

The luacolor package

Heiko Oberdiek
<heiko.oberdiek at gmail.com>

2011/11/01 v1.8

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	\pdfxform 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
6	History	14
	[2007/12/12 v1.0]	14
	[2009/04/10 v1.1]	15
	[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

7 Index	16
----------------	-----------

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 [2011/11/01 v1.8 Color support via LuaTeX's attributes (H0)]
```

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

```

\LuaCol@directlua

```

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

```

2.3 Check for disabled colors

```

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

```

2.4 Load module and check version

```

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

90 \begingroup
91 \edef\x{\LuaCol@directlua{tex.write("2011/11/01 v1.8")}}%
92 \edef\y{%
93 \LuaCol@directlua{%
94 if oberdiek.luacolor.getversion then %
95 oberdiek.luacolor.getversion()%
96 end%
97 }%
98 }%
99 \ifx\x\y
100 \else
101 \@PackageError{luacolor}{%
102 Wrong version of lua module.\MessageBreak
103 Package version: \x\MessageBreak
104 Lua module: \y
105 }\@ehc
106 \fi
107 \endgroup

```

2.5 Find driver

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

2.6 Attribute setting

\LuaCol@Attribute

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

\set@color

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

\reset@color

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

2.7 Whatsit insertion

`\luacolorProcessBox`

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

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

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

2.8 \pdfxform support

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

205 \LuaCol@AtEnd%
206 </package>
```

2.9 Lua module

```
207 <*lua>
```

Box zero contains a `\hbox` with the color `\special`. That is analyzed to get the prefix for the color setting `\special`.

```
208 module("oberdiek.luacolor", package.seeall)

getversion()

209 function getversion()
210   tex.write("2011/11/01 v1.8")
211 end
```

2.9.1 Driver detection

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

info()

236 local function info(msg, term)
237   local target = "log"
238   if term then
239     target = "term and log"
240   end
241   texio.write_nl(target, "Package luacolor info: " .. msg .. ".")
242   texio.write_nl(target, "")
243 end

dvidetect()

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

```

260     end
261   end
262   info("Missing \\special", true)
263 end

```

2.9.2 Color strings

```

264 local map = {
265   n = 0,
266 }

get()

267 function get(color)
268   tex.write("'" .. getvalue(color))
269 end

```

```

getvalue()

270 function getvalue(color)
271   local n = map[color]
272   if not n then
273     n = map.n + 1
274     map.n = n
275     map[n] = color
276     map[color] = n
277   end
278   return n
279 end

```

2.9.3 Attribute register

```

setattribute()

280 local attribute
281 function setattribute(attr)
282   attribute = attr
283 end

```

```

getattribute()

284 function getattribute()
285   return attribute
286 end

```

2.9.4 Whatsit insertion

```

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

```



```

306         return COLOR
307     else
308         return LIST_LEADERS
309     end
310 end
311 end,
312 }

get_type()
313 local function get_type(n)
314     local ret = node_types[n.id]
315     if type(ret) == 'table' then
316         ret = ret[n.subtype]
317     end
318     if type(ret) == 'function' then
319         ret = ret(n)
320     end
321     return ret
322 end

323 local mode = 2 -- luatex.pdfliteral.direct
324 local WHATSIT = node.id("whatsit")
325 local SPECIAL = 3
326 local PDFLITERAL = 8
327 local DRY_FALSE = false
328 local DRY_TRUE = true

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

```

```

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

```

process()

```

394 function process(box)
395     local color = ""
396     local list = tex.getbox(box)
397     traverse(list, color, DRY_FALSE)
398 end
399 */lua)

```

3 Test

```

400 */test1)
401 \documentclass{article}
402 \usepackage{color}
403 */test1)

```

3.1 Catcode checks for loading

```

404 */test1)
405 \catcode`\{=1 %
406 \catcode`\}=2 %
407 \catcode`\#=6 %
408 \catcode`\@=11 %
409 \expandafter\ifx\csname count@\endcsname\relax
410     \countdef\count@=255 %
411 \fi
412 \expandafter\ifx\csname @gobble\endcsname\relax
413     \long\def@gobble#1{#1}%
414 \fi
415 \expandafter\ifx\csname @firstofone\endcsname\relax
416     \long\def@firstofone#1{#1}%

```

