

The luacolor package

Heiko Oberdiek
<heiko.oberdiek at gmail.com>

2010/03/09 v1.2

Abstract

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

Contents

1	Documentation	2
1.1	Introduction	2
1.2	Usage	2
2	Implementation	2
2.1	Catcodes and identification	2
2.2	Check for LuaTeX	3
2.3	Check for disabled colors	3
2.4	Find driver	3
2.5	Attribute setting	4
2.6	Whatsit insertion	4
2.7	Lua module	5
2.7.1	Driver detection	5
2.7.2	Color strings	6
2.7.3	Attribute register	6
2.7.4	Whatsit insertion	6
3	Test	7
3.1	Catcode checks for loading	7
3.2	Driver detection	9
3.3	Short test for plain-TeX	9
4	Installation	9
4.1	Download	9
4.2	Bundle installation	10
4.3	Package installation	10
4.4	Refresh file name databases	10
4.5	Some details for the interested	10
5	History	11
	[2007/12/12 v1.0]	11
	[2009/04/10 v1.1]	11
	[2010/03/09 v1.2]	11
6	Index	11

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.

2 Implementation

```
1 <*package>
```

2.1 Catcodes and identification

```
2 \begingroup
3   \catcode123 1 % {
4   \catcode125 2 % }
5   \def\x{\endgroup
6     \expandafter\edef\csname LuaCol@AtEnd\endcsname{%
7       \catcode35 \the\catcode35\relax
8       \catcode64 \the\catcode64\relax
9       \catcode123 \the\catcode123\relax
10      \catcode125 \the\catcode125\relax
11    }%
12  }%
13 \x
14 \catcode35 6 % #
15 \catcode64 11 % @
16 \catcode123 1 % {
17 \catcode125 2 % }
18 \def\TMP@EnsureCode#1#2{%
19   \edef\LuaCol@AtEnd{%
20     \LuaCol@AtEnd
```

```

21     \catcode#1 \the\catcode#1\relax
22   }%
23   \catcode#1 #2\relax
24 }
25 \TMP@EnsureCode{34}{12}% "
26 \TMP@EnsureCode{39}{12}% '
27 \TMP@EnsureCode{40}{12}% (
28 \TMP@EnsureCode{41}{12}% )
29 \TMP@EnsureCode{42}{12}% *
30 \TMP@EnsureCode{43}{12}% +
31 \TMP@EnsureCode{44}{12}% ,
32 \TMP@EnsureCode{45}{12}% -
33 \TMP@EnsureCode{46}{12}% .
34 \TMP@EnsureCode{47}{12}% /
35 \TMP@EnsureCode{58}{12}% :
36 \TMP@EnsureCode{60}{12}% <
37 \TMP@EnsureCode{61}{12}% =
38 \TMP@EnsureCode{62}{12}% >
39 \TMP@EnsureCode{95}{12}% _ (other!)
40 \TMP@EnsureCode{96}{12}% `
41 \edef\LuaCol@AtEnd{%
42   \LuaCol@AtEnd
43   \noexpand\endinput
44 }

Package identification.
45 \NeedsTeXFormat{LaTeX2e}
46 \ProvidesPackage{luacolor}%
47 [2010/03/09 v1.2 Coloring based on LuaTeX's node attributes (H0)]

```

2.2 Check for LuaTeX

Without LuaTeX there is no point in using this package.

```

48 \RequirePackage{infwarerr}[2007/09/09]%
49 \RequirePackage{ifluatex}[2009/04/10]%
50 \RequirePackage{color}

51 \ifluatex
52   \RequirePackage{luatex}[2010/03/09]%
53 \else
54   \@PackageError{luacolor}{%
55     This package may only be run using LuaTeX%
56   }\@ehc
57   \expandafter\LuaCol@AtEnd
58 \fi

```

`\LuaCol@directlua`

```

59 \ifnum\luatexversion<36 %
60   \def\LuaCol@directlua{\directlua0 }%
61 \else
62   \let\LuaCol@directlua\directlua
63 \fi

```

2.3 Check for disabled colors

```

64 \ifcolors@
65 \else
66   \@PackageWarningNoLine{luacolor}{%
67     Colors are disabled by option 'monochrome'%
68   }%
69   \expandafter\LuaCol@AtEnd
70 \fi

