# The luacolor package

Heiko Oberdiek
<heiko.oberdiek at googlemail.com>

2011/04/22 v1.5

**Abstract**

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

## Contents

# 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 whatsits for color.

At last the attribute annotations of the nodes in the output box must be analyzed to insert the necessary color whatsits. 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 respondibility 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
```

```
    \expandafter\edef\csname LuaCol@AtEnd\endcsname{%
      \endlinechar=\the\endlinechar\relax
      \catcode13=\the\catcode13\relax
      \catcode32=\the\catcode32\relax
      \catcode35=\the\catcode35\relax
      \catcode61=\the\catcode61\relax
      \catcode64=\the\catcode64\relax
      \catcode123=\the\catcode123\relax
      \catcode125=\the\catcode125\relax
    }%
  }%
\x\catcode61\catcode48\catcode32=10\relax%
\catcode13=5 % ^^M
\endlinechar=13 %
\catcode35=6 % #
\catcode64=11 % @
\catcode123=1 % {
\catcode125=2 % }
\def\TMP@EnsureCode#1#2{%
  \edef\LuaCol@AtEnd{%
    \LuaCol@AtEnd
    \catcode#1=\the\catcode#1\relax
  }%
  \catcode#1=#2\relax
}
\TMP@EnsureCode{34}{12}% "
\TMP@EnsureCode{39}{12}% ’
\TMP@EnsureCode{40}{12}% (
\TMP@EnsureCode{41}{12}% )
\TMP@EnsureCode{42}{12}% *
\TMP@EnsureCode{43}{12}% +
\TMP@EnsureCode{44}{12}% ,
\TMP@EnsureCode{45}{12}% -
\TMP@EnsureCode{46}{12}% .
\TMP@EnsureCode{47}{12}% /
\TMP@EnsureCode{58}{12}% :
\TMP@EnsureCode{60}{12}% <
\TMP@EnsureCode{62}{12}% >
\TMP@EnsureCode{91}{12}% [
\TMP@EnsureCode{93}{12}% ]
\TMP@EnsureCode{95}{12}% _ (other!)
\TMP@EnsureCode{96}{12}% ‘
\edef\LuaCol@AtEnd{\LuaCol@AtEnd\noexpand\endinput}
```

Package identification.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{luacolor}%
  [2011/04/22 v1.5 Coloring based on LuaTeX’s node attributes (HO)]
```

## 2.2 Check for LuaTEX

Without LuaTEX there is no point in using this package.

```
\RequirePackage{infwarerr}[2010/04/08]%
\RequirePackage{ifluatex}[2010/03/01]%
\RequirePackage{ifpdf}[2011/01/30]%
\RequirePackage{ltxcmds}[2011/04/18]%
\RequirePackage{color}

\ifluatex
  \ltx@ifpackageloaded{luatexbase-attr}{%
  }{%
    \RequirePackage{luatex}[2010/03/09]%
  }%
\else
```

```
  \@PackageError{luacolor}{%
    This package may only be run using LuaTeX%
  }\@ehc
  \expandafter\LuaCol@AtEnd
\fi%
```

\LuaCol@directlua

```
\ifnum\luatexversion<36 %
  \def\LuaCol@directlua{\directlua0 }%
\else
  \let\LuaCol@directlua\directlua
\fi
```

## 2.3 Check for disabled colors

```
\ifcolors@
\else
  \@PackageWarningNoLine{luacolor}{%
    Colors are disabled by option 'monochrome'%
  }%
  \def\set@color{}%
  \def\reset@color{}%
  \def\set@page@color{}%
  \def\define@color#1#2{}%
  \expandafter\LuaCol@AtEnd
\fi%
```

## 2.4 Find driver

```
\LuaCol@directlua{%
  require("oberdiek.luacolor\ifnum\luatexversion<65 -pre065\fi")%
}
\ifpdf
\else
  \begingroup
    \def\current@color{}%
    \def\reset@color{}%
    \setbox\z@=\hbox{%
      \begingroup
        \set@color
      \endgroup
    }%
    \edef\reserved@a{%
      \LuaCol@directlua{%
        oberdiek.luacolor.dvidetect()%
      }%
    }%
    \ifx\reserved@a\@empty
      \@PackageError{luacolor}{%
        DVI driver detection failed because of\MessageBreak
        unrecognized color \string\special
      }\@ehc
      \endgroup
      \expandafter\expandafter\expandafter\LuaCol@AtEnd
    \else
      \@PackageInfoNoLine{luacolor}{%
        Type of color \string\special: \reserved@a
      }%
    \fi%
  \endgroup
\fi
```