```

417 \fi
418 \expandafter\ifx\csname loop\endcsname\relax
419 \expandafter\@firstofone
420 \else
421 \expandafter\@gobble
422 \fi
423 {%
424 \def\loop#1\repeat{%
425 \def\body{#1}%
426 \iterate
427 }%
428 \def\iterate{%
429 \body
430 \let\next\iterate
431 \else
432 \let\next\relax
433 \fi
434 \next
435 }%
436 \let\repeat=\fi
437 }%
438 \def\RestoreCatcodes{}
439 \count@=0 %
440 \loop
441 \edef\RestoreCatcodes{%
442 \RestoreCatcodes
443 \catcode\the\count@=\the\catcode\count@\relax
444 }%
445 \ifnum\count@<255 %
446 \advance\count@ 1 %
447 \repeat
448
449 \def\RangeCatcodeInvalid#1#2{%
450 \count@=#1\relax
451 \loop
452 \catcode\count@=15 %
453 \ifnum\count@<#2\relax
454 \advance\count@ 1 %
455 \repeat
456 }
457 \def\RangeCatcodeCheck#1#2#3{%
458 \count@=#1\relax
459 \loop
460 \ifnum#3=\catcode\count@
461 \else
462 \errmessage{%
463 Character \the\count@\space
464 with wrong catcode \the\catcode\count@\space
465 instead of \number#3%
466 }%
467 \fi
468 \ifnum\count@<#2\relax
469 \advance\count@ 1 %
470 \repeat
471 }
472 \def\space{ }
473 \expandafter\ifx\csname LoadCommand\endcsname\relax
474 \def\LoadCommand{\input luacolor.sty\relax}%
475 \fi
476 \def\Test{%
477 \RangeCatcodeInvalid{0}{47}%
478 \RangeCatcodeInvalid{58}{64}%

```

```

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

```

3.2 Driver detection

```

503 <*test2>
504 \NeedsTeXFormat{LaTeX2e}
505 \ifcsname driver\endcsname
506 \expandafter\PassOptionsToPackage\expandafter{\driver}{color}%
507 \pdfoutput=0 %
508 \fi
509 \documentclass{minimal}
510 \usepackage{luacolor}[2011/11/01]
511 \csname @@end\endcsname
512 \end
513 </test2>

514 <*test3>
515 \NeedsTeXFormat{LaTeX2e}

516 \documentclass{minimal}
517 \usepackage{luacolor}[2011/11/01]
518 \usepackage{qstest}
519 \IncludeTests{*}
520 \LogTests{log}{*}{*}
521 \makeatletter

522 \@@end
523 </test3>

```

4 Installation

4.1 Download

Package. This package is available on CTAN¹:

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

[CTAN:macros/latex/contrib/oberdiek/luacolor.pdf](#) Documentation.

¹<ftp://ftp.ctan.org/tex-archive/>

Bundle. All the packages of the bundle ‘oberdiek’ are also available in a TDS compliant ZIP archive. There the packages are already unpacked and the documentation files are generated. The files and directories obey the TDS standard.

CTAN: [install/macros/latex/contrib/oberdiek.tds.zip](#)

TDS refers to the standard “A Directory Structure for 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, miK_TE_X, ...) relies on file name databases, you must refresh these. For example, teT_EX users run `texhash` or `mktexlsr`.

4.5 Some details for the interested

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

```
pdftk luacolor.pdf unpack_files output .
```

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`.

