

The luacolor package

Heiko Oberdiek*
<heiko.oberdiek at gmail.com>

2016/05/16 v1.10

Abstract

Package `luacolor` implements color support based on LuaTeX's node attributes.

Contents

1	Documentation	2
1.1	Introduction	2
1.2	Usage	2
1.3	Limitations	2
2	Implementation	3
2.1	Catcodes and identification	3
2.2	Check for LuaTeX	3
2.3	Check for disabled colors	4
2.4	Load module and check version	4
2.5	Find driver	5
2.6	Attribute setting	5
2.7	Whatsit insertion	6
2.8	<code>\pdfxform</code> support	6
2.9	Lua module	6
2.9.1	Driver detection	7
2.9.2	Color strings	8
2.9.3	Attribute register	8
2.9.4	Whatsit insertion	8
3	Test	10
3.1	Catcode checks for loading	10
3.2	Driver detection	12
4	Installation	12
4.1	Download	12
4.2	Bundle installation	13
4.3	Package installation	13
4.4	Refresh file name databases	13
4.5	Some details for the interested	13
5	Catalogue	14

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

6 History	14
[2007/12/12 v1.0]	14
[2009/04/10 v1.1]	14
[2010/03/09 v1.2]	15
[2010/12/13 v1.3]	15
[2011/03/29 v1.4]	15
[2011/04/22 v1.5]	15
[2011/04/23 v1.6]	15
[2011/10/22 v1.7]	15
[2011/11/01 v1.8]	15
[2016/05/13 v1.9]	15
[2016/05/16 v1.10]	15

1 Documentation

1.1 Introduction

This package uses a LuaTeX's attribute register to to annotate nodes with color information. If a color is set, then the attribute register is set to this color and all nodes created in its scope (current group) are annotated with this attribute. Now the color property behaves much the same way as the font property.

1.2 Usage

Package `color` is loaded automatically by this package `luacolor`. If you need a special driver option or you prefer package `xcolor`, then load it before package `luacolor`, for example:

```
\usepackage[dvipdfmx]{xcolor}
```

The package `luacolor` is loaded without options:

```
\usepackage{luacolor}
```

It is able to detect PDF mode and DVI drivers are differentiated by its color specials. Therefore the package do need driver options.

Then it redefines the color setting commands to set attributes instead of what-sits for color.

At last the attribute annotations of the nodes in the output box must be analyzed to insert the necessary color what-sits. Currently LuaTeX lacks an appropriate callback function. Therefore package `atbegshi` is used to get control before a box is shipped out.

`\luacolorProcessBox {⟨box⟩}`

Macro `\luacolorProcessBox` processes the box `⟨box⟩` in the previously described manner. It is automatically called for pages, but not for XForm objects. Before passing a box to `\pdfxform`, call `\luacolorProcessBox` first.

1.3 Limitations

Ligatures with different colored components: Package `luacolor` sees the ligature after the paragraph building and page breaking, when a page is to be shipped out. Therefore it cannot break ligatures, because the components might occupy different space. Therefore it is the responsibility of the ligature forming process to deal with different colored glyphs that form a ligature. The user can avoid the problem entirely by explicitly breaking the ligature at the places where the color changes.

...

