The \texttt{luacolor} package

Heiko Oberdiek$^{*}$

2021-02-17 v1.17

Abstract

Package \texttt{luacolor} implements color support based on \LaTeX{}'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 \LaTeX{} ....................................... 4
  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 ...................................... 5
  2.8 \texttt{\pdfxform}/\texttt{\saveboxresource} support .......... 6
  2.9 Lua module ............................................. 8
    2.9.1 Driver detection ................................... 8
    2.9.2 Color strings ...................................... 9
    2.9.3 Attribute register .................................... 9
    2.9.4 Whatsit insertion ................................... 9

3 Installation 12
  3.1 Download ............................................... 12
  3.2 Bundle installation .................................. 12
  3.3 Package installation .................................. 12
  3.4 Refresh file name databases ............................ 13
  3.5 Some details for the interested ....................... 13

4 History 13
  [2007/12/12 v1.0] ........................................ 13
  [2009/04/10 v1.1] ........................................ 13
  [2010/03/09 v1.2] ........................................ 13
  [2010/12/13 v1.3] ........................................ 13
  [2011/03/29 v1.4] ........................................ 14
  [2011/04/22 v1.5] ........................................ 14

$^{*}$Please report any issues at \url{https://github.com/ho-tex/luacolor/issues}
1 Documentation

1.1 Introduction

This package uses Lua\TeX{}'s attribute register 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 \texttt{color} is loaded automatically by this package \texttt{luacolor}. If you need a special driver option or you prefer package \texttt{xcolor}, then load it before package \texttt{luacolor}, for example:

\begin{verbatim}
\usepackage[dvipdfmx]{xcolor}
\end{verbatim}

The package \texttt{luacolor} is loaded without options:

\begin{verbatim}
\usepackage{luacolor}
\end{verbatim}

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 whatsit for color.

At last the attribute annotations of the nodes in the output box must be analyzed to insert the necessary color whatsit. Currently Lua\TeX{} lacks an appropriate callback function. Therefore package \texttt{atbegshi} is used to get control before a box is shipped out.

\begin{verbatim}
\luacolorProcessBox {⟨box⟩}
\end{verbatim}

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

2.1 Catcodes and identification

\begin{verbatim}
\begin{group}
code61\code48\code32=10\relax\%
code13=5 \^^M
\endlinechar=13 \%
code123=1 \%
\code125=2 \%
code64=11 \%
\def\x{\endgroup%
\expandafter\edef\csname LuaCol@AtEnd\endcsname{%
\endlinechar=\the\endlinechar\relax
\code13=\the\code13\relax
\code32=\the\code32\relax
\code35=\the\code35\relax
\code61=\the\code61\relax
\code64=\the\code64\relax
\code123=\the\code123\relax
\code125=\the\code125\relax
}
\catcode61\catcode48\catcode32=10\relax\%
\catcode13=5 \^^M
\endlinechar=13 \%
\catcode64=11 \%
\catcode123=1 \%
\catcode125=2 \%
\def\x{\endgroup%
\expandafter\edef\csname LuaCol0AtEnd\endcsname{%
\endlinechar=\the\endlinechar\relax
\code13=\the\code13\relax
\code32=\the\code32\relax
\code35=\the\code35\relax
\code61=\the\code61\relax
\code64=\the\code64\relax
\code123=\the\code123\relax
\code125=\the\code125\relax
}
\}
\x
\end{group}
code61\code48\code32=10\relax\%
code13=5 \^^M
\endlinechar=13 \%
code35=6 \%
code64=11 \%
code123=1 \%
code125=2 \%
\def\x{\endgroup%
\end{verbatim}
2.2 Check for LuaTEX

Without LuaTEX there is no point in using this package.

