

The luacolor package

Heiko Oberdiek
<heiko.oberdiek at gmail.com>

2011/04/23 v1.6

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	2
2.1	Catcodes and identification	2
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	4
2.6	Attribute setting	5
2.7	Whatsit insertion	5
2.8	\pdfxform support	6
2.9	Lua module	6
2.9.1	Driver detection	7
2.9.2	Color strings	7
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	12
4.3	Package installation	13
4.4	Refresh file name databases	13
4.5	Some details for the interested	13
5	History	14
	[2007/12/12 v1.0]	14
	[2009/04/10 v1.1]	14
	[2010/03/09 v1.2]	14
	[2010/12/13 v1.3]	14
	[2011/03/29 v1.4]	14
	[2011/04/22 v1.5]	14
	[2011/04/23 v1.6]	14

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 [2011/04/23 v1.6 Coloring based on LuaTeX's node 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("2011/04/23 v1.6")}}%
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 \endpackage

```

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("2011/04/23 v1.6")
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 == 3 then -- special
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       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   local n = map[color]
269   if not n then
270     n = map.n + 1
271     map.n = n
272     map[n] = color
273     map[color] = n
274   end
275   tex.write("'" .. n)
276 end

```

2.9.3 Attribute register

```

setattribute()

277 local attribute
278 function setattribute(attr)
279   attribute = attr
280 end

```

2.9.4 Whatsit insertion

```

281 local LIST = 1
282 local LIST_LEADERS = 2
283 local COLOR = 3
284 local RULE = node.id("rule")
285 local node_types = {
286   [node.id("hlist")] = LIST,
287   [node.id("vlist")] = LIST,
288   [node.id("rule")] = COLOR,
289   [node.id("glyph")] = COLOR,
290   [node.id("disc")] = COLOR,
291   [node.id("whatsit")] = {
292     [3] = COLOR, -- special
293     [8] = COLOR, -- pdf_literal
294     [14] = COLOR, -- pdf_refximage
295   },
296   [node.id("glue")] =
297     function(n)
298       if n.subtype >= 100 then -- leaders
299         if n.leader.id == RULE then
300           return COLOR
301         else
302           return LIST_LEADERS
303         end
304       end
305     end,
306 }

get_type()

307 local function get_type(n)
308   local ret = node_types[n.id]
309   if type(ret) == 'table' then
310     ret = ret[n.subtype]
311   end
312   if type(ret) == 'function' then
313     ret = ret(n)
314   end
315   return ret
316 end

```



```

317 local mode = 2 -- luatex.pdfliteral.direct
318 local WHATSIT = node.id("whatsit")
319 local SPECIAL = 3
320 local PDFLITERAL = 8
321 local DRY_FALSE = false
322 local DRY_TRUE = true

traverse()

323 local function traverse(list, color, dry)
324   if not list then
325     return color
326   end
327   if get_type(list) ~= LIST then
328     texio.write_nl("!!! Error: Wrong list type: " .. node.type(list.id))
329     return color
330   end
331   (debug)texio.write_nl("traverse: " .. node.type(list.id))
332   (!pre065) local head = list.head
333   (pre065) local head = list.list
334   for n in node.traverse(head) do
335     (debug)texio.write_nl(" node: " .. node.type(n.id))
336     local t = get_type(n)
337     if t == LIST then
338       color = traverse(n, color, dry)
339     elseif t == LIST_LEADERS then
340       local color_after = traverse(n.leader, color, DRY_TRUE)
341       if color == color_after then
342         traverse(n.leader, color, DRY_FALSE or dry)
343       else
344         traverse(n.leader, '', DRY_FALSE or dry)
345       % The color status is unknown here, because the leader box
346       % will or will not be set.
347       color = ''
348     end
349     elseif t == COLOR then
350       local v = node.has_attribute(n, attribute)
351       if v then
352         local newColor = map[v]
353         if newColor ~= color then
354           color = newColor
355           if dry == DRY_FALSE then
356             local newNode
357             if ifpdf then
358               newNode = node.new(WHATSIT, PDFLITERAL)
359               newNode.mode = mode
360               newNode.data = color
361             else
362               newNode = node.new(WHATSIT, SPECIAL)
363               newNode.data = prefix .. color
364             end
365             (*!pre065)
366             head = node.insert_before(head, n, newNode)
367             (/!pre065)
368             (*pre065)
369             if head == n then
370               newNode.next = head
371               local old_prev = head.prev
372               head.prev = newNode
373               head = newNode
374               head.prev = old_prev
375             else
376               head = node.insert_before(head, n, newNode)
377             end