2 Implementation

```
1 (*package)
```

2.1 Catcodes and identification

```
2 \begingroup\catcode61\catcode48\catcode32=10\relax%
3 \catcode13=5 % ^^M
4 \endlinechar=13 %
5 \catcode123=1 % {
6 \catcode125=2 % }
7 \catcode64=11 % @
8 \def\x{\endgroup
9 \expandafter\edef\csname LuaCol@AtEnd\endcsname{%
10 \endlinechar=\the\endlinechar\relax
11 \catcode13=\the\catcode13\relax
12 \catcode32=\the\catcode32\relax
13 \catcode35=\the\catcode35\relax
14 \catcode61=\the\catcode61\relax
15 \catcode64=\the\catcode64\relax
16 \catcode123=\the\catcode123\relax
17 \catcode125=\the\catcode125\relax
18 }%
19 }%
20 \x\catcode61\catcode48\catcode32=10\relax%
21 \catcode13=5 % ^^M
22 \endlinechar=13 %
23 \catcode35=6 % #
24 \catcode64=11 % @
25 \catcode123=1 % {
26 \catcode125=2 % }
27 \def\TMP@EnsureCode#1#2{%
28 \edef\LuaCol@AtEnd{%
29 \LuaCol@AtEnd
30 \catcode#1=\the\catcode#1\relax
31 }%
32 \catcode#1=#2\relax
33 }
34 \TMP@EnsureCode{34}{12}% "
35 \TMP@EnsureCode{39}{12}% '
36 \TMP@EnsureCode{40}{12}% (
37 \TMP@EnsureCode{41}{12}% )
38 \TMP@EnsureCode{42}{12}% *
39 \TMP@EnsureCode{43}{12}% +
40 \TMP@EnsureCode{44}{12}% ,
41 \TMP@EnsureCode{45}{12}% -
42 \TMP@EnsureCode{46}{12}% .
43 \TMP@EnsureCode{47}{12}% /
44 \TMP@EnsureCode{58}{12}% :
45 \TMP@EnsureCode{60}{12}% <
46 \TMP@EnsureCode{62}{12}% >
47 \TMP@EnsureCode{91}{12}% [
48 \TMP@EnsureCode{93}{12}% ]
49 \TMP@EnsureCode{95}{12}% _ (other!)
50 \TMP@EnsureCode{96}{12}% `
51 \edef\LuaCol@AtEnd{\LuaCol@AtEnd\noexpand\endinput}

Package identification.
52 \NeedsTeXFormat{LaTeX2e}
53 \ProvidesPackage{luacolor}%
54 [2016/05/16 v1.10 Color support via LuaTeX's attributes (HO)]
```

2.2 Check for LuaTeX

Without LuaTeX there is no point in using this package.

```

55 \RequirePackage{infwarerr}[2010/04/08]%
56 \RequirePackage{ifluatex}[2010/03/01]%
57 \RequirePackage{ifpdf}[2011/01/30]%
58 \RequirePackage{ltxcmds}[2011/04/18]%
59 \RequirePackage{color}

60 \ifluatex
61   \ifx\newattribute\@undefined
62     \ltx@ifpackageloaded{luatexbase-attr}{%
63       }{%
64         \RequirePackage{luatex}[2010/03/09]%
65       }%
66     \fi
67   \else
68     \@PackageError{luacolor}{%
69       This package may only be run using LuaTeX%
70     }\@ehc
71     \expandafter\LuaCol@AtEnd
72   \fi%

```

\LuaCol@directlua

```

73 \ifnum\luatexversion<36 %
74   \def\LuaCol@directlua{\directlua0 }%
75 \else
76   \let\LuaCol@directlua\directlua
77 \fi

```

2.3 Check for disabled colors

```

78 \ifcolors@
79 \else
80   \@PackageWarningNoLine{luacolor}{%
81     Colors are disabled by option `monochrome'%
82   }%
83   \def\set@color{}%
84   \def\reset@color{}%
85   \def\set@page@color{}%
86   \def\define@color#1#2{}%
87   \expandafter\LuaCol@AtEnd
88 \fi%

```

2.4 Load module and check version

```

89 \LuaCol@directlua{%
90   require("oberdiek.luacolor\ifnum\luatexversion<65 -pre065\fi")%
91 }

92 \begingroup
93   \edef\x{\LuaCol@directlua{tex.write("2016/05/16 v1.10")}}%
94   \edef\y{%
95     \LuaCol@directlua{%
96       if oberdiek.luacolor.getversion then %
97         oberdiek.luacolor.getversion()%
98       end%
99     }%
100   }%
101   \ifx\x\y
102   \else
103     \@PackageError{luacolor}{%
104       Wrong version of lua module.\MessageBreak
105       Package version: \x\MessageBreak
106       Lua module: \y
107     }\@ehc
108   \fi