```

2.4 Find driver

```

71 \LuaCol@directlua{%
72   require("oberdiek.luacolor")%
73 }
74 \RequirePackage{ifpdf}[2007/09/09]
75 \ifpdf
76 \else
77   \begingroup
78     \def\current@color{%
79     \def\reset@color{%
80     \setbox\z@=\hbox{%
81       \begingroup
82         \set@color
83       \endgroup
84     }%
85     \edef\reserved@a{%
86       \LuaCol@directlua{%
87         oberdiek.luacolor.dvidetect()%
88       }%
89     }%
90     \ifx\reserved@a\@empty
91       \@PackageError{luacolor}{%
92         DVI driver detection failed because of\MessageBreak
93         unrecognized color \string\special
94       }{\@ehc
95     \endgroup
96     \expandafter\expandafter\expandafter\LuaCol@AtEnd
97   \else
98     \@PackageInfoNoLine{luacolor}{%
99       Type of color \string\special: \reserved@a
100     }%
101   \fi
102 \endgroup
103 \fi

```

2.5 Attribute setting

\LuaCol@Attribute

```

104 \newattribute\LuaCol@Attribute
105 \LuaCol@directlua{%
106   oberdiek.luacolor.setattribute(\number\allocationnumber)%
107 }

```

\set@color

```

108 \protected\def\set@color{%
109   \setattribute\LuaCol@Attribute{%
110     \LuaCol@directlua{%
111       oberdiek.luacolor.get("\luatexluaescapestring{\current@color}")%
112     }%
113   }%
114 }

```

\reset@color

```

115 \def\reset@color{}

```

2.6 Whatsit insertion

\luacolorProcessBox

```

116 \def\luacolorProcessBox#1{%
117   \LuaCol@directlua{%
118     oberdiek.luacolor.process(\number#1)%
119   }%
120 }

```

```

121 \RequirePackage{atbegshi}[2007/09/09]
122 \AtBeginShipout{%
123   \luacolorProcessBox\AtBeginShipoutBox
124 }

```

Set default color.

```

125 \set@color
126 \LuaCol@AtEnd
127 \</package>

```

2.7 Lua module

```

128 <*lua>

```

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

```

129 module("oberdiek.luacolor", package.seeall)

```

2.7.1 Driver detection

```

130 local ifpdf
131 if tonumber(tex.pdfoutput) > 0 then
132   ifpdf = true
133 else
134   ifpdf = false
135 end
136 local prefix
137 local prefixes = {
138   dvips    = "color ",
139   dvipdfm  = "pdf:sc ",
140   truetex  = "textcolor:",
141   pctexps  = "ps::",
142 }
143 local patterns = {
144   ["^color "]      = "dvips",
145   ["^pdf: *begincolor "] = "dvipdfm",
146   ["^pdf: *bcolor "]   = "dvipdfm",
147   ["^pdf: *bc "]       = "dvipdfm",
148   ["^pdf: *setcolor "] = "dvipdfm",
149   ["^pdf: *scolor "]   = "dvipdfm",
150   ["^pdf: *sc "]       = "dvipdfm",
151   ["^textcolor:"]      = "truetex",
152   ["^ps::"]            = "pctexps",
153 }
154 local function info(msg, term)
155   local target = "log"
156   if term then
157     target = "term and log"
158   end
159   texio.write_nl(target, "Package luacolor info: " .. msg .. ".")
160   texio.write_nl(target, "")
161 end
162 function dvidetect()
163   local v = tex.box[0]
164   assert(v.id == node.id("hlist"))
165   for v in node.traverse_id(node.id("whatsit"), v.list) do
166     if v and v.subtype == 3 then -- special
167       local data = v.data
168       for pattern, driver in pairs(patterns) do
169         if string.find(data, pattern) then
170           prefix = prefixes[driver]
171           tex.write(driver)
172           return

```

```

173         end
174     end
175     info("\\special{" .. data .. "}", true)
176     return
177 end
178 end
179 info("Missing \\special", true)
180 end

```

2.7.2 Color strings

```

181 local map = {
182     n = 0,
183 }
184 function get(color)
185     local n = map[color]
186     if not n then
187         n = map.n + 1
188         map.n = n
189         map[n] = color
190         map[color] = n
191     end
192     tex.write("'" .. n)
193 end

```

2.7.3 Attribute register

```

194 local attribute
195 function setattribute(attr)
196     attribute = attr
197 end

```

2.7.4 Whatsit insertion

```

198 function process(box)
199     local color = ""
200     local list = tex.getbox(box)
201     traverse(list, color)
202 end

203 local LIST = 1
204 local COLOR = 2
205 local type = {
206     [node.id("hlist")] = LIST,
207     [node.id("vlist")] = LIST,
208     [node.id("rule")] = COLOR,
209     [node.id("glyph")] = COLOR,
210     [node.id("disc")] = COLOR,
211 }
212 local subtype = {
213     [3] = COLOR, -- special
214     [8] = COLOR, -- pdf_literal
215 }
216 local mode = 2 -- luatex.pdf_literal.direct
217 local WHATSIT = node.id("whatsit")
218 local SPECIAL = 3
219 local PDFLITERAL = 8

220 function traverse(list, color)
221     if not list then
222         return color
223     end
224     if type[list.id] ~= LIST then
225         texio.write_nl("!!! Error: Wrong list type: " .. node.type(list.id))
226         return color
227     end
228     (debug)texio.write_nl("traverse: " .. node.type(list.id))

