# The `hyphen.cfg` file for LuaTEX

Khaled Hosny, Élie Roux, and Manuel Pégourié-Gonnard
khaledhosny@eglug.org
elie.roux@telecom-bretagne.eu
mpg@elzevir.fr

2010/04/28 v1.3beta

**Abstract**

This is a modified version of the file `hyphen.cfg` distributed with the babel package, with a supporting Lua module, aimed at adapting babel's hyphenation patterns loading mechanism to LuaTEX's dynamic pattern loading capabilities. It makes use of a `language.dat.lua` file (whose format is described below) that should be present in the distribution, in addition to the regular `language.dat` file.

Much of the modified code here is shared with a version of `etex.src` modified for the same reasons, which also make use of the `luatex-hyphen.lua` file described here.

## 1 Documentation

Hyphenation patterns should be loaded at runtime with LuaTEX: if they appear in the format, they will be rehashed when the format is loaded anyway, which makes the format quite long to load (many seconds even on modern machines) and provides for bad user experience. Hence, it is desirable to load as few patterns as possible in the format, and load on-demand the needed patterns at runtime.

This package provides a modified version of hyphen.cfg adapted to LuaTEX, as well as a supporting Lua module. Since a lot of things, especially the catcodes, are not as predictable at runtime than at format creation time, we don't `\input` the usual pattern files, but rather load the patterns using the Lua interface, using a special plain text version of the pattern files if available.

The existence and file name of such a version cannot be guessed, so we need a specific database: the file `language.dat.lua`. This file should be loadable by Lua and return a table whose keys are the canonical language names as found in `language.dat`, and the values are Lua tables consisting of:

1. A fixed part with two fields:

```
loader = <string> name of the TeX loader
synonyms = { <string> alternative name
```

The `loader` field is currently unused.

2. A variable part consisting of either:

   - For most languages:

     ```
     code = <string> language code
     lefthyphenmin = <number> value for \letfhyphenmin
     righthyphenmin = <number> value for \letfhyphenmin
     ```

     The `code` field determines where patterns and exceptions will be searched: in files `hyph-<code>.pat.txt` and `hyph-<code>.hyp.txt` respectively. The values of `*hyphenmin` are currently unused.

   - Special case are supported by a field `special`. Currently, two kinds of value are recognized: `'null'` for languages with no hyphenation patterns nor exceptions. `'disabled:<reason>'` allows to disable specific languages: when the user tries to load this language, an error will be issued, with the `<reason>`.

Languages that are mentioned in `language.dat` but not in `language.dat.lua` will be loaded in the format. So, if the `language.dat.lua` file is missing or incomplete, languages will just go back to the "old" behaviour, resulting in longer startup time, which does seem less bad than complete breakage.

For backward compatibility, Knuth's original patterns for US English are always loaded in the format, as `\language0`.

The modified version of `hyphen.cfg` provided here checks for the engine, and should continue to work with any engine without any modified behaviour. However, it is recommended to install it in such a way that the original `hyphen.cfg` from babel is found first by any engine other than LuaTEX.

# 2 Package code

## 2.1 luatex-hyphen.lua

```
1
2 luatexhyphen = {}
3
4 luatexhyphen.version = "1.3beta"
5
6 local dbname = "language.dat.lua"
7
8 local function warn (msg, ...)
9     texio.write_nl('luatex-hyphen: '..string.format(msg, ...))
10 end
11
12 luatexhyphen.language_dat = {}
13 local dbfile = kpse.find_file(dbname)
14 if not dbfile then
15     warn("file not found: "..dbname)
```

```
16 else
17     luatexhyphen.language_dat = dofile(dbfile)
18 end
19
20 function luatexhyphen.lookupname(l)
21     if luatexhyphen.language_dat[l] then
22         return luatexhyphen.language_dat[l], l
23     else
24         for orig,lt in pairs(luatexhyphen.language_dat) do
25             for _,syn in ipairs(lt.synonyms) do
26                 if syn == l then
27                     return lt, orig
28                 end
29             end
30         end
31     end
32     return nil
33 end
34
35 function luatexhyphen.loadlanguage(l, id)
36     local lt, orig = luatexhyphen.lookupname(l)
37     if not lt then
38         warn("no entry in %s for this language: %s", dbname, l)
39         return
40     end
41     local msg = "loading%s patterns and exceptions for: %s (\\language%d)"
42     if lt.special then
43         if lt.special == 'null' then
44             warn(msg, ' (null)', orig, id)
45         elseif lt.special:find('^disabled:') then
46             warn("language disabled by %s: %s (%s)", dbname, orig,
47                 lt.special:gsub('^disabled:', ''))
48         else
49             warn("bad entry in %s for language %s")
50         end
51         return
52     end
53     warn(msg, '', orig, id)
54     for ext, fun in pairs({pat = lang.patterns, hyp = lang.hyphenation}) do
55         local n = 'hyph-'..lt.code..'.'..ext..'.txt'
56         local f = kpse.find_file(n)
57         if not f then
58             warn("file not found: %s", n)
59             return
60         end
61         f = io.open(f, 'r')
62         local data = f:read('*a')
63         f:close()
64         if not data then
65             warn("file not readable: %s", f)
```

```
            return
        end
        fun(lang.new(id), data)
    end
end
```