```

```

378 </pre065>
379         end
380     end
381 end
382 end
383 end
384 <!pre065> list.head = head
385 <pre065> list.list = head
386 return color
387 end

process()

388 function process(box)
389     local color = ""
390     local list = tex.getbox(box)
391     traverse(list, color, DRY_FALSE)
392 end

393 </lua>

```

3 Test

```

394 <*test1>
395 \documentclass{article}
396 \usepackage{color}
397 </test1>

```

3.1 Catcode checks for loading

```

398 <*test1>

399 \catcode'\{=1 %
400 \catcode'\}=2 %
401 \catcode'\#=6 %
402 \catcode'\@=11 %
403 \expandafter\ifx\csname count@\endcsname\relax
404     \countdef\count@=255 %
405 \fi
406 \expandafter\ifx\csname @gobble\endcsname\relax
407     \long\def@gobble#1{}%
408 \fi
409 \expandafter\ifx\csname @firstofone\endcsname\relax
410     \long\def@firstofone#1{#1}%
411 \fi
412 \expandafter\ifx\csname loop\endcsname\relax
413     \expandafter\@firstofone
414 \else
415     \expandafter\@gobble
416 \fi
417 {%
418     \def\loop#1\repeat{%
419         \def\body{#1}%
420         \iterate
421     }%
422     \def\iterate{%
423         \body
424         \let\next\iterate
425     \else
426         \let\next\relax
427     \fi
428     \next
429 }%
430 \let\repeat=\fi

```

```

431 }%
432 \def\RestoreCatcodes{}
433 \count@=0 %
434 \loop
435   \edef\RestoreCatcodes{%
436     \RestoreCatcodes
437     \catcode\the\count@=\the\catcode\count@\relax
438   }%
439 \ifnum\count@<255 %
440   \advance\count@ 1 %
441 \repeat
442
443 \def\RangeCatcodeInvalid#1#2{%
444   \count@=#1\relax
445   \loop
446     \catcode\count@=15 %
447     \ifnum\count@<#2\relax
448       \advance\count@ 1 %
449     \repeat
450 }
451 \def\RangeCatcodeCheck#1#2#3{%
452   \count@=#1\relax
453   \loop
454     \ifnum#3=\catcode\count@
455     \else
456       \errmessage{%
457         Character \the\count@\space
458         with wrong catcode \the\catcode\count@\space
459         instead of \number#3%
460       }%
461     \fi
462     \ifnum\count@<#2\relax
463       \advance\count@ 1 %
464     \repeat
465 }
466 \def\space{ }
467 \expandafter\ifx\csname LoadCommand\endcsname\relax
468   \def\LoadCommand{\input luacolor.sty\relax}%
469 \fi
470 \def\Test{%
471   \RangeCatcodeInvalid{0}{47}%
472   \RangeCatcodeInvalid{58}{64}%
473   \RangeCatcodeInvalid{91}{96}%
474   \RangeCatcodeInvalid{123}{255}%
475   \catcode'\@=12 %
476   \catcode'\=0 %
477   \catcode'\%=14 %
478   \LoadCommand
479   \RangeCatcodeCheck{0}{36}{15}%
480   \RangeCatcodeCheck{37}{37}{14}%
481   \RangeCatcodeCheck{38}{47}{15}%
482   \RangeCatcodeCheck{48}{57}{12}%
483   \RangeCatcodeCheck{58}{63}{15}%
484   \RangeCatcodeCheck{64}{64}{12}%
485   \RangeCatcodeCheck{65}{90}{11}%
486   \RangeCatcodeCheck{91}{91}{15}%
487   \RangeCatcodeCheck{92}{92}{0}%
488   \RangeCatcodeCheck{93}{96}{15}%
489   \RangeCatcodeCheck{97}{122}{11}%
490   \RangeCatcodeCheck{123}{255}{15}%
491   \RestoreCatcodes
492 }