```

```

229 local head = list.list
230 for n in node.traverse(head) do
231 (debug)texio.write_nl(" node: " .. node.type(n.id))
232 local type = type[n.id]
233 if type == LIST then
234 color = traverse(n, color)
235 elseif type == COLOR
236 or (type == WHATSIT
237 and subtype[n.subtype]) then
238 local v = node.has_attribute(n, attribute)
239 if v then
240 local newColor = map[v]
241 if newColor ~= color then
242 color = newColor
243 local newNode
244 if ifpdf then
245 newNode = node.new(WHATSIT, PDFLITERAL)
246 newNode.mode = mode
247 newNode.data = color
248 else
249 newNode = node.new(WHATSIT, SPECIAL)
250 newNode.data = prefix .. color
251 end
252 if head == n then
253 newNode.next = head
254 local old_prev = head.prev
255 head.prev = newNode
256 head = newNode
257 head.prev = old_prev
258 else
259 head = node.insert_before(head, n, newNode)
260 end
261 end
262 end
263 end
264 end
265 list.list = head
266 return color
267 end
268 </lua>

```

3 Test

```

269 (*test1)
270 \documentclass{article}
271 \usepackage{color}
272 </test1>

```

3.1 Catcode checks for loading

```

273 (*test1)
274 \catcode'\{=1 %
275 \catcode'\}=2 %
276 \catcode'\#=6 %
277 \catcode'\@=11 %
278 \expandafter\ifx\csname count@\endcsname\relax
279 \countdef\count@=255 %
280 \fi
281 \expandafter\ifx\csname @gobble\endcsname\relax
282 \long\def\@gobble#1{}%
283 \fi
284 \expandafter\ifx\csname @firstofone\endcsname\relax
285 \long\def\@firstofone#1{#1}%

```

```

286 \fi
287 \expandafter\ifx\csname loop\endcsname\relax
288   \expandafter\@firstofone
289 \else
290   \expandafter\@gobble
291 \fi
292 {%
293   \def\loop#1\repeat{%
294     \def\body{#1}%
295     \iterate
296   }%
297   \def\iterate{%
298     \body
299     \let\next\iterate
300   \else
301     \let\next\relax
302   \fi
303   \next
304 }%
305 \let\repeat=\fi
306 }%
307 \def\RestoreCatcodes{}
308 \count@=0 %
309 \loop
310   \edef\RestoreCatcodes{%
311     \RestoreCatcodes
312     \catcode\the\count@=\the\catcode\count@\relax
313   }%
314   \ifnum\count@<255 %
315     \advance\count@ 1 %
316 \repeat
317
318 \def\RangeCatcodeInvalid#1#2{%
319   \count@=#1\relax
320   \loop
321     \catcode\count@=15 %
322   \ifnum\count@<#2\relax
323     \advance\count@ 1 %
324   \repeat
325 }
326 \expandafter\ifx\csname LoadCommand\endcsname\relax
327   \def\LoadCommand{\input luacolor.sty\relax}%
328 \fi
329 \def\Test{%
330   \RangeCatcodeInvalid{0}{47}%
331   \RangeCatcodeInvalid{58}{64}%
332   \RangeCatcodeInvalid{91}{96}%
333   \RangeCatcodeInvalid{123}{255}%
334   \catcode'\@=12 %
335   \catcode'\=0 %
336   \catcode'\{=1 %
337   \catcode'\}=2 %
338   \catcode'\#=6 %
339   \catcode'\[=12 %
340   \catcode'\]=12 %
341   \catcode'\%=14 %
342   \catcode'\ =10 %
343   \catcode13=5 %
344   \LoadCommand
345   \RestoreCatcodes
346 }
347 \Test

```



```

348 \csname @@end\endcsname
349 \end
350 </test1>

```

3.2 Driver detection

```

351 <*test2>
352 \NeedsTeXFormat{LaTeX2e}
353 \ifcsname driver\endcsname
354   \expandafter\PassOptionsToPackage\expandafter{\driver}{color}%
355   \pdfoutput=0 %
356 \fi
357 \documentclass{minimal}
358 \usepackage{luacolor}[2010/03/09]
359 \csname @@end\endcsname
360 \end
361 </test2>

362 <*test3>
363 \NeedsTeXFormat{LaTeX2e}
364 \documentclass{minimal}
365 \usepackage{luacolor}[2010/03/09]
366 \usepackage{qstest}
367 \IncludeTests{*}
368 \LogTests{log}{*}{*}
369 \makeatletter

370 \@@end
371 </test3>

```

3.3 Short test for plain-TeX

```

372 <*test4>
373 \input luacolor.sty\relax
374 \newluastate\TestLuaState
375 \newattribute\TestAttr
376 \setattribute\TestAttr{10}
377 \unsetattribute\TestAttr
378 \newcatcodetable\TestCTa
379 \begingroup
380   \SetCatcodeRange{'A'}{'Z'}{12}%
381 \endgroup
382 \BeginCatcodeRegime\__CT__LaTeX
383 \EndCatcodeRegime
384 \end
385 </test4>

```

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.