## 2.5 Attribute setting

\LuaCol@Attribute

```
\ltx@IfUndefined{newluatexattribute}{%
  \newattribute\LuaCol@Attribute
}{%
  \newluatexattribute\LuaCol@Attribute
}
\ltx@IfUndefined{setluatexattribute}{%
  \let\LuaCol@setattribute\setattribute
}{%
  \let\LuaCol@setattribute\setluatexattribute
}
\LuaCol@directlua{%
  oberdiek.luacolor.setattribute(\number\allocationnumber)%
}
```

\set@color

```
\protected\def\set@color{%
  \LuaCol@setattribute\LuaCol@Attribute{%
    \LuaCol@directlua{%
      oberdiek.luacolor.get("\luatexluaescapestring{\current@color}")%
    }%
  }%
}
```

\reset@color

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

## 2.6 Whatsit insertion

\luacolorProcessBox

```
\def\luacolorProcessBox#1{%
  \LuaCol@directlua{%
    oberdiek.luacolor.process(\number#1)%
  }%
}
```

```
\RequirePackage{atbegshi}[2011/01/30]
\AtBeginShipout{%
  \luacolorProcessBox\AtBeginShipoutBox
}
```

Set default color.

```
\set@color
```

## 2.7 \pdfxform support

```
\ifpdf
  \ltx@IfUndefined{pdfxform}{%
    \ifnum\luatexversion>36 %
      \directlua{%
        tex.enableprimitives('',{%
          'pdfxform','pdflastxform','pdfrefxform'%
        })%
      }%
    \fi
  }{}%
  \ltx@IfUndefined{protected}{%
    \ifnum\luatexversion>36 %
      \directlua{tex.enableprimitives('',{'protected'})}%
    \fi
```

```
  }{}%
  \ltx@IfUndefined{pdfxform}{%
    \@PackageWarning{luacolor}{\string\pdfxform\space not found}%
  }{%
    \let\LuaCol@org@pdfxform\pdfxform
    \begingroup\expandafter\expandafter\expandafter\endgroup
    \expandafter\ifx\csname protected\endcsname\relax
      \@PackageWarning{luacolor}{\string\protected\space not found}%
    \else
      \expandafter\protected
    \fi
    \def\pdfxform{%
      \begingroup
      \afterassignment\LuaCol@pdfxform
      \count@=%
    }%
    \def\LuaCol@pdfxform{%
      \luacolorProcessBox\count@
      \LuaCol@org@pdfxform\count@
      \endgroup
    }%
  }%
\fi
\LuaCol@AtEnd%
```
⟨/package⟩

## 2.8 Lua module

⟨*lua⟩

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

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

### 2.8.1 Driver detection

```
local ifpdf
if tonumber(tex.pdfoutput) > 0 then
  ifpdf = true
else
  ifpdf = false
end
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",
}
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
function dvidetect()
  local v = tex.box[0]
  assert(v.id == node.id("hlist"))
⟨!pre065⟩  for v in node.traverse_id(node.id("whatsit"), v.head) do
⟨pre065⟩  for v in node.traverse_id(node.id("whatsit"), v.list) do
    if v and v.subtype == 3 then -- special
      local data = v.data
      for pattern, driver in pairs(patterns) do
        if string.find(data, pattern) then
          prefix = prefixes[driver]
          tex.write(driver)
          return
        end
      end
      info("\\special{" .. data .. "}", true)
      return
    end
  end
  info("Missing \\special", true)
end
```

### 2.8.2 Color strings

```
local map = {
  n = 0,
}
function get(color)
  local n = map[color]
  if not n then
    n = map.n + 1
    map.n = n
    map[n] = color
    map[color] = n
  end
  tex.write("" .. n)
end
```