```

```

493 \Test
494 \csname @@end\endcsname
495 \end
496 </test1>

3.2 Driver detection

497 (*test2)
498 \NeedsTeXFormat{LaTeX2e}
499 \ifcsname driver\endcsname
500 \expandafter\PassOptionsToPackage\expandafter{\driver}{color}%
501 \pdfoutput=0 %
502 \fi
503 \documentclass{minimal}
504 \usepackage{luacolor}[2011/04/23]
505 \csname @@end\endcsname
506 \end
507 </test2>

508 (*test3)
509 \NeedsTeXFormat{LaTeX2e}
510 \documentclass{minimal}
511 \usepackage{luacolor}[2011/04/23]
512 \usepackage{qstest}
513 \IncludeTests{*}
514 \LogTests{log}{*}{*}
515 \makeatletter
516 \@@end
517 </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.

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

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

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

4.2 Bundle installation

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

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

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

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

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

4.3 Package installation

Unpacking. The `.dtx` file is a self-extracting `docstrip` archive. The files are extracted by running the `.dtx` through plain $\mathrm{T}_{\mathrm{E}}\mathrm{X}$:

```
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>	\rightarrow <code>tex/latex/oberdiek/luacolor.sty</code>
<code>oberdiek.luacolor.lua</code>	\rightarrow <code>scripts/oberdiek/oberdiek.luacolor.lua</code>
<code>luacolor.lua</code>	\rightarrow <code>scripts/oberdiek/luacolor.lua</code>
<code>oberdiek.luacolor-pre065.lua</code>	\rightarrow <code>scripts/oberdiek/oberdiek.luacolor-pre065.lua</code>
<code>luacolor-pre065.lua</code>	\rightarrow <code>scripts/oberdiek/luacolor-pre065.lua</code>
<code>luacolor.pdf</code>	\rightarrow <code>doc/latex/oberdiek/luacolor.pdf</code>
<code>test/luacolor-test1.tex</code>	\rightarrow <code>doc/latex/oberdiek/test/luacolor-test1.tex</code>
<code>test/luacolor-test2.tex</code>	\rightarrow <code>doc/latex/oberdiek/test/luacolor-test2.tex</code>
<code>test/luacolor-test3.tex</code>	\rightarrow <code>doc/latex/oberdiek/test/luacolor-test3.tex</code>
<code>luacolor.dtx</code>	\rightarrow <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 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ distribution (`te $\mathrm{T}_{\mathrm{E}}\mathrm{X}$` , `mik $\mathrm{T}_{\mathrm{E}}\mathrm{X}$` , ...) relies on file name databases, you must refresh these. For example, `te $\mathrm{T}_{\mathrm{E}}\mathrm{X}$` 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 luacolor.pdf unpack_files output .
```

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

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

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

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

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

5 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.

6 Index

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

Symbols

`\#` 401

<code>\%</code>	477	<code>\ifluatex</code>	60
<code>\@</code>	402, 475	<code>\ifnum</code> 71, 88, 170, 179, 439, 447, 454, 462	
<code>\@@end</code>	516	<code>\ifpdf</code>	108, 168
<code>\@PackageError</code>	66, 101, 124	<code>\ifx</code> 99, 123, 188, 403, 406, 409, 412, 467	
<code>\@PackageInfoNoLine</code>	131	<code>\IncludeTests</code>	513
<code>\@PackageWarning</code>	184, 189	<code>\info()</code>	236
