The luaotfload package

Elie Roux * Khaled hosny †

2010/05/10 v1.07

Abstract

Adaptation of ConTEXt font loading system, providing the ability to load OpenType fonts with a lot of features, and extended font loading syntax.

Contents

1 Documentation

1.1 Introduction

Font management and installation has always been painful with TEX (and even more with LATEX). A lot of files are needed for one font (tfm, pfb, map, fd, vf), and they are limited to 256 characters. But the font world has evolved since, and new standard types of fonts have appeared, like TrueType and OpenType fonts. These fonts can contain a lot of characters, and have some functionalities (ligatures, old-style numbers, small capitals, etc.). They are everywhere, as the system fonts and most modern text softwares fonts are of this type. Until now the (almost) only way to use them with TEX was to use them with XETEX.

Unlike XETEX, LuaTEX does not provide facilities for these fonts by default, but it provides a way to hook lua code in some points of the TEX algorithm,

*elie.roux@telecom-bretagne.eu
†khaledhosny@eglug.org

for instance we can improve the font loading system; this is what we do in this package.

This package is quite low-level, and should be loaded directly in the macro package, like it is in ConTEXt. Sadly, Plain and LATEX are frozen and it's even impossible to adapt them to the new engines.

1.2 Loading fonts

luaotfload supports an extended font loading syntax which looks like:

`\font\foo={`⟨*prefix*⟩`:`⟨*font name*⟩`:`⟨*font features*⟩`}` ⟨*TEX font features*⟩

The curly brackets are optional and are used for escaping spaces in font names (XETEX-like double quotes can also used for the same purpose).

Prefix Can be either `file:` or `name:` and are used to select between filename based or font name based search mechanisms. Loading fonts based on filename is restricted to files found by kpathsea (typically in the TEXMF tree). Surrounding font name with square brackets is synonym to using `file:` prefix (for compatibility with XETEX). This is usually used for loading old TFM fonts. Accessing fonts by fontname allows loading system installed fonts as well as TEXMF ones, and requires a font names database that can be generated using the bundled `mkluatexfontdb.lua` script. [1]

If no prefix is specified, then `file:` is assumed.

Font name Font name can be either a font filename or actual font name; based on the prefix specified.

Font features Font features are a semicolon separated list of items, which are either `key=value` font parameters, or switches to enable/disable certain font features, in the form of `+feat`/`-feat`. The supported keys are:

mode

luaotfload has two OpenType processing modes; `base` mode which supports only a subset of OpenType features and works by mapping those features to traditional TEX ligaturing and kerning mechanisms and is a bit faster, and `node` mode which, hopefully, supports OpenType fully and works by direct processing of node list at lua end and is a bit slower. Note that `node` mode doesn't work inside math.

By default, `base` mode is used, however it is advisable to always enable `node` made, except for math fonts, otherwise many OpenType features will not function properly or even not work at all, especially for advanced scripts like Arabic.

[1] run `mkluatexfontdb.lua --help` for help and usage information

script
: OpenType script string, default value is `dflt`. Some fonts don't assign features to `dflt` script, in this case script need to be set explicitly.

language
: OpenType language string, default value is `latn`.

featurefile
: feature files are textual representation of OpenType tables and can be used to extend OpenType features of the font on fly. The file name of the feature file is passed, then features defined in the file can be enabled/disabled like any other feature. The actual syntax is described at http://fontforge.sourceforge.net/featurefile.html and http://www.adobe.com/devnet/opentype/afdko/topic_feature_file_syntax.html.

  For example, to set a `tkrn` feature from `mykern.fea` file:

  ```
  \font\lmr=Latin Modern Roman:featurefile=mykern.fea;+tkrn
  ```

color
: font color, defined as a triplet of two-digit Hex RGB values, with optionally another value for the transparency (where `00` is completely transparent and `FF` is opaque.)

  For example, to set text in semitransparent red:

  ```
  \font\lmr=Latin Modern Roman:color=FF0000BB
  ```

protrusion & expansion
: Both keys control microtypographic features of the font, namely glyph protrusion and expansion. The value of the key is the name of predefined lua tables of protrusion and expansion values, see the bottom of `otfl-font-dum.lua` file for an example of such tables. The only predefined value is `default`.