## 2.2 hyphen.cfg

```
\ifx\ProvidesFile\@undefined
  \def\ProvidesFile#1[#2 #3 #4]{%
    \wlog{File: #1 #4 #3 <#2>}%
```

Use a modified banner for LuaTEX.

```
    \ifx\directlua\@undefined
      \toks8{Babel <#3> and hyphenation patterns for }%
    \else
      \toks8{LuaTeX adaptation of babel <#3>
        and hyphenation patterns for }%
    \fi
    \let\ProvidesFile\@undefined
    }
  \def\ProvidesLanguage#1[#2 #3 #4]{%
    \wlog{Language: #1 #4 #3 <#2>}%
    }
\else
  \let\bbl@tempa\ProvidesFile
  \def\ProvidesFile#1[#2 #3 #4]{%
```

Same here.

```
    \ifx\directlua\@undefined
      \toks8{Babel <#3> and hyphenation patterns for }%
    \else
      \toks8{LuaTeX adaptation of babel <#3>
        and hyphenation patterns for }%
    \fi
    \bbl@tempa#1[#2 #3 #4]%
    \let\ProvidesFile\bbl@tempa}
  \def\ProvidesLanguage#1{%
    \begingroup
      \catcode`\ 10 %
      \@makeother\/%
      \@ifnextchar[%]
        {\@provideslanguage{#1}}{\@provideslanguage{#1}[]}}
  \def\@provideslanguage#1[#2]{%
    \wlog{Language: #1 #2}%
    \expandafter\xdef\csname ver@#1.ldf\endcsname{#2}%
    \endgroup}
\fi
```

