The luaotfload package

Elie Roux
elie.roux@telecom-bretagne.eu

2010/02/03 v1.06a

Abstract

ConTEXt font loading system, providing the possibility to load OTF fonts with a lot of features, and the XeTEX font loading syntax.

Contents

1 Documentation **1**
1.1 Introduction 1
1.2 ConTEXt files needed 2
1.3 Troubleshooting 3

2 luaotfload.lua **3**

3 luaotfload.sty **7**

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 or 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, 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 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 (theorically) 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`

1.3 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 otf 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;
```

2 luaotfload.lua

First some usual initializations.

```
luaotfload          = { }

luaotfload.module = {
    name          = "luaotfload",
    version       = 1.06,
    date          = "2010/02/03",
    description   = "ConTeXt font loading system.",
    author        = "Elie Roux & Hans Hagen",
    copyright     = "Elie Roux",
    license       = "CC0"
}

luatextra.provides_module(luaotfload.module)

```

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.

Some ConTEXt files introduce incompatibilities with old LuaTEXs, thus the old versions of these files are kept under the name `otfl-compat-...` and are called according to the version. The only file in this case is a `font-otf.lua` that, in version 2009.11.26 of ConTEXt generates invalid PDFs with LuaTEX older than version 0.45.

```


function luaotfload.loadmodule(name, compat)
    local tofind
    if compat then
        if tex.luatexversion < 45 then
            tofind = 'otfl-compat-'..name
        else
            tofind = 'otfl-'..name
        end
    else
        tofind = 'otfl-'..name
    end
```

```
    local found = kpse.find_file(tofind,"tex")
    if not found then
      luatextra.module_error('luaotfload', string.format('file %s not found.', tofind))
      return
    end
    luatextra.module_log('luaotfload', "loading file "..found)
    dofile(found)
end

```

The following functions are made to map ConTEXt functions to luaextra functions.

```

string.strip = string.stripspaces

file = fpath
file.extname = fpath.suffix

```

These are small functions that are not already in luatextra.

```

local splitters_s, splitters_m = { }, { }

function lpeg.splitat(separator,single)
    local splitter = (single and splitters_s[separator]) or splitters_m[separator]
    if not splitter then
        separator = lpeg.P(separator)
        if single then
            local other, any = lpeg.C((1 - separator)^0), lpeg.P(1)
            splitter = other * (separator * lpeg.C(any^0) + "")
            splitters_s[separator] = splitter
        else
            local other = lpeg.C((1 - separator)^0)
            splitter = other * (separator * other)^0
            splitters_m[separator] = splitter
        end
    end
    return splitter
end

function table.compact(t)
    if t then
        for k,v in next, t do
            if not next(v) then
                t[k] = nil
            end
        end
    end
end

```

```
function table.sortedhashkeys(tab) -- fast one
    local srt = { }
    for key,_ in next, tab do
        srt[#srt+1] = key
    end
    table.sort(srt)
    return srt
end

function table.reverse_hash(h)
    local r = { }
    for k,v in next, h do
        r[v] = string.lower(string.gsub(k," ",""))
    end
    return r
end

function table.reverse(t)
    local tt = { }
    if #t > 0 then
        for i=#t,1,-1 do
            tt[#tt+1] = t[i]
        end
    end
    return tt
end

```

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.

```

luaotfload.loadmodule('luat-dum.lua') -- not used in context at all
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)
```

```
109     local number = tex.attributenumber['otfl@'..name]
110     if not number then
111         luatextra.module_error('luaotfload', string.format('asking for attribute %s, but not
112     end
113     return number
114 end
115
```

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 `font-fnt.lua` as it not relevant with LuaTEXtra.

```
116
117 tex.attribute[0] = 0
118
119 luaotfload.loadmodule('node-res.lua')
120 luaotfload.loadmodule('node-inj.lua')
121 luaotfload.loadmodule('node-fnt.lua')
122 luaotfload.loadmodule('node-dum.lua')
123
124 luaotfload.loadmodule('font-ini.lua')
125 luaotfload.loadmodule('font-tfm.lua')
126 luaotfload.loadmodule('font-cid.lua')
127 luaotfload.loadmodule('font-ott.lua')
128 luaotfload.loadmodule('font-otf.lua', 1)
129 luaotfload.loadmodule('font-otd.lua')
130 luaotfload.loadmodule('font-oti.lua')
131 luaotfload.loadmodule('font-otb.lua')
132 luaotfload.loadmodule('font-otn.lua')
133 luaotfload.loadmodule('font-ota.lua')
134 luaotfload.loadmodule('font-otc.lua')
135
```

`font-def.lua` calls the function `callback.register` to register its callbacks. We override it with a dumb function so that it does not register any callback. We will register the callbacks later.

```
136
137 do
138   local temp = callback.register
139   callback.register = function (...)
140     return
141   end
142   luaotfload.loadmodule('font-def.lua')
143   callback.register = temp
144 end
145
146 luaotfload.loadmodule('font-xtx.lua')
147 luaotfload.loadmodule('font-map.lua')
148 luaotfload.loadmodule('font-dum.lua')
149
```

This is a small patch that prevents errors in some LaTeX files.

```
150
151 fonts.enc.known = {}
152
```

We have to register a function in the `find_vf_file` callback in order to make everything work.

```
153
154 function luaotfload.find_vf_file(name)
155     name = file.removesuffix(file.basename(name))
156     local result = kpse.find_file(name, "vf") or ""
157     if result == "" then
158         result = kpse.find_file(name, "ovf") or ""
159     end
160     return result
161 end
162
```

Finally two functions

```
163
164 function luaotfload.register_callbacks()
165     callback.add('pre_linebreak_filter', nodes.simple_font_handler, 'luaotfload.pre_linebrea
166     callback.add('hpack_filter',         nodes.simple_font_handler, 'luaotfload.hpack_filter
167     callback.reset('define_font')
168     callback.add('define_font', fonts.define.read, 'luaotfload.define_font', 1)
169     callback.add('find_vf_file', luaotfload.find_vf_file, 'luaotfload.find_vf_file')
170 end
171
172 function luaotfload.unregister_callbacks()
173     callback.remove('pre_linebreak_filter', 'luaotfload.pre_linebreak_filter')
174     callback.remove('hpack_filter', 'luaotfload.hpack_filter')
175     callback.remove('define_font', 'luaotfload.define_font')
176     callback.remove('find_vf_file', 'luaotfload.find_vf_file')
177 end
```

3 luaotfload.sty

Classical Plain+LaTeX package initialization.

```
178 \csname ifluaotfloadloaded\endcsname
179 \let\ifluaotfloadloaded\endinput
180
181 \expandafter\ifx\csname ProvidesPackage\endcsname\relax
182   \input luatextra.sty
183 \else
184   \NeedsTeXFormat{LaTeX2e}
185   \ProvidesPackage{luaotfload}%
186     [2010/02/03 v1.06a ConTeXt font loading system]
187   \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{
\luadirect{luaotfload.unregister_callbacks()}
}

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

```

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

```

\luatexUseModule{luaotfload}

\otfl@on

\otfl@AtEnd
```