```
524 (*catalogue)
525 <?xml version='1.0' encoding='us-ascii'?>
526 <!DOCTYPE entry SYSTEM 'catalogue.dtd'>
527 <entry datestamp='$Date$' modifier='$Author$' id='luacolor'>
528   <name>luacolor</name>
529   <caption>Color support based on LuaTeX's node attributes.</caption>
530   <authorref id='auth:oberdiek' />
531   <copyright owner='Heiko Oberdiek' year='2007,2009-2011' />
532   <license type='lpl1.3' />
533   <version number='1.8' />
534   <description>
535     This package implements color support based on LuaTeX's node
536     attributes.
537   <p/>
538   The package is part of the <xref refid='oberdiek'>oberdiek</xref> bundle.
539 </description>
540 <documentation details='Package documentation'
541   href='ctan:/macros/latex/contrib/oberdiek/luacolor.pdf' />
542 <ctan file='true' path='/macros/latex/contrib/oberdiek/luacolor.dtx' />
543 <miktex location='oberdiek' />
544 <texlive location='oberdiek' />
545 <install path='/macros/latex/contrib/oberdiek/oberdiek.tds.zip' />
546 </entry>
547 </catalogue>
```

6 History

[2007/12/12 v1.0]

- First public version.

[2009/04/10 v1.1]

- Fixes for changed syntax of `\directlua` in LuaTeX 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 `\pdfliteral` 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.

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	
\#	407
\%	483
\@	408, 481
\@end	522
\@PackageError	66, 101, 124
\@PackageInfoNoLine	131
\@PackageWarning	184, 189
\@PackageWarningNoLine	78
\@ehc	68, 105, 127
\@empty	123
\@firstofone	416, 419
\@gobble	413, 421
\\	258, 262, 482
\{	405
\}	406
A	
\advance	446, 454, 469
\afterassignment	195
\allocationnumber	148
\AtBeginShipout	164
\AtBeginShipoutBox	165
B	
\body	425, 429
C	
\catcode	2, 3, 5, 6, 7, 11, 12, 13, 14, 15, 16, 17, 20, 21, 23, 24, 25, 26, 30, 32, 405, 406, 407, 408, 443, 452, 460, 464, 481, 482, 483
\count@	196, 199, 200, 410, 439, 443, 445, 446, 450, 452, 453, 454, 458, 460, 463, 464, 468, 469
\countdef	410
\csname	9, 188, 409, 412, 415, 418, 473, 500, 511
\current@color	111, 153
D	
\define@color	84
\directlua	72, 74, 171, 180
\documentclass	401, 509, 516
\driver	506
\dvidetect()	<u>244</u>
E	
\end	501, 512
\endcsname	9, 188, 409, 412, 415, 418, 473, 500, 505, 511
\endinput	51
\endlinechar	4, 10, 22
\errmessage	462
G	
\get()	<u>267</u>
\get_type()	313
\getattribute()	284
\getvalue()	270
\getversion()	<u>209</u>
H	
\hbox	113
I	
\ifcolors@	76
\ifcsname	505
\ifluatex	60
\ifnum	71, 88, 170, 179, 445, 453, 460, 468
\ifpdf	108, 168
\ifx	99, 123, 188, 409, 412, 415, 418, 473
\IncludeTests	519
\info()	236
\input	474
\iterate	426, 428, 430
L	
\LoadCommand	474, 484
\LogTests	520
\loop	424, 440, 451, 459
\ltx@ifpackageloaded	61
\ltx@ifundefined	137, 142, 169, 178, 183
\LuaCol@AtEnd	28, 29, 51, 69, 85, 129, 205
\LuaCol@Attribute	<u>137</u> , 151
\LuaCol@directlua	71, 87, 91, 93, 119, 147, 152, 159
\LuaCol@org@pdfxform	186, 200
\LuaCol@pdfxform	195, 198
\LuaCol@setattribute	143, 145, 151
\luacolorProcessBox	2, <u>158</u> , 165, 199
\luatexluaescapestring	153
\luatexversion	71, 88, 170, 179
M	
\makeatletter	521
\MessageBreak	102, 103, 125
N	
\NeedsTeXFormat	52, 504, 515
\newattribute	138
\newluatexattribute	140
\next	430, 432, 434
\number	148, 160, 465
P	
\PassOptionsToPackage	506
\pdfoutput	507
\pdfxform	184, 186, 193
\process()	<u>394</u>
\protected	150, 189, <u>191</u>
\ProvidesPackage	53

R		<code>\special</code> 126, 132
<code>\RangeCatcodeCheck</code>		T
. 457, 485, 486, 487, 488, 489,		<code>\Test</code> 476, 499
490, 491, 492, 493, 494, 495, 496		<code>\the</code> 10, 11, 12,
<code>\RangeCatcodeInvalid</code>		13, 14, 15, 16, 17, 30, 443, 463, 464
..... 449, 477, 478, 479, 480		<code>\TMP@EnsureCode</code> 27,
<code>\repeat</code> 424, 436, 447, 455, 470		34, 35, 36, 37, 38, 39, 40, 41,
<code>\RequirePackage</code>		42, 43, 44, 45, 46, 47, 48, 49, 50
..... 55, 56, 57, 58, 59, 63, 163		<code>\traverse()</code> 329
<code>\reserved@a</code> 118, 123, 132		U
<code>\reset@color</code> 82, 112, 157		<code>\usepackage</code> 402, 510, 517, 518
<code>\RestoreCatcodes</code> .. 438, 441, 442, 497		X
S		<code>\x</code> 8, 20, 91, 99, 103
<code>\set@color</code> 81, 115, 150, 167		Y
<code>\set@page@color</code> 83		<code>\y</code> 92, 99, 104
<code>\setattribute</code> 143		Z
<code>\setattribute()</code> 280		<code>\z@</code> 113
<code>\setbox</code> 113		
<code>\setluatexattribute</code> 145		
<code>\space</code> 184, 189, 463, 464, 472		