File identification is modified again.

```
\ProvidesFile{hyphen.cfg}
                [2010/04/26 v3.8l-luatex-1.3beta %
      Language switching mechanism for LuaTeX, adapted from babel v3.8l]
\ifx\AtBeginDocument\@undefined
  \input plain.def\relax
\fi
\ifx\language\@undefined
  \csname newcount\endcsname\language
\fi
\ifx\newlanguage\@undefined
  \csname newcount\endcsname\last@language
\else
  \countdef\last@language=19
\fi
\ifx\newlanguage\@undefined
  \def\addlanguage#1{%
    \global\advance\last@language \@ne
    \ifnum\last@language<\@cclvi
    \else
        \errmessage{No room for a new \string\language!}%
    \fi
    \global\chardef#1\last@language
    \wlog{\string#1 = \string\language\the\last@language}}
\else
  \def\addlanguage{\alloc@9\language\chardef\@cclvi}
\fi
\def\adddialect#1#2{%
    \global\chardef#1#2\relax
    \wlog{\string#1 = a dialect from \string\language#2}}
\def\iflanguage#1{%
  \expandafter\ifx\csname l@#1\endcsname\relax
    \@nolanerr{#1}%
  \else
    \bbl@afterfi{\ifnum\csname l@#1\endcsname=\language
      \expandafter\@firstoftwo
    \else
      \expandafter\@secondoftwo
    \fi}%
  \fi}
\edef\selectlanguage{%
  \noexpand\protect
  \expandafter\noexpand\csname selectlanguage \endcsname
  }
\ifx\@undefined\protect\let\protect\relax\fi
\ifx\documentclass\@undefined
  \def\xstring{\string\string\string}
\else
  \let\xstring\string
\fi
```

```
\xdef\bbl@language@stack{}
\def\bbl@push@language{%
  \xdef\bbl@language@stack{\languagename+\bbl@language@stack}%
  }
\def\bbl@pop@lang#1+#2-#3{%
  \def\languagename{#1}\xdef#3{#2}%
  }
\def\bbl@pop@language{%
  \expandafter\bbl@pop@lang\bbl@language@stack-\bbl@language@stack
  \expandafter\bbl@set@language\expandafter{\languagename}%
  }
\expandafter\def\csname selectlanguage \endcsname#1{%
  \bbl@push@language
  \aftergroup\bbl@pop@language
  \bbl@set@language{#1}}
\def\bbl@set@language#1{%
  \edef\languagename{%
    \ifnum\escapechar=\expandafter`\string#1\@empty
    \else \string#1\@empty\fi}%
  \select@language{\languagename}%
  \if@filesw
    \protected@write\@auxout{}{\string\select@language{\languagename}}%
    \addtocontents{toc}{\xstring\select@language{\languagename}}%
    \addtocontents{lof}{\xstring\select@language{\languagename}}%
    \addtocontents{lot}{\xstring\select@language{\languagename}}%
  \fi}
\def\select@language#1{%
  \expandafter\ifx\csname l@#1\endcsname\relax
    \@nolanerr{#1}%
  \else
    \expandafter\ifx\csname date#1\endcsname\relax
      \@noopterr{#1}%
    \else
      \bbl@patterns{\languagename}%
      \originalTeX
      \expandafter\def\expandafter\originalTeX
          \expandafter{\csname noextras#1\endcsname
                       \let\originalTeX\@empty}%
      \languageshorthands{none}%
      \babel@beginsave
      \csname captions#1\endcsname
      \csname date#1\endcsname
      \csname extras#1\endcsname\relax
      \babel@savevariable\lefthyphenmin
      \babel@savevariable\righthyphenmin
      \expandafter\ifx\csname #1hyphenmins\endcsname\relax
        \set@hyphenmins\tw@\thr@@\relax
      \else
        \expandafter\expandafter\expandafter\set@hyphenmins
          \csname #1hyphenmins\endcsname\relax
```

```
      \fi
    \fi
  \fi}
\long\def\otherlanguage#1{%
  \csname selectlanguage \endcsname{#1}%
  \ignorespaces
  }
\long\def\endotherlanguage{%
  \originalTeX
  \global\@ignoretrue\ignorespaces
  }
\expandafter\def\csname otherlanguage*\endcsname#1{%
  \foreign@language{#1}%
  }
\expandafter\def\csname endotherlanguage*\endcsname{%
  \csname noextras\languagename\endcsname
  }
\def\foreignlanguage{\protect\csname foreignlanguage \endcsname}
\expandafter\def\csname foreignlanguage \endcsname#1#2{%
  \begingroup
    \originalTeX
    \foreign@language{#1}%
    #2%
    \csname noextras#1\endcsname
  \endgroup
  }
\def\foreign@language#1{%
  \def\languagename{#1}%
  \expandafter\ifx\csname l@#1\endcsname\relax
    \@nolanerr{#1}%
  \else
    \bbl@patterns{\languagename}%
    \languageshorthands{none}%
    \csname extras#1\endcsname
    \expandafter\ifx\csname #1hyphenmins\endcsname\relax
      \set@hyphenmins\tw@\thr@@\relax
    \else
      \expandafter\expandafter\expandafter\set@hyphenmins
        \csname #1hyphenmins\endcsname\relax
    \fi
  \fi
  }
\def\bbl@patterns#1{%
  \language=\expandafter\ifx\csname l@#1:\f@encoding\endcsname\relax
    \csname l@#1\endcsname
  \else
    \csname l@#1:\f@encoding\endcsname
  \fi\relax
```

With LuaTEX, load patterns and exceptions on the fly using functions from the

supporting Lua module, unless of course they are already loaded for this language (identified by its number to avoid problems with synonyms).