```

```
109 \endgroup
```

2.5 Find driver

```
110 \ifpdf
111 \else
112   \begingroup
113     \def\current@color{}%
114     \def\reset@color{}%
115     \setbox\z@=\hbox{%
116       \begingroup
117         \set@color
118       \endgroup
119     }%
120     \edef\reserved@a{%
121       \LuaCol@directlua{%
122         oberdiek.luacolor.dvidetect()}%
123     }%
124   }%
125   \ifx\reserved@a\@empty
126     \@PackageError{luacolor}{%
127       DVI driver detection failed because of\MessageBreak
128       unrecognized color \string\special
129     }\@ehc
130   \endgroup
131   \expandafter\expandafter\expandafter\LuaCol@AtEnd
132 \else
133   \@PackageInfoNoLine{luacolor}{%
134     Type of color \string\special: \reserved@a
135   }%
136 \fi%
137 \endgroup
138 \fi
```

2.6 Attribute setting

\LuaCol@Attribute

```
139 \ltx@ifundefined{newluatexattribute}{%
140   \newattribute\LuaCol@Attribute
141 }{%
142   \newluatexattribute\LuaCol@Attribute
143 }
144 \ltx@ifundefined{setluatexattribute}{%
145   \let\LuaCol@setattribute\setattribute
146 }{%
147   \let\LuaCol@setattribute\setluatexattribute
148 }
149 \LuaCol@directlua{%
150   oberdiek.luacolor.setattribute(\number\allocationnumber)%
151 }
```

\set@color

```
152 \protected\def\set@color{%
153   \LuaCol@setattribute\LuaCol@Attribute{%
154     \LuaCol@directlua{%
155       oberdiek.luacolor.get("\luaescapestring{\current@color}")%
156     }%
157   }%
158 }
```

\reset@color

```
159 \def\reset@color{}
```

2.7 Whatsit insertion

```
\luacolorProcessBox

160 \def\luacolorProcessBox#1{%
161   \LuaCol@directlua{%
162     oberdiek.luacolor.process(\number#1)%
163   }%
164 }

165 \RequirePackage{atbegshi}[2011/01/30]
166 \AtBeginShipout{%
167   \luacolorProcessBox\AtBeginShipoutBox
168 }

    Set default color.
169 \set@color
```

2.8 \pdfxform support

```
170 \ifpdf
171   \ifx\pdfxform\@undefined
172     \let\pdfxform\saveboxresource
173   \fi
174   \ltx@ifundefined{pdfxform}{%
175     \ifnum\luatexversion>36 %
176       \directlua{%
177         tex.enableprimitives('{%
178           'pdfxform','pdflastxform','pdfrefxform'%
179         })%
180       }%
181     \fi
182   }{%}%
183   \ltx@ifundefined{protected}{%
184     \ifnum\luatexversion>36 %
185       \directlua{tex.enableprimitives('{{'protected'}}}%
186     \fi
187   }{%}%
188   \ltx@ifundefined{pdfxform}{%
189     \@PackageWarning{luacolor}{\string\pdfxform\space not found}%
190   }{%%
191     \let\LuaCol@org@pdfxform\pdfxform
192     \begingroup\expandafter\expandafter\expandafter\endgroup
193     \expandafter\ifx\csname protected\endcsname\relax
194       \@PackageWarning{luacolor}{\string\protected\space not found}%
195     \else
196       \expandafter\protected
197     \fi
198     \def\pdfxform{%
199       \begingroup
200       \afterassignment\LuaCol@pdfxform
201       \count@=%
202     }%
203     \def\LuaCol@pdfxform{%
204       \luacolorProcessBox\count@
205       \LuaCol@org@pdfxform\count@
206     \endgroup
207   }%
208 }%
209 \fi

210 \LuaCol@AtEnd%
211 \endpackage
```