  For example, to enable default protrusion[2]:

  ```
  \font\lmr=Latin Modern Roman:protrusion=default
  ```

Non-standard font features luaotfload defines some additional font feature not defined in OpenType, currently three features are defined:

- `anum`: replaces European numbers with eastern Arabic numbers or Persian numbers, depending on the value of `language`.
- `tlig`: applies legacy TEX ligatures (`‘‘’’-- -- !‘ ?‘ <<>>`).
- `trep`: applies legacy TEX replacements (`‘’"`).

[2]You also need to set `\pdfprotrudechars2 \pdfadjustspacing2` to activate protrusion and expansion, respectively. See PDFTEX manual for details

1.3 ConTEXt files needed

This package is a wrapper for several files taken from the ConTEXt macro package. The philosophy is to let ConTEXt do all the implementation and update these files from time to time. To do so we did not modify the files taken from ConTEXt, we only changed their names to prevent name clashes. You can thus update the font system of this package simply by updating the files taken from ConTEXt, without (theoretically) changing the `.sty` file nor the main `.lua` file.

The ConTEXt files are renamed by adding the prefix `otfl-` to them (`otfl` as `OTF` Load). The files are:

- `luat-dum.lua`
- `data-con.lua`
- `node-ini.lua`
- `node-inj.lua`
- `node-fnt.lua`
- `node-dum.lua`
- `font-ini.lua`
- `font-tfm.lua`
- `font-cid.lua`
- `font-ott.lua`
- `font-otf.lua`
- `font-otd.lua`
- `font-oti.lua`
- `font-otb.lua`
- `font-otn.lua`
- `font-ota.lua`
- `font-otc.lua`
- `font-def.lua`
- `font-xtx.lua`
- `font-map.lua`
- `font-dum.lua`

1.4 Troubleshooting

If you encounter problems with some fonts, please first update to the latest version of this package before reporting a bug, as this package is under active development.

A very common problem is the lack of features for some OpenType fonts even when specified. It can be related to the fact that some fonts do not provide features for the `dflt` script, which is the default one in this package, so you may have to specify the script in the command line, for example:

`\font\myfont = MyFont.otf:script=latn;+liga;`

Also, some feature like contextual substitution, `calt`, will only work with `node` mode.

2 `luaotfload.lua`

First some usual initializations.

```
1 module('luaotfload', package.seeall)
2
3 luaotfload.module = {
4     name          = "luaotfload",
5     version       = 1.07,
6     date          = "2010/05/10",
7     description   = "ConTeXt font loading system.",
8     author        = "Elie Roux & Hans Hagen",
9     copyright     = "Elie Roux",
10    license       = "CC0"
11 }
12
```

```
13 luatexbase.provides_module(luaotfload.module)
14
```

Some helper functions.

```
15
16 local format = string.format
17
18 local function log(...)
19     luatexbase.module_log    ('luaotfload', format(...))
20 end
21
22 local function error(...)
23     luatexbase.module_error  ('luaotfload', format(...))
24 end
25
26 local function warning(...)
27     luatexbase.module_warning('luaotfload', format(...))
28 end
29
```

The minimal required LuaTEX version.

```
30
31 local luatex_version = 60
32
33 if tex.luatexversion < luatex_version then
34     warning('LuaTeX v%.2f is old, v%.2f is recommended.',
35             tex.luatexversion/100,
36             luatex_version   /100)
37 end
38
```

We load the ConTEXt files with this function. It automatically adds the `otfl-` prefix to it, so that we call it with the actual ConTEXt name.

```
39
40 function luaotfload.loadmodule(name)
41     local tofind = 'otfl-'..name
42     local found = kpse.find_file(tofind,"tex")
43     if found then
44         log('loading file %s.', found)
45         dofile(found)
46     else
47         error('file %s not found.', tofind)
48     end
49 end
50
```

We start loading some lua files. These two are some code not used by ConTEXt at all that allow other modules to be used, it provides some low-level ConTEXt functions.

```
51
```

```
luaotfload.loadmodule('luat-dum.lua') -- not used in context at all
luaotfload.loadmodule('luat-ovr.lua') -- override some luat-dum functions
luaotfload.loadmodule('data-con.lua') -- maybe some day we don't need this one

```

This one is for node support.

```

luaotfload.loadmodule('node-ini.lua')

```

By default ConTEXt takes some private attributes for internal use. With Plain and LaTeX we can't do so, we use `\newluaattribute`. This functions overrides a function defined in the previous module that returns the number of a private attribute. We allocate new attributes in the `.sty` file, and this function returns their number. Like this we don't need any private attribute, and this package is compatible with the others. We use the `otfl@` prefix for attributes.

```

function attributes.private(name)
    local number = luatexbase.attributes['otfl@'..name]
    if not number then
        error('asking for attribute %s, but not declared. '
            ..'Please report to the maintainer of luaotfload.',
               name)
    end
    return number
end

```

Some more modules. We don't load neither `font-enc.lua` nor `font-afm.lua` as it will never be used here.