2.3 Check for disabled colors

2.4 Load module and check version
2.5 Find driver

\ifnum\outputmode=\@ne
\else
\begingroup
\def\current@color{}\%
\def\reset@color{}\%
\setbox\z@=\hbox{\begingroup\set@color\endgroup}{\directlua{oberdiek.luacolor.dvidetect()}}\%
\edef\reserved@a{\directlua{oberdiek.luacolor.dvidetect()}}\%
\ifx\reserved@a\@empty\PackageError{luacolor}{DVI driver detection failed because of \string\special\@ehc\endgroup\expandafter\expandafter\expandafter\LuaCol@AtEnd\else\PackageInfo{luacolor}{Type of color \string\special: \reserved@a\@gobble}\fi\endgroup\fi

2.6 Attribute setting

\LuaCol@Attribute
\newattribute\LuaCol@Attribute
\let\LuaCol@setattribute\setattribute\directlua{oberdiek.luacolor.setattribute(\number\allocationnumber)}
\set@color
\protected\def\set@color{%\LuaCol@setattribute\LuaCol@Attribute\directlua{oberdiek.luacolor.get("\luaescapestring{\current@color}")}}
\reset@color
\def\reset@color{}}
2.7 Whatsit insertion

\luacolorProcessBox
\def\luacolorProcessBox#1{%
  oberdiek.luacolor.process(#1)%
}%
\directlua{%
  if luatexbase.callbacktypes.pre_shipout_filter then
    token.get_next()
  end
}\@secondoftwo\@gobble{
\RequirePackage{atbegshi}[2011/01/30]
\AtBeginShipout{%
  \luacolorProcessBox\AtBeginShipoutBox
}%
\set@color

Set default color.

2.8 \pdfxform/\saveboxresource support

\ifnum\outputmode=\@one
  \let\LuaCol@org@pdfxform\saveboxresource
\fi
\def\LuaCol@iii@i@ii#1#2#3{#3{#1}{#2}}
\def\LuaCol@ii@i#1#2{{#2#1}}
\def\LuaCol@if@keyword#1#2#3{%
  \expanded{\unexpanded{\LuaCol@iii@i@ii{#2}{#3}}\expandafter}%
  \directlua{%
    \token.put_next(token.create(token.scan_keyword(token.scan_string())
      and '@firstoftwo'
    or '@secondoftwo'))
  }{#1}%
}%

The following macro scans a integer and expands to a token equivalent to a chardef
whose value corresponds to the scanned integer. This allows the integer to be
passed around as a undelimited argument.
\def\LuaCol@scan@number{%
  \directlua{%
    \token.put_next(token.new(token.scan_int(), token.command_id'char_given'))
  }%}
\def\LuaCol@scan@tobrace{%
  \directlua{%
    local relax, space = token.command_id'relax', token.command_id'spacer'
    local t
    repeat
      t = token.scan_token()
    until not (t.command == relax or t.command == space)
  }%}

\TeX{} primitives like \saveboxresource read braced arguments in a special way.
Especially they expand everything until they find a left brace. To simulate this,
we use Lua to expand everything else:
\def\LuaCol@scan@tobrace{%
  \directlua{%
    local relax, space = token.command_id'relax', token.command_id'spacer'
    local t
    repeat
      t = token.scan_token()
    until not (t.command == relax or t.command == space)
This could be written in Lua, but at least up to LuaTeX 1.11, feeding back too many tokens from Lua to TeX triggers a segmentation fault. This is written in Lua so the integer setting is expandable and does not interfere with a preceding \immediate.