2.9 Lua module

```

212  $\langle *lua \rangle$ 

Box zero contains a  $\backslash hbox$  with the color  $\backslash special$ . That is analyzed to get the
prefix for the color setting  $\backslash special$ .
213 module("oberdiek.luacolor", package.seeall)

getversion()

214 function getversion()
215   tex.write("2016/05/16 v1.10")
216 end

2.9.1 Driver detection

217 local ifpdf
218 if tonumber(tex.outputmode or tex.pdfoutput) > 0 then
219   ifpdf = true
220 else
221   ifpdf = false
222 end
223 local prefix
224 local prefixes = {
225   dvips = "color ",
226   dvipdfm = "pdf:sc ",
227   truetex = "textcolor:",
228   pctexps = "ps::",
229 }
230 local patterns = {
231   [ "^color " ] = "dvips",
232   [ "^pdf: *begincolor " ] = "dvipdfm",
233   [ "^pdf: *bcolor " ] = "dvipdfm",
234   [ "^pdf: *bc " ] = "dvipdfm",
235   [ "^pdf: *setcolor " ] = "dvipdfm",
236   [ "^pdf: *scolor " ] = "dvipdfm",
237   [ "^pdf: *sc " ] = "dvipdfm",
238   [ "^textcolor:" ] = "truetex",
239   [ "^ps::" ] = "pctexps",
240 }

info()

241 local function info(msg, term)
242   local target = "log"
243   if term then
244     target = "term and log"
245   end
246   texio.write_nl(target, "Package luacolor info: " .. msg .. ".")
247   texio.write_nl(target, "")
248 end

dvidetect()

249 function dvidetect()
250   local v = tex.box[0]
251   assert(v.id == node.id("hlist"))
252    $\langle !pre065 \rangle$  for v in node.traverse_id(node.id("whatsit"), v.head) do
253    $\langle pre065 \rangle$  for v in node.traverse_id(node.id("whatsit"), v.list) do
254     if v and v.subtype == node.subtype("special") then
255       local data = v.data
256       for pattern, driver in pairs(patterns) do
257         if string.find(data, pattern) then
258           prefix = prefixes[driver]
259           tex.write(driver)
260           return
261         end
262       end
263       info("\\special{" .. data .. "}", true)

```

```

264     return
265   end
266 end
267 info("Missing \\special", true)
268 end

```

2.9.2 Color strings

```

269 local map = {
270   n = 0,
271 }

get()

272 function get(color)
273   tex.write("" .. getvalue(color))
274 end

getvalue()

275 function getvalue(color)
276   local n = map[color]
277   if not n then
278     n = map.n + 1
279     map.n = n
280     map[n] = color
281     map[color] = n
282   end
283   return n
284 end

```

2.9.3 Attribute register

```

setattribute()

285 local attribute
286 function setattribute(attr)
287   attribute = attr
288 end

getattribute()

289 function getattribute()
290   return attribute
291 end

```

2.9.4 Whatsit insertion

```

292 local LIST = 1
293 local LIST_LEADERS = 2
294 local COLOR = 3
295 local RULE = node.id("rule")
296 local node_types = {
297   [node.id("hlist")] = LIST,
298   [node.id("vlist")] = LIST,
299   [node.id("rule")] = COLOR,
300   [node.id("glyph")] = COLOR,
301   [node.id("disc")] = COLOR,
302   [node.id("whatsit")] = {
303     [node.subtype("special")] = COLOR,
304     [node.subtype("pdf_literal")] = COLOR,
305     -- TODO (DPC) [node.subtype("pdf_refximage")] = COLOR,
306   },
307   [node.id("glue")] =
308     function(n)
309       if n.subtype >= 100 then -- leaders