<code>\@PackageWarningNoLine</code>	78	<code>\input</code>	468
<code>\@ehc</code>	68, 105, 127	<code>\iterate</code>	420, 422, 424
<code>\@empty</code>	123		
<code>\@firstofone</code>	410, 413	L	
<code>\@gobble</code>	407, 415	<code>\LoadCommand</code>	468, 478
<code>\@</code>	258, 262, 476	<code>\LogTests</code>	514
<code>\{</code>	399	<code>\loop</code>	418, 434, 445, 453
<code>\}</code>	400	<code>\ltx@ifpackageloaded</code>	61
		<code>\ltx@ifundefined</code> 137, 142, 169, 178, 183	
A		<code>\LuaCol@AtEnd</code> 28, 29, 51, 69, 85, 129, 205	
<code>\advance</code>	440, 448, 463	<code>\LuaCol@Attribute</code>	137, 151
<code>\afterassignment</code>	195	<code>\LuaCol@directlua</code>	
<code>\allocationnumber</code>	148	. 71, 87, 91, 93, 119, 147, 152, 159	
<code>\AtBeginShipout</code>	164	<code>\LuaCol@org@pdfxform</code>	186, 200
<code>\AtBeginShipoutBox</code>	165	<code>\LuaCol@pdfxform</code>	195, 198
		<code>\LuaCol@setattribute</code> ..	143, 145, 151
B		<code>\luacolorProcessBox</code> ..	2, 158, 165, 199
<code>\body</code>	419, 423	<code>\luatexluaescapestring</code>	153
		<code>\luatexversion</code>	71, 88, 170, 179
C			
<code>\catcode</code> 2, 3, 5, 6, 7, 11, 12, 13, 14,		M	
15, 16, 17, 20, 21, 23, 24, 25,		<code>\makeatletter</code>	515
26, 30, 32, 399, 400, 401, 402,		<code>\MessageBreak</code>	102, 103, 125
437, 446, 454, 458, 475, 476, 477			
<code>\count@</code> ... 196, 199, 200, 404, 433,		N	
437, 439, 440, 444, 446, 447,		<code>\NeedsTeXFormat</code>	52, 498, 509
448, 452, 454, 457, 458, 462, 463		<code>\newattribute</code>	138
<code>\countdef</code>	404	<code>\newluatexattribute</code>	140
<code>\csname</code>	9, 188,	<code>\next</code>	424, 426, 428
403, 406, 409, 412, 467, 494, 505		<code>\number</code>	148, 160, 459
<code>\current@color</code>	111, 153		
D		P	
<code>\define@color</code>	84	<code>\PassOptionsToPackage</code>	500
<code>\directlua</code>	72, 74, 171, 180	<code>\pdfoutput</code>	501
<code>\documentclass</code>	395, 503, 510	<code>\pdfxform</code>	184, 186, 193
<code>\driver</code>	500	<code>\process()</code>	388
<code>\dvidetect()</code>	244	<code>\protected</code>	150, 189, 191
		<code>\ProvidesPackage</code>	53
E			
<code>\end</code>	495, 506	R	
<code>\endcsname</code>	9, 188, 403,	<code>\RangeCatcodeCheck</code>	
406, 409, 412, 467, 494, 499, 505		. 451, 479, 480, 481, 482, 483,	
<code>\endinput</code>	51	484, 485, 486, 487, 488, 489, 490	
<code>\endlinechar</code>	4, 10, 22	<code>\RangeCatcodeInvalid</code>	
<code>\errmessage</code>	456	. 443, 471, 472, 473, 474	
		<code>\repeat</code>	418, 430, 441, 449, 464
G		<code>\RequirePackage</code>	
<code>\get()</code>	267	. 55, 56, 57, 58, 59, 63, 163	
<code>\get_type()</code>	307	<code>\reserved@a</code>	118, 123, 132
<code>\getversion()</code>	209	<code>\reset@color</code>	82, 112, 157
		<code>\RestoreCatcodes</code> ..	432, 435, 436, 491
H			
<code>\hbox</code>	113	S	
		<code>\set@color</code>	81, 115, 150, 167
I		<code>\set@page@color</code>	83
<code>\ifcolors@</code>	76	<code>\setattribute</code>	143
<code>\ifcsname</code>	499	<code>\setattribute()</code>	277
		<code>\setbox</code>	113
		<code>\setluatexattribute</code>	145

\space	184, 189, 457, 458, 466	U
\special	126, 132	\usepackage 396, 504, 511, 512
T		X
\Test	470, 493	\x 8, 20, 91, 99, 103
\the	10, 11, 12,	Y
	13, 14, 15, 16, 17, 30, 437, 457, 458	
\TMP@EnsureCode	27,	\y 92, 99, 104
	34, 35, 36, 37, 38, 39, 40, 41,	Z
	42, 43, 44, 45, 46, 47, 48, 49, 50	
\traverse()	<u>323</u>	\z@ 113