### 2.8.3 Attribute register

```
local attribute
function setattribute(attr)
  attribute = attr
end
```

### 2.8.4 Whatsit insertion

```
local LIST = 1
local LIST_LEADERS = 2
local COLOR = 3
local RULE = node.id("rule")
local node_types = {
  [node.id("hlist")] = LIST,
  [node.id("vlist")] = LIST,
  [node.id("rule")]  = COLOR,
  [node.id("glyph")] = COLOR,
  [node.id("disc")]  = COLOR,
  [node.id("whatsit")] = {
    [3] = COLOR, -- special
    [8] = COLOR, -- pdf_literal
    [14] = COLOR, -- pdf_refximage
  },
```

```
  [node.id("glue")] =
    function(n)
      if n.subtype >= 100 then -- leaders
        if n.leader.id == RULE then
          return COLOR
        else
          return LIST_LEADERS
        end
      end
    end,
}
local function get_type(n)
  local ret = node_types[n.id]
  if type(ret) == 'table' then
    ret = ret[n.subtype]
  end
  if type(ret) == 'function' then
    ret = ret(n)
  end
  return ret
end
local mode = 2 -- luatex.pdfliteral.direct
local WHATSIT = node.id("whatsit")
local SPECIAL = 3
local PDFLITERAL = 8
local DRY_FALSE = 0
local DRY_TRUE = 1
local function traverse(list, color, dry)
  if not list then
    return color
  end
  if get_type(list) ~= LIST then
    texio.write_nl("!!! Error: Wrong list type: " .. node.type(list.id))
    return color
  end
⟨debug⟩texio.write_nl("traverse: " .. node.type(list.id))
⟨!pre065⟩  local head = list.head
⟨pre065⟩  local head = list.list
  for n in node.traverse(head) do
⟨debug⟩texio.write_nl("  node: " .. node.type(n.id))
    local t = get_type(n)
    if t == LIST then
      color = traverse(n, color, DRY_FALSE)
    elseif t == LIST_LEADERS then
      local color_after = traverse(n.leader, color, DRY_TRUE)
      if color == color_after then
        traverse(n.leader, color, DRY_FALSE)
      else
        traverse(n.leader, '', DRY_FALSE)
      end
    elseif t == COLOR then
      local v = node.has_attribute(n, attribute)
      if v then
        local newColor = map[v]
        if newColor ~= color then
          color = newColor
          if dry == DRY_FALSE then
            local newNode
            if ifpdf then
              newNode = node.new(WHATSIT, PDFLITERAL)
              newNode.mode = mode
              newNode.data = color
```

```
            else
              newNode = node.new(WHATSIT, SPECIAL)
              newNode.data = prefix .. color
            end
⟨*!pre065⟩
            head = node.insert_before(head, n, newNode)
⟨/!pre065⟩
⟨*pre065⟩
            if head == n then
              newNode.next = head
              local old_prev = head.prev
              head.prev = newNode
              head = newNode
              head.prev = old_prev
            else
              head = node.insert_before(head, n, newNode)
            end
⟨/pre065⟩
          end
        end
      end
    end
  end
⟨!pre065⟩  list.head = head
⟨pre065⟩  list.list = head
  return color
end

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

⟨/lua⟩
```