```



```

310     if n.leader.id == RULE then
311         return COLOR
312     else
313         return LIST_LEADERS
314     end
315 end
316 end,
317 }

get_type()

318 local function get_type(n)
319     local ret = node_types[n.id]
320     if type(ret) == 'table' then
321         ret = ret[n.subtype]
322     end
323     if type(ret) == 'function' then
324         ret = ret(n)
325     end
326     return ret
327 end

328 local mode = 2 -- luatex.pdfliteral.direct
329 local WHATSIT = node.id("whatsit")
330 local SPECIAL = node.subtype("special")
331 local PDFLITERAL = node.subtype("pdf_literal")
332 local DRY_FALSE = false
333 local DRY_TRUE = true

traverse()

334 local function traverse(list, color, dry)
335     if not list then
336         return color
337     end
338     if get_type(list) ~= LIST then
339         texio.write_nl("!!! Error: Wrong list type: " .. node.type(list.id))
340         return color
341     end
342     <debug>texio.write_nl("traverse: " .. node.type(list.id))
343     <!pre065> local head = list.head
344     <pre065> local head = list.list
345     for n in node.traverse(head) do
346         <debug>texio.write_nl(" node: " .. node.type(n.id))
347         local t = get_type(n)
348         if t == LIST then
349             color = traverse(n, color, dry)
350         elseif t == LIST_LEADERS then
351             local color_after = traverse(n.leader, color, DRY_TRUE)
352             if color == color_after then
353                 traverse(n.leader, color, DRY_FALSE or dry)
354             else
355                 traverse(n.leader, "", DRY_FALSE or dry)
356             % The color status is unknown here, because the leader box
357             % will or will not be set.
358             color = ""
359         end
360         elseif t == COLOR then
361             local v = node.has_attribute(n, attribute)
362             if v then
363                 local newColor = map[v]
364                 if newColor ~= color then
365                     color = newColor
366                     if dry == DRY_FALSE then
367                         local newNode

```

```

368         if ifpdf then
369             newNode = node.new(WHATSIT, PDFLITERAL)
370             newNode.mode = mode
371             newNode.data = color
372         else
373             newNode = node.new(WHATSIT, SPECIAL)
374             newNode.data = prefix .. color
375         end
376 <!*pre065>
377         head = node.insert_before(head, n, newNode)
378 </!pre065>
379 <!*pre065>
380         if head == n then
381             newNode.next = head
382             local old_prev = head.prev
383             head.prev = newNode
384             head = newNode
385             head.prev = old_prev
386         else
387             head = node.insert_before(head, n, newNode)
388         end
389 </pre065>
390     end
391 end
392 end
393 end
394 end
395 <!pre065> list.head = head
396 <pre065> list.list = head
397 return color
398 end

```

process()

```

399 function process(box)
400     local color = ""
401     local list = tex.getbox(box)
402     traverse(list, color, DRY_FALSE)
403 end
404 </lua>

```

3 Test

```

405 <!*test1>
406 \documentclass{article}
407 \usepackage{color}
408 </test1>

```

3.1 Catcode checks for loading

```

409 <!*test1>
410 \catcode`\{=1 %
411 \catcode`\}=2 %
412 \catcode`\#=6 %
413 \catcode`\@=11 %
414 \expandafter\ifx\csname count@\endcsname\relax
415     \countdef\count@=255 %
416 \fi
417 \expandafter\ifx\csname @gobble\endcsname\relax
418     \long\def\@gobble#1{}%
419 \fi
420 \expandafter\ifx\csname @firstofone\endcsname\relax

```

```

421 \long\def\@firstofone#1{#1}%
422 \fi
423 \expandafter\ifx\csname loop\endcsname\relax
424 \expandafter\@firstofone
425 \else
426 \expandafter\@gobble
427 \fi
428 {%
429 \def\loop#1\repeat{%
430 \def\body{#1}%
431 \iterate
432 }%
433 \def\iterate{%
434 \body
435 \let\next\iterate
436 \else
437 \let\next\relax
438 \fi
439 \next
440 }%
441 \let\repeat=\fi
442 }%
443 \def\RestoreCatcodes{}
444 \count@=0 %
445 \loop
446 \edef\RestoreCatcodes{%
447 \RestoreCatcodes
448 \catcode\the\count@=\the\catcode\count@\relax
449 }%
450 \ifnum\count@<255 %
451 \advance\count@ 1 %
452 \repeat
453
454 \def\RangeCatcodeInvalid#1#2{%
455 \count@=#1\relax
456 \loop
457 \catcode\count@=15 %
458 \ifnum\count@<#2\relax
459 \advance\count@ 1 %
460 \repeat
461 }
462 \def\RangeCatcodeCheck#1#2#3{%
463 \count@=#1\relax
464 \loop
465 \ifnum#3=\catcode\count@
466 \else
467 \errmessage{%
468 Character \the\count@\space
469 with wrong catcode \the\catcode\count@\space
470 instead of \number#3%
471 }%
472 \fi
473 \ifnum\count@<#2\relax
474 \advance\count@ 1 %
475 \repeat
476 }
477 \def\space{ }
478 \expandafter\ifx\csname LoadCommand\endcsname\relax
479 \def\LoadCommand{\input luacolor.sty\relax}%
480 \fi
481 \def\Test{%
482 \RangeCatcodeInvalid{0}{47}%

