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 v1.4

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.

There is a version of `etex.src` modified for the same reasons using similar code, which also makes use of the `luatex-hyphen.lua` and `language.dat.lua` files 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 one mandatory field:

```
synonyms = { <string> alternative name, ...}
```

This field's value must be the same as in `language.dat`.

2. A variable part consisting of either:

   - For most languages:

     ```
     patterns = <string> filenames for patterns
     hyphenation = <string> filenames for exceptions
     ```

     Each string contains a coma-separated list of file names (whitespace before or after the coma is not accepted). The files given by `patterns` (resp. `hypenation`) must be plain text files encoded in utf8, with only patterns (resp. exceptions) and not even comments: their content will be used directly without being parsed by TeX. If one of these keys is missing or is the empty string, it is ignored and no patterns (resp. exceptions) are loaded for this language.

   - Special cases are supported by a field `special`. Currently, the following kind of values are recognized:

     `'disabled:<reason>'` allows to disable specific languages: when the user tries to load this language, an error will be issued, with the `<reason>`.

     `'language0'` only `english` should use this type of special, to indicate it is normally dumped in the format as `\language0` (see below).

     Special languages may have `*hyphenmin` information when it makes sense (mostly `\language0`).

3. Optional fields may be added. For example:

   ```
   loader = <string> name of the TeX loader
   lefthyphenmin = <number> value for \lefthyphenmin
   righthyphenmin = <number> value for \righthyphenmin
   ```

   Those fields are present in `language.dat.lua` as generated by `tlmgr`, for example, but they *are not* used by the present code in any way.

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 seems less bad than complete breakage.

For backward compatibility, Knuth's original patterns for US English are always loaded in the format, as `\language0`.[1]

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.

[1]It is assumed to be the first entry in `language.dat`.

2 Implementation

2.1 luatex-hyphen.lua

```
⟨*lua⟩
```

Start a Lua module, importing only the necessary functions as locals.

```
local error, dofile, pairs, ipairs = error, dofile, pairs, ipairs
local io, texio, lang, kpse = io, texio, lang, kpse
module('luatexhyphen')
```

Two functions for error and information reporting.

```
local function wlog(msg, ...)
    texio.write_nl('log', 'luatex-hyphen: '..msg:format(...))
end
local function err(msg, ...)
    error('luatex-hyphen: '..msg:format(...), 2)
end
```

Load the `language.dat.lua` file with the Lua version of the language database.

```
local dbname = "language.dat.lua"
local language_dat
local dbfile = kpse.find_file(dbname)
if not dbfile then
    err("file not found: "..dbname)
else
    wlog('using data file: %s', dbfile)
    language_dat = dofile(dbfile)
end
```

Look up a language in the database, and return the associated information, as well as the canonical name of the language.

```
function lookupname(name)
    if language_dat[name] then
        return language_dat[name], name
    else
        for canon, data in pairs(language_dat) do
            for _,syn in ipairs(data.synonyms) do
                if syn == name then
                    return data, canon
                end
            end
        end
    end
end
```

Set hyphenation patterns and exceptions for a language given by its name (in the database) and number (value of `\language`). Doesn't return anything, but will call `error()` if things go wrong.

```
function loadlanguage(lname, id)
    local msg = "loading%s patterns and exceptions for: %s (\\language%d)"
```

Lookup the language in the database.

```
    local ldata, cname = lookupname(lname)
    if not ldata then
        err("no entry in %s for this language: %s", dbname, lname)
    end
```

Handle special languages.

```
    if ldata.special then
        if ldata.special:find('^disabled:') then
            err("language disabled by %s: %s (%s)", dbname, cname,
                ldata.special:gsub('^disabled:', ''))
        elseif ldata.special == 'language0' then
            err("\\language0 should be dumped in the format")
        else
            err("bad entry in %s for language %s")
        end
    end
```

The generic case: load hyphenation patterns and exceptions from files given by the language code.

```
    wlog(msg, '', cname, id)
    for _, item in ipairs{'patterns', 'hyphenation'} do
        local filelist = ldata[item]
        if filelist ~= nil and filelist ~= '' then
          for _, file in ipairs(filelist:explode(',')) do
            local file = kpse.find_file(file) or err("file not found: %s", file)
            local fh = io.open(file, 'r')
            local data = fh:read('*a') or err("file not readable: %s", f)
            fh:close()
            lang[item](lang.new(id), data)
          end
        else
            if item == 'hyphenation' then item = item..' exceptions' end
            wlog("info: no %s for this language", item)
        end
    end
end
```

⟨/lua⟩

2.2 hyphen.cfg

⟨*hyphen⟩

Start with unmodified code from babel.

```
\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-1.4 %
      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 LuaTEX case, we have to decide wether to load the language now. Remember our choice, since we’ll need it two times more.

If we choose to load the language now, mark it as loaded. This is done using TEX macros in order to survive the format dumping-loading cycle, which would not be as straigthforward using Lua objects.

```
  \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
```

Conditionally input the patterns file.

```
  \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
```

Conditionnaly input the exceptions file.

```
    \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
```

⟨/hyphen⟩