We also remove a warning from `node-fnt.lua` as it is ConTEXt specific.

```

tex.attribute[0] = 0

luaotfload.loadmodule('node-res.lua')
luaotfload.loadmodule('node-inj.lua')
luaotfload.loadmodule('node-fnt.lua')
luaotfload.loadmodule('node-dum.lua')

luaotfload.loadmodule('font-ini.lua')
luaotfload.loadmodule('font-tfm.lua')
luaotfload.loadmodule('font-cid.lua')
luaotfload.loadmodule('font-map.lua')
luaotfload.loadmodule('font-ott.lua')
luaotfload.loadmodule('font-otf.lua')
luaotfload.loadmodule('font-otd.lua')
luaotfload.loadmodule('font-oti.lua')
luaotfload.loadmodule('font-otb.lua')
luaotfload.loadmodule('font-otn.lua')
```

```
luaotfload.loadmodule('font-ota.lua')
luaotfload.loadmodule('font-otc.lua')
luaotfload.loadmodule('font-def.lua')
luaotfload.loadmodule('font-xtx.lua')
luaotfload.loadmodule('font-dum.lua')
luaotfload.loadmodule('font-nms.lua')
luaotfload.loadmodule('font-clr.lua')

```

Finally two functions

```

function luaotfload.register_callbacks()
    luatexbase.add_to_callback('pre_linebreak_filter',
                               nodes.simple_font_handler,
                              'luaotfload.pre_linebreak_filter')
    luatexbase.add_to_callback('hpack_filter',
                               nodes.simple_font_handler,
                              'luaotfload.hpack_filter')
    luatexbase.reset_callback('define_font')
    luatexbase.add_to_callback('define_font',
                               fonts.define.read,
                              'luaotfload.define_font', 1)
    luatexbase.add_to_callback('find_vf_file',
                               fonts.vf.find,
                              'luaotfload.find_vf_file')
end

function luaotfload.unregister_callbacks()
    luatexbase.remove_from_callback('pre_linebreak_filter',
                                    'luaotfload.pre_linebreak_filter')
    luatexbase.remove_from_callback('hpack_filter',
                                    'luaotfload.hpack_filter')
    luatexbase.remove_from_callback('define_font',
                                    'luaotfload.define_font')
    luatexbase.remove_from_callback('find_vf_file',
                                     'luaotfload.find_vf_file')
end
```

3 luaotfload.sty

Classical Plain+LaTeX package initialization.

```
\csname ifluaotfloadloaded\endcsname
\let\ifluaotfloadloaded\endinput

\expandafter\ifx\csname ProvidesPackage\endcsname\relax
  \input luatextra.sty
\else
  \NeedsTeXFormat{LaTeX2e}
```

```
  \ProvidesPackage{luaotfload}%
    [2010/05/10 v1.07 ConTeXt font loading system]
  \RequirePackage{luatextra}
\fi

\expandafter\edef\csname otfl@AtEnd\endcsname{%
  \catcode64 \the\catcode64\relax
}

\catcode64 11

```

The attributes are allocated here. The `otfl@` prefix is added to prevent name collision.

```

\newluatexattribute\otfl@state
\newluatexattribute\otfl@markbase
\newluatexattribute\otfl@markdone
\newluatexattribute\otfl@markmark
\newluatexattribute\otfl@cursbase
\newluatexattribute\otfl@curscurs
\newluatexattribute\otfl@cursdone
\newluatexattribute\otfl@kernpair
\newluatexattribute\otfl@color

```

Two small macros to register or unregister the callbacks. Without the callbacks this package is totally turned off.

```

\def\otfl@off{
\directlua{luaotfload.unregister_callbacks()}
}

\def\otfl@on{
\directlua{luaotfload.register_callbacks()}
}

```

We load the `lua` file, and we turn the package on.

```

\luatexUseModule{luaotfload}

\otfl@on

\otfl@AtEnd
```