```

```

483 \RangeCatcodeInvalid{58}{64}%
484 \RangeCatcodeInvalid{91}{96}%
485 \RangeCatcodeInvalid{123}{255}%
486 \catcode`\@=12 %
487 \catcode`\=0 %
488 \catcode`\%=14 %
489 \LoadCommand
490 \RangeCatcodeCheck{0}{36}{15}%
491 \RangeCatcodeCheck{37}{37}{14}%
492 \RangeCatcodeCheck{38}{47}{15}%
493 \RangeCatcodeCheck{48}{57}{12}%
494 \RangeCatcodeCheck{58}{63}{15}%
495 \RangeCatcodeCheck{64}{64}{12}%
496 \RangeCatcodeCheck{65}{90}{11}%
497 \RangeCatcodeCheck{91}{91}{15}%
498 \RangeCatcodeCheck{92}{92}{0}%
499 \RangeCatcodeCheck{93}{96}{15}%
500 \RangeCatcodeCheck{97}{122}{11}%
501 \RangeCatcodeCheck{123}{255}{15}%
502 \RestoreCatcodes
503 }
504 \Test
505 \csname @@end\endcsname
506 \end
507 </test1>

```

3.2 Driver detection

```

508 <*test2>
509 \NeedsTeXFormat{LaTeX2e}
510 \ifcsname driver\endcsname
511 \expandafter\PassOptionsToPackage\expandafter{\driver}{color}%
512 \pdfoutput=0 %
513 \fi
514 \documentclass{minimal}
515 \usepackage{luacolor}[2016/05/16]
516 \csname @@end\endcsname
517 \end
518 </test2>

519 <*test3>
520 \NeedsTeXFormat{LaTeX2e}

521 \documentclass{minimal}
522 \usepackage{luacolor}[2016/05/16]
523 \usepackage{qstest}
524 \IncludeTests{*}
525 \LogTests{log}{*}{*}
526 \makeatletter

527 \@@end
528 </test3>

```

4 Installation

4.1 Download

Package. This package is available on CTAN¹:

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

[CTAN:macros/latex/contrib/oberdiek/luacolor.pdf](http://ctan.org/macros/latex/contrib/oberdiek/luacolor.pdf) Documentation.

¹<http://ctan.org/pkg/luacolor>

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 luacolor.dtx
```

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

<code>luacolor.sty</code>	→ <code>tex/latex/oberdiek/luacolor.sty</code>
<code>oberdiek.luacolor.lua</code>	→ <code>scripts/oberdiek/oberdiek.luacolor.lua</code>
<code>luacolor.lua</code>	→ <code>scripts/oberdiek/luacolor.lua</code>
<code>oberdiek.luacolor-pre065.lua</code>	→ <code>scripts/oberdiek/oberdiek.luacolor-pre065.lua</code>
<code>luacolor-pre065.lua</code>	→ <code>scripts/oberdiek/luacolor-pre065.lua</code>
<code>luacolor.pdf</code>	→ <code>doc/latex/oberdiek/luacolor.pdf</code>
<code>test/luacolor-test1.tex</code>	→ <code>doc/latex/oberdiek/test/luacolor-test1.tex</code>
<code>test/luacolor-test2.tex</code>	→ <code>doc/latex/oberdiek/test/luacolor-test2.tex</code>
<code>test/luacolor-test3.tex</code>	→ <code>doc/latex/oberdiek/test/luacolor-test3.tex</code>
<code>luacolor.dtx</code>	→ <code>source/latex/oberdiek/luacolor.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 T_EX distribution (teT_EX, mikT_EX, ...) relies on file name databases, you must refresh these. For example, teT_EX users run `texhash` or `mktextlsr`.

4.5 Some details for the interested

Unpacking with L^AT_EX. The `.dtx` chooses its action depending on the format:

plain T_EX: Run `docstrip` and extract the files.

L^AT_EX: Generate the documentation.

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