Also, since this code will be executed at runtime, be careful while testing if we're using LuaTEX.

```
  \ifx\directlua\@undefined\else
    \ifx\directlua\relax\else
      \ifcsname lu@texhyphen@loaded@\the\language\endcsname \else
        \global\@namedef{lu@texhyphen@loaded@\the\language}{}%
        \directlua{
          if not luatexhyphen then
              dofile(assert(kpse.find_file("luatex-hyphen.lua")))
          end
          luatexhyphen.loadlanguage("\luatexluaescapestring{#1}",
            \the\language)}%
      \fi
    \fi
  \fi
}
\def\hyphenrules#1{%
  \expandafter\ifx\csname l@#1\endcsname\@undefined
    \@nolanerr{#1}%
  \else
    \bbl@patterns{#1}%
    \languageshorthands{none}%
       \expandafter\ifx\csname #1hyphenmins\endcsname\relax
         \set@hyphenmins\tw@\thr@@\relax
       \else
         \expandafter\expandafter\expandafter\set@hyphenmins
         \csname #1hyphenmins\endcsname\relax
       \fi
  \fi
  }
\def\endhyphenrules{}
\def\providehyphenmins#1#2{%
  \expandafter\ifx\csname #1hyphenmins\endcsname\relax
    \@namedef{#1hyphenmins}{#2}%
  \fi}
\def\set@hyphenmins#1#2{\lefthyphenmin#1\righthyphenmin#2}
\def\LdfInit{%
  \chardef\atcatcode=\catcode`\@
  \catcode`\@=11\relax
  \input babel.def\relax
  \catcode`\@=\atcatcode \let\atcatcode\relax
  \LdfInit}
\ifx\originalTeX\@undefined\let\originalTeX\@empty\fi
\ifx\babel@beginsave\@undefined\let\babel@beginsave\relax\fi
\ifx\PackageError\@undefined
  \def\@nolanerr#1{%
    \errhelp{Your command will be ignored, type <return> to proceed}%
```

```
    \errmessage{You haven't defined the language #1\space yet}}
  \def\@nopatterns#1{%
    \message{No hyphenation patterns were loaded for}%
    \message{the language `#1'}%
    \message{I will use the patterns loaded for \string\language=0
          instead}}
  \def\@noopterr#1{%
    \errmessage{The option #1 was not specified in \string\usepackage}
    \errhelp{You may continue, but expect unexpected results}}
  \def\@activated#1{%
    \wlog{Package babel Info: Making #1 an active character}}
\else
  \newcommand*{\@nolanerr}[1]{%
    \PackageError{babel}%
                {You haven't defined the language #1\space yet}%
        {Your command will be ignored, type <return> to proceed}}
  \newcommand*{\@nopatterns}[1]{%
    \PackageWarningNoLine{babel}%
        {No hyphenation patterns were loaded for\MessageBreak
          the language `#1'\MessageBreak
          I will use the patterns loaded for \string\language=0
          instead}}
  \newcommand*{\@noopterr}[1]{%
    \PackageError{babel}%
                {You haven't loaded the option #1\space yet}%
            {You may proceed, but expect unexpected results}}
  \newcommand*{\@activated}[1]{%
    \PackageInfo{babel}{%
      Making #1 an active character}}
\fi
\def\process@line#1#2 #3/{%
  \ifx=#1
    \process@synonym#2 /
  \else
    \process@language#1#2 #3/%
  \fi
  }
\toks@{}
\def\process@synonym#1 /{%
  \ifnum\last@language=\m@ne
    \expandafter\chardef\csname l@#1\endcsname0\relax
    \wlog{\string\l@#1=\string\language0}
    \toks@\expandafter{\the\toks@
      \expandafter\let\csname #1hyphenmins\expandafter\endcsname
      \csname\languagename hyphenmins\endcsname}%
  \else
    \expandafter\chardef\csname l@#1\endcsname\last@language
    \wlog{\string\l@#1=\string\language\the\last@language}
    \expandafter\let\csname #1hyphenmins\expandafter\endcsname
    \csname\languagename hyphenmins\endcsname
```

```
  \fi
  }
\def\process@language#1 #2 #3/{%
  \expandafter\addlanguage\csname l@#1\endcsname
  \expandafter\language\csname l@#1\endcsname
  \def\languagename{#1}%
```

