversion	171, 191
Numbers		N	
\0	193, 292	\next	602, 604, 606
_	645	\number	252, 253
A		P	
\advance	618, 626	\PackageInfo	23
\aftergroup	26	\pdf@escapehex	3, 134, 201
B		\pdf@escapehexnative	134, 206
\body	597, 601	\pdf@escapename	230
C		\pdf@escapenamenative	235
\catcode	3, 4, 5, 6, 7, 8, 9, 17, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 64, 65, 68, 69, 70, 71, 75, 76, 77, 78, 82, 84, 193, 292, 577, 578, 579, 580, 615, 624, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646	\pdf@escapestring	225
\count@	582, 611, 615, 617, 618, 622, 624, 625, 626	\pdf@filedump	3, 153, 250
\countdef	582	\pdf@filemdfivesum	3, 163, 267
\csname	10, 18, 44, 60, 67, 107, 123, 126, 128, 142, 150, 158, 177, 186, 189, 581, 584, 587, 590, 629, 651	\pdf@filemoddate	3, 245
D		\pdf@filesize	3, 240
\directlua	172, 174, 196, 202, 207, 213, 220, 226, 231, 236, 241, 246, 251, 258, 263, 268, 273, 278, 283, 288	\pdf@lastsystemexit	287
E		\pdf@lastsystemstatus	282
\empty	13, 14	\pdf@mdfivesum	3, 161, 162, 257
\end	652	\pdf@mdfivesumnative	162, 262
\endcsname	10, 18, 44, 60, 67, 107, 123, 126, 128, 142, 150, 158, 177, 186, 189, 581, 584, 587, 590, 629, 651	\pdf@pipe	4, 293
		\pdf@shellescape	3, 145, 272
		\pdf@strcmp	2, 195
		\pdf@system	3, 165, 277
		\pdf@unescapehex	2, 136, 211
		\pdf@unescapehexnative	3, 136, 218
		\pdf@filedump	154
		\pdfmdfivesum	161, 163
		\pdfshellescape	146
		\pdftexcmds@AtEnd	80, 81, 168, 308
		\pdftexcmds@directlua	171, 182, 294, 302
		\pdftexcmds@nopdf tex	117, 119, 124, 143, 151, 159
		\pdftexcmds@temp	121, 132, 133, 135, 137, 138, 139, 140
		\pdftexcmds@toks	185, 212, 219, 301
		\ProvidesPackage	15, 61

	R		<code>\TMP@EnsureCode</code> ... 79, 86, 87, 88,
<code>\RangeCatcodeInvalid</code>			89, 90, 91, 92, 93, 94, 95, 96, 97,
..... 621, 633, 634, 635, 636			98, 99, 100, 101, 102, 103, 104, 105
<code>\repeat</code>	596, 608, 619, 627	<code>\toksdef</code>	187
<code>\RequirePackage</code>	111, 112, 180		
<code>\RestoreCatcodes</code> ..	610, 613, 614, 648	W	
		<code>\write</code>	20, 46, 166
	T		
<code>\Test</code>	632, 650	X	
<code>\the</code> 68, 69, 70, 71, 82, 212, 219, 301, 615		<code>\x</code> 10, 11, 14, 19, 23, 25, 45, 50, 60, 66, 74	