Legacy alias.
2.9 Lua module

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

\let\pdfxform\saveboxresource
\fi\LuaCol@AtEnd%

2.9.1 Driver detection

local ifpdf = tonumber(tex.outputmode or tex.pdfoutput) > 0
local prefix
local prefixes = {
dvips = "color ",
dvipdfm = "pdf:sc ",
truetex = "textcolor:",
pctexps = "ps:",
}
local patterns = {
["^color "] = "dvips",
["^pdf: *begincolor "] = "dvipdfm",
["^pdf: *bcolor "] = "dvipdfm",
["^pdf: *bc "] = "dvipdfm",
["^pdf: *setcolor "] = "dvipdfm",
["^pdf: *scolor "] = "dvipdfm",
["^pdf: *sc "] = "dvipdfm",
["^textcolor:" ] = "truetex",
["^ps:"] = "pctexps",
}

info()

local function info(msg, term)
local target = "log"
if term then
  target = "term and log"
end
texio.write_nl(target, "Package luacolor info: " .. msg .. ".")
texio.write_nl(target, ")
end

dvideetect()

function luacolor.dvideetect()
local v = tex.box[0]
assert(v.id == node.id("hlist"))
for v in node.traverse_id(node.id("whatsit"), v.head) do
  if v and v.subtype == node.subtype("special") then
    local data = v.data
    for pattern, driver in pairs(patterns) do
      if pattern:match(pattern) then
        prefix = driver
        break
      end
    end
  end
end

getversion()

function luacolor.getversion()
tex.write("2021-02-17 v1.17")
end

oberdiek = oberdiek or {}
local luacolor = oberdiek.luacolor or {}
oberdiek.luacolor = luacolor

info()
if string.find(data, pattern) then
  prefix = prefixes[driver]
  tex.write(driver)
  return
end

info("\special{" .. data .. "}", true)
return
end

info("Missing \special", true)
end

2.9.2 Color strings

local map = {
  n = 0,
}

function luacolor.get(color)
  tex.write("" .. luacolor.getvalue(color))
end

function luacolor.getvalue(color)
  local n = map[color]
  if not n then
    n = map.n + 1
    map.n = n
    map[n] = color
    map[color] = n
  end
  return n
end

2.9.3 Attribute register

local attribute

function luacolor.setattribute(attr)
  attribute = attr
end

function luacolor.getattribute()
  return attribute
end

2.9.4 Whatsit insertion

local LIST = 1
local LIST_LEADERS = 2
local LIST_DISC = 3
local COLOR = 4
local NOCOLOR = 5
local RULE = node.id("rule")
local node_types = {

315  [node.id("hlist")] = LIST,
316  [node.id("vlist")] = LIST,
317  [node.id("rule")] = COLOR,
318  [node.id("glyph")] = COLOR,
319  [node.id("disc")] = LIST_DISC,
320  [node.id("whatsit")] = {
321     [node.subtype("pdf_colorstack")] =
322        function(n)
323           return n.stack == 0 and NOCOLOR or nil
324        end,
325     [node.subtype("special")] = COLOR,
326     [node.subtype("pdf_literal")] = COLOR,
327     [node.subtype("pdf_save")] = COLOR,
328     [node.subtype("pdf_restore")] = COLOR, -- probably not needed
329     -- TODO (DPC)  [node.subtype("pdf_refimage")] = COLOR,
330  },
331  [node.id("glue")] =
332     function(n)
333        if n.subtype >= 100 then -- leaders
334           if n.leader.id == RULE then
335              return COLOR
336           else
337              return LIST_LEADERS
338           end
339        end
340     end,
341 }

get_type()

342 local function get_type(n)
343    local ret = node_types[n.id]
344    if type(ret) == 'table' then
345        ret = ret[n.subtype]
346    end
347    if type(ret) == 'function' then
348        ret = ret(n)
349    end
350    return ret
351 end

352 local mode = 2 -- luatex.pdfliteral.direct
353 local WHATSIT = node.id("whatsit")
354 local SPECIAL = node.subtype("special")
355 local PDFLITERAL = node.subtype("pdf_literal")
356 local DRY_FALSE = false
357 local DRY_TRUE = true

traverse()