In the LuaTEXcase, we have to decide wether to load the language now.

```
  \ifx\directlua\@undefined
    \global\toks8\expandafter{\the\toks8#1, }%
  \else
    \directlua{
      if not luatexhyphen then
        dofile(assert(kpse.find_file("luatex-hyphen.lua")))
      end
      processnow = (tex.language == 0) or
        (luatexhyphen.lookupname("\luatexluaescapestring{#1}") == nil)}%
    \ifnum0=\directlua{tex.sprint(processnow and "0" or "1")}\relax
      \global\toks8\expandafter{\the\toks8#1, }%
      \global\@namedef{lu@texhyphen@loaded@\the\language}{}%
    \fi
  \fi

  \begingroup
    \bbl@get@enc#1:\@@@
    \ifx\bbl@hyph@enc\@empty
    \else
      \fontencoding{\bbl@hyph@enc}\selectfont
    \fi
    \lefthyphenmin\m@ne
```

Assume the first (that is, zeroth) language in `language.dat` is English. This assumption is very reasonnable, since otherwise it would break compatibility with frozen TEXby not providing Knuth's orginal patterns as `\language0`, so we're pretty sure about this point. We do load this first language, since we want Knuth's patterns to be active as soon as the format is loaded.

```
    \ifx\directlua\@undefined
      \input #2\relax
    \else
      \ifnum0=\directlua{tex.sprint(processnow and "0" or "1")}\relax
        \input #2\relax
      \fi
    \fi

    \ifnum\lefthyphenmin=\m@ne
    \else
      \expandafter\xdef\csname #1hyphenmins\endcsname{%
        \the\lefthyphenmin\the\righthyphenmin}%
    \fi
  \endgroup
  \ifnum\the\language=\z@
    \expandafter\ifx\csname #1hyphenmins\endcsname\relax
```

```
      \set@hyphenmins\tw@\thr@@\relax
    \else
      \expandafter\expandafter\expandafter\set@hyphenmins
        \csname #1hyphenmins\endcsname
    \fi
    \the\toks@
  \fi
  \toks@{}%
  \def\bbl@tempa{#3}%
  \ifx\bbl@tempa\@empty
  \else
    \ifx\bbl@tempa\space
    \else
```

Likewise, don't load hyphenation exceptions now, but rather when we load the patterns. (Anyway, in practice, the third field of `language.dat` is never used since exceptions are defined in the same file as patterns, so it doesn't really matter.)

There are no hyphenation exceptions for english, and since it is frozen, we can rely on this, so no need for a special case for english here.

```
      \ifx\directlua\@undefined
        \input #3\relax
      \else
        \ifnum0=\directlua{tex.sprint(processnow and "0" or "1")}\relax
          \input #3\relax
        \fi
        \directlua{processnow = nil}%
      \fi
    \fi
  \fi
  }
\def\bbl@get@enc#1:#2\@@@{%
  \def\bbl@tempa{#1}%
  \def\bbl@tempb{#2}%
  \ifx\bbl@tempb\@empty
    \let\bbl@hyph@enc\@empty
  \else
    \bbl@get@enc#2\@@@
    \edef\bbl@hyph@enc{\bbl@tempa}%
  \fi}
\openin1 = language.dat
\ifeof1
  \message{I couldn't find the file language.dat,\space
           I will try the file hyphen.tex}
  \input hyphen.tex\relax
\else
  \last@language\m@ne
  \loop
    \endlinechar\m@ne
    \read1 to \bbl@line
    \endlinechar`\^^M
```

```
    \ifx\bbl@line\@empty
    \else
      \edef\bbl@line{\bbl@line\space/}%
      \expandafter\process@line\bbl@line
    \fi
    \iftrue \csname fi\endcsname
    \csname if\ifeof1 false\else true\fi\endcsname
  \repeat
  \language=0
\fi
\closein1
\let\process@language\@undefined
\let\process@synonym\@undefined
\let\process@line\@undefined
\let\bbl@tempa\@undefined
\let\bbl@tempb\@undefined
\let\bbl@eq@\@undefined
\let\bbl@line\@undefined
\let\bbl@get@enc\@undefined
\ifx\addto@hook\@undefined
\else
  \expandafter\addto@hook\expandafter\everyjob\expandafter{%
    \expandafter\typeout\expandafter{\the\toks8 loaded.}}
\fi
```