```
latex \let\install=y\input{luacolor.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^AT_EX:

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

5 Catalogue

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

```
529 (*catalogue)
530 <?xml version='1.0' encoding='us-ascii'?>
531 <!DOCTYPE entry SYSTEM 'catalogue.dtd'>
532 <entry datestamp='$Date$' modifier='$Author$' id='luacolor'>
533   <name>luacolor</name>
534   <caption>Color support based on LuaTeX's node attributes.</caption>
535   <authorref id='auth:oberdiek' />
536   <copyright owner='Heiko Oberdiek' year='2007,2009-2011' />
537   <license type='lppl1.3' />
538   <version number='1.10' />
539   <description>
540     This package implements color support based on LuaTeX's node
541     attributes.
542   </p>
543   The package is part of the <xref refid='oberdiek'>oberdiek</xref> bundle.
544 </description>
545 <documentation details='Package documentation'
546   href='ctan:/macros/latex/contrib/oberdiek/luacolor.pdf' />
547 <ctan file='true' path='/macros/latex/contrib/oberdiek/luacolor.dtx' />
548 <miktex location='oberdiek' />
549 <texlive location='oberdiek' />
550 <install path='/macros/latex/contrib/oberdiek/oberdiek.tds.zip' />
551 </entry>
552 </catalogue>
```

6 History

[2007/12/12 v1.0]

- First public version.

[2009/04/10 v1.1]

- Fixes for changed syntax of `\directlua` in LuaT_EX 0.36.

[2010/03/09 v1.2]

- Adaptation for package `luatex` 2010/03/09 v0.4.

[2010/12/13 v1.3]

- Support for `\pdfxform` added.
- Loaded package `luatexbase-attr` recognized.
- Update for LuaTeX: ‘list’ fields renamed to ‘head’ in v0.65.0.

[2011/03/29 v1.4]

- Avoid whatsit insertion if option `monochrome` is used (thanks Manuel Pégourié-Gonnard).

[2011/04/22 v1.5]

- Bug fix by Manuel Pégourié-Gonnard: A typo prevented the detection of whatsits and applying color changes for `\pdfiternal` and `\special` nodes that might contain typesetting material.
- Bug fix by Manuel Pégourié-Gonnard: Now colors are also applied to leader boxes.
- Unnecessary color settings are removed for leaders boxes, if after the leader box the color has not changed. The costs are a little runtime, leader boxes are processed twice.
- Additional whatsits that are colored: `pdf_refximage`.
- Workaround for bug with `node.insert_before` removed for the version after LuaTeX 0.65, because bug was fixed in 0.27. (Thanks Manuel Pégourié-Gonnard.)

[2011/04/23 v1.6]

- Bug fix for nested leader boxes.
- Bug fix for leader boxes that change color, but are not set because of missing place.
- Version check for Lua module added.

[2011/10/22 v1.7]

- Lua functions `getattribute` and `getvalue` added to tell other external Lua functions the attribute register number for coloring.

[2011/11/01 v1.8]

- Use of `node.subtype` instead of magic numbers.

[2016/05/13 v1.9]

- More use of `node.subtype` instead of magic numbers.
- `luatex` 85 updates

[2016/05/16 v1.10]

- Documentation updates.