358 local function traverse(list, color, dry)
359    if not list then
360       return color
361    end
362    local head
363    if get_type(list) == LIST then
364       head = list.head
365    elseif get_type(list) == LIST_DISC then
366       head = list.replace
367    else
368       texio.write_nl("!!! Error: Wrong list type: " .. node.type(list.id))
return color
end

for n in node.traverse(head) do
  local t = get_type(n)
  local color_after = traverse(n.leader, color, DRY_TRUE)
  if color == color_after then
    traverse(n.leader, color, DRY_FALSE or dry)
  else
    traverse(n.leader, '', DRY_FALSE or dry)
  end
end

if get_type(list) == LIST then
  list.head = head
else
  list.replace = head
end
return color
end
end

function luacolor.process(box)
  local color ="
  local list = tex.getbox(box)
  traverse(list, color, DRY_FALSE)
end

if luatexbase.callbacktypes.pre_shipout_filter then
For recent versions of luaotfload, we can register a callback to control how coloring glyph is handled for the color feature.

```lua
if luaotfload.set_colorhandler then
    local set_attribute = node.direct.set_attribute
    luaotfload.set_colorhandler(function(head, n, color)
      set_attribute(n, attribute, luacolor.getvalue(color))
      return head, n
    end)
end
```

3 Installation

3.1 Download

Package. This package is available on CTAN:


Bundle. All the packages of the bundle ‘luacolor’ 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/luacolor.tds.zip](http://ctan.org/pkg/luacolor)

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

3.2 Bundle installation

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

```
unzip luacolor.tds.zip -d "~/texmf"
```

Script installation. Check the directory `TDS:scripts/luacolor/` for scripts that need further installation steps.

3.3 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
```

---

1[CTAN:pkg/luacolor]
TDS. Now the different files must be moved into the different directories in your installation TDS tree (also known as texmf tree):

- luacolor.sty → tex/latex/luacolor/luacolor.sty
- luacolor.lua → scripts/luacolor/luacolor.lua
- luacolor.pdf → doc/latex/luacolor/luacolor.pdf
- luacolor.dtx → source/latex/luacolor/luacolor.dtx

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.

3.4 Refresh file name databases

If your TeX distribution (TeX Live, MiKTeX, ...) relies on file name databases, you must refresh these. For example, TeX Live users run texhash or mktexlsr.

3.5 Some details for the interested

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 pdf\LaTeX:

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

4 History

[2007/12/12 v1.0]
- First public version.

[2009/04/10 v1.1]
- Fixes for changed syntax of \directlua in Lua\TeX 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.

[2016/05/13 v1.9]
- More use of node.subtype instead of magic numbers.
- luatex 85 updates
[2016/05/16 v1.10]
• Documentation updates.

[2018/11/22 v1.11]
• handle issue 43.
• removed pre-0.65 stuff

[2019/07/25 v1.12]
• removed uses of module function, see PR70

[2019/11/29 v1.13]
• Documentation updates.
• Use iftex directly.

[2020-02-22 v1.14]
• Drop use of iftex ltxcmds and infwarerr.
• Assume lualatex preloaded into format (true since 2015).
• Patch \saveboxresource rather than \pdfxform (keep old name as alias).
• Grab the number via Lua so that a \immediate prefix still works with \saveboxresource/\pdfxform.
• Added handler for the color feature of luaotfload

[2020-02-24 v1.15]
• Grab all possible arguments for \saveboxresource/\pdfxform

[2020-04-04 v1.16]
• Reset color after pdf_colorstack whatsits.

[2021-02-17 v1.17]
• Use \exp X^E_2’s new pre_shipout_filter callback if it’s available to allow coloring background and foreground layer material

5 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.

<table>
<thead>
<tr>
<th>Symbols</th>
<th>\one</th>
<th>94, 152</th>
</tr>
</thead>
<tbody>
<tr>
<td>@ehc</td>
<td>59, 91, 113</td>
<td></td>
</tr>
<tr>
<td>@empty</td>
<td>109</td>
<td>@undefined 56</td>
</tr>
<tr>
<td>@gobble</td>
<td>119, 145</td>
<td>| 279, 283</td>
</tr>
</tbody>
</table>

15