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

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-`TEX`:

```
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>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 `TEX` distribution (`teTEX`, `mikTEX`, ...) relies on file name databases, you must refresh these. For example, `teTEX` 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 \LaTeX . The `.dtx` chooses its action depending on the format:

plain-`TEX`: Run `docstrip` and extract the files.

\LaTeX : Generate the documentation.

If you insist on using \LaTeX for `docstrip` (really, `docstrip` does not need \LaTeX), 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 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.

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		<code>\AtBeginShipoutBox</code> 123
<code>\#</code>	276, 338	
<code>\%</code>	341	
<code>\@</code>	277, 334	
<code>\@@end</code>	370	
<code>\@PackageError</code>	54, 91	
<code>\@PackageInfoNoLine</code>	98	
<code>\@PackageWarningNoLine</code>	66	
<code>\@ehc</code>	56, 94	
<code>\@empty</code>	90	
<code>\@firstofone</code>	285, 288	
<code>\@gobble</code>	282, 290	
<code>\[</code>	339	
<code>\]</code>	175, 179, 335	
<code>\{</code>	274, 336	
<code>\}</code>	275, 337	
<code>\]</code>	340	
<code>_</code>	382	
		B
		<code>\BeginCatcodeRegime</code> 382
		<code>\body</code> 294, 298
		C
		<code>\catcode</code> 3, 4, 7, 8, 9, 10, 14,
		15, 16, 17, 21, 23, 274, 275, 276,
		277, 312, 321, 334, 335, 336,
		337, 338, 339, 340, 341, 342, 343
		<code>\count@</code> 279, 308,
		312, 314, 315, 319, 321, 322, 323
		<code>\countdef</code> 279
		<code>\csname</code> 6,
		278, 281, 284, 287, 326, 348, 359
		<code>\current@color</code> 78, 111
		D
		<code>\directlua</code> 60, 62
<code>_</code>	342	<code>\documentclass</code> 270, 357, 364
		<code>\driver</code> 354
	A	
<code>\advance</code>	315, 323	
<code>\allocationnumber</code>	106	
<code>\AtBeginShipout</code>	122	
		E
		<code>\end</code> 349, 360, 384
		<code>\EndCatcodeRegime</code> 383

\endcsname	6, 278, 281, 284, 287, 326, 348, 353, 359		
\endinginput	43		
H			
\hbox	80		
I			
\ifcolors@	64		
\ifcsname	353		
\ifluatex	51		
\ifnum	59, 314, 322		
\ifpdf	75		
\ifx	90, 278, 281, 284, 287, 326		
\IncludeTests	367		
\input	327, 373		
\iterate	295, 297, 299		
L			
\LoadCommand	327, 344		
\LogTests	368		
\loop	293, 309, 320		
\LuaCol@AtEnd	19, 20, 41, 42, 57, 69, 96, 126		
\LuaCol@Attribute	104, 109		
\LuaCol@directlua	59, 71, 86, 105, 110, 117		
\luacolorProcessBox	2, 116, 123		
\luatexluaescapestring	111		
\luatexversion	59		
M			
\makeatletter	369		
\MessageBreak	92		
N			
\NeedsTeXFormat	45, 352, 363		
\newattribute	104, 375		
\newcatcodetable	378		
\newluastate	374		
\next	299, 301, 303		
\number	106, 118		
P			
\PassOptionsToPackage	354		
\pdfoutput	355		
\protected	108		
\ProvidesPackage	46		
R			
\RangeCatcodeInvalid	318, 330, 331, 332, 333		
\repeat	293, 305, 316, 324		
\RequirePackage	48, 49, 50, 52, 74, 121		
\reserved@a	85, 90, 99		
\reset@color	79, 115		
\RestoreCatcodes	307, 310, 311, 345		
S			
\set@color	82, 108, 125		
\setattribute	109, 376		
\setbox	80		
\SetCatcodeRange	380		
\special	93, 99		
T			
\Test	329, 347		
\TestAttr	375, 376, 377		
\TestCTa	378		
\TestLuaState	374		
\the	7, 8, 9, 10, 21, 312		
\TMP@EnsureCode	18, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40		
U			
\unsetattribute	377		
\usepackage	271, 358, 365, 366		
X			
\x	5, 13		
Z			
\z@	80		