7 Index

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

Symbols	G
\# 412	\get() <u>272</u>
\% 488	\get_type() <u>318</u>
\@ 413, 486	\getattribute() <u>289</u>
\@@end 527	\getvalue() <u>275</u>
\@PackageError 68, 103, 126	\getversion() <u>214</u>
\@PackageInfoNoLine 133	
\@PackageWarning 189, 194	H
\@PackageWarningNoLine 80	\hbox 115
\@ehc 70, 107, 129	
\@empty 125	I
\@firstofone 421, 424	\ifcolors@ 78
\@gobble 418, 426	\ifcsname 510
\@undefined 61, 171	\ifuatex 60
\\ 263, 267, 487	\ifnum 73, 90, 175, 184, 450, 458, 465, 473
\{ 410	\ifpdf 110, 170
\} 411	\ifx 61, 101, 125, 171, 193, 414, 417, 420, 423, 478
A	\IncludeTests 524
\advance 451, 459, 474	\info() <u>241</u>
\afterassignment 200	\input 479
\allocationnumber 150	\iterate 431, 433, 435
\AtBeginShipout 166	
\AtBeginShipoutBox 167	L
B	\LoadCommand 479, 489
\body 430, 434	\LogTests 525
C	\loop 429, 445, 456, 464
\catcode 2, 3, 5, 6, 7, 11, 12, 13, 14, 15, 16, 17, 20, 21, 23, 24, 25, 26, 30, 32, 410, 411, 412, 413, 448, 457, 465, 469, 486, 487, 488	\ltx@ifpackageloaded 62
\count@ 201, 204, 205, 415, 444, 448, 450, 451, 455, 457, 458, 459, 463, 465, 468, 469, 473, 474	\ltx@ifUndefined 139, 144, 174, 183, 188
\countdef 415	\LuaCol@AtEnd 28, 29, 51, 71, 87, 131, 210
\csname 9, 193, 414, 417, 420, 423, 478, 505, 516	\LuaCol@Attribute <u>139</u> , 153
\current@color 113, 155	\LuaCol@directlua 73, 89, 93, 95, 121, 149, 154, 161
D	\LuaCol@org@pdfxform 191, 205
\define@color 86	\LuaCol@pdfxform 200, 203
\directlua 74, 76, 176, 185	\LuaCol@setattribute ... 145, 147, 153
\documentclass 406, 514, 521	\luacolorProcessBox .. 2, <u>160</u> , 167, 204
\driver 511	\luaescapestring 155
\dvidetect() <u>249</u>	\luatexversion 73, 90, 175, 184
E	M
\end 506, 517	\makeatletter 526
\endcsname 9, 193, 414, 417, 420, 423, 478, 505, 510, 516	\MessageBreak 104, 105, 127
\endinput 51	N
\endlinechar 4, 10, 22	\NeedsTeXFormat 52, 509, 520
\errmessage 467	\newattribute 61, 140
	\newluatexattribute 142
	\next 435, 437, 439
	\number 150, 162, 470
	P
	\PassOptionsToPackage 511
	\pdfoutput 512
	\pdfxform 171, 172, 189, 191, 198
	\process() <u>399</u>

<code>\protected</code>	152, 194, 196	<code>\space</code>	189, 194, 468, 469, 477
<code>\ProvidesPackage</code>	53	<code>\special</code>	128, 134
R		T	
<code>\RangeCatcodeCheck</code>		<code>\Test</code>	481, 504
.	462, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501	<code>\the</code>	10, 11, 12, 13, 14, 15, 16, 17, 30, 448, 468, 469
<code>\RangeCatcodeInvalid</code>		<code>\TMP@EnsureCode</code>	27, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50
.	454, 482, 483, 484, 485	<code>\traverse()</code>	<u>334</u>
<code>\repeat</code>	429, 441, 452, 460, 475	U	
<code>\RequirePackage</code> 55, 56, 57, 58, 59, 64, 165		<code>\usepackage</code>	407, 515, 522, 523
<code>\reserved@a</code>	120, 125, 134	X	
<code>\reset@color</code>	84, 114, <u>159</u>	<code>\x</code>	8, 20, 93, 101, 105
<code>\RestoreCatcodes</code>	443, 446, 447, 502	Y	
S		<code>\y</code>	94, 101, 106
<code>\saveboxresource</code>	172	Z	
<code>\set@color</code>	83, 117, <u>152</u> , 169	<code>\z@</code>	115
<code>\set@page@color</code>	85		
<code>\setattribute</code>	145		
<code>\setattribute()</code>	<u>285</u>		
<code>\setbox</code>	115		
<code>\setluatexattribute</code>	147		