# 3 Test

```
⟨*test1⟩
\documentclass{article}
\usepackage{color}
⟨/test1⟩
```

## 3.1 Catcode checks for loading

```
⟨*test1⟩

\catcode`\{=1 %
\catcode`\}=2 %
\catcode`\#=6 %
\catcode`\@=11 %
\expandafter\ifx\csname count@\endcsname\relax
  \countdef\count@=255 %
\fi
\expandafter\ifx\csname @gobble\endcsname\relax
  \long\def\@gobble#1{}%
\fi
\expandafter\ifx\csname @firstofone\endcsname\relax
  \long\def\@firstofone#1{#1}%
\fi
\expandafter\ifx\csname loop\endcsname\relax
  \expandafter\@firstofone
\else
  \expandafter\@gobble
\fi
```

```
{%
  \def\loop#1\repeat{%
    \def\body{#1}%
    \iterate
  }%
  \def\iterate{%
    \body
      \let\next\iterate
    \else
      \let\next\relax
    \fi
    \next
  }%
  \let\repeat=\fi
}%
\def\RestoreCatcodes{}
\count@=0 %
\loop
  \edef\RestoreCatcodes{%
    \RestoreCatcodes
    \catcode\the\count@=\the\catcode\count@\relax
  }%
\ifnum\count@<255 %
  \advance\count@ 1 %
\repeat

\def\RangeCatcodeInvalid#1#2{%
  \count@=#1\relax
  \loop
    \catcode\count@=15 %
  \ifnum\count@<#2\relax
    \advance\count@ 1 %
  \repeat
}
\def\RangeCatcodeCheck#1#2#3{%
  \count@=#1\relax
  \loop
    \ifnum#3=\catcode\count@
    \else
      \errmessage{%
        Character \the\count@\space
        with wrong catcode \the\catcode\count@\space
        instead of \number#3%
      }%
    \fi
  \ifnum\count@<#2\relax
    \advance\count@ 1 %
  \repeat
}
\def\space{ }
\expandafter\ifx\csname LoadCommand\endcsname\relax
  \def\LoadCommand{\input luacolor.sty\relax}%
\fi
\def\Test{%
  \RangeCatcodeInvalid{0}{47}%
  \RangeCatcodeInvalid{58}{64}%
  \RangeCatcodeInvalid{91}{96}%
  \RangeCatcodeInvalid{123}{255}%
  \catcode`\@=12 %
  \catcode`\\=0 %
  \catcode`\%=14 %
  \LoadCommand
```

```
  \RangeCatcodeCheck{0}{36}{15}%
  \RangeCatcodeCheck{37}{37}{14}%
  \RangeCatcodeCheck{38}{47}{15}%
  \RangeCatcodeCheck{48}{57}{12}%
  \RangeCatcodeCheck{58}{63}{15}%
  \RangeCatcodeCheck{64}{64}{12}%
  \RangeCatcodeCheck{65}{90}{11}%
  \RangeCatcodeCheck{91}{91}{15}%
  \RangeCatcodeCheck{92}{92}{0}%
  \RangeCatcodeCheck{93}{96}{15}%
  \RangeCatcodeCheck{97}{122}{11}%
  \RangeCatcodeCheck{123}{255}{15}%
  \RestoreCatcodes
}
\Test
\csname @@end\endcsname
\end
```
⟨/test1⟩

## 3.2 Driver detection

```
⟨*test2⟩
\NeedsTeXFormat{LaTeX2e}
\ifcsname driver\endcsname
  \expandafter\PassOptionsToPackage\expandafter{\driver}{color}%
  \pdfoutput=0 %
\fi
\documentclass{minimal}
\usepackage{luacolor}[2011/04/22]
\csname @@end\endcsname
\end
⟨/test2⟩
```

```
⟨*test3⟩
\NeedsTeXFormat{LaTeX2e}
```

```
\documentclass{minimal}
\usepackage{luacolor}[2011/04/22]
\usepackage{qstest}
\IncludeTests{*}
\LogTests{log}{*}{*}
\makeatletter
```

```
\@@end
```
⟨/test3⟩

# 4 Installation

## 4.1 Download

**Package.** This package is available on CTAN[1]:

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/

## 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):

```
luacolor.sty                → tex/latex/oberdiek/luacolor.sty
oberdiek.luacolor.lua       → scripts/oberdiek/oberdiek.luacolor.lua
luacolor.lua                → scripts/oberdiek/luacolor.lua
oberdiek.luacolor-pre065.lua → scripts/oberdiek/oberdiek.luacolor-pre065.lua
luacolor-pre065.lua         → scripts/oberdiek/luacolor-pre065.lua
luacolor.pdf                → doc/latex/oberdiek/luacolor.pdf
test/luacolor-test1.tex     → doc/latex/oberdiek/test/luacolor-test1.tex
test/luacolor-test2.tex     → doc/latex/oberdiek/test/luacolor-test2.tex
test/luacolor-test3.tex     → doc/latex/oberdiek/test/luacolor-test3.tex
luacolor.dtx                → source/latex/oberdiek/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.

## 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 `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 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 pdfLaTeX:

```
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.)

# 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.