

The `hyphen.cfg` file for Lua_T_EX

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 Lua_T_EX's dynamic pattern loading capabilities.

1 Documentation

Hyphenation patterns should be loaded at runtime with Lua_T_EX: 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.

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

This package provides a modified version of `hyphen.cfg` adapted to Lua_T_EX, 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 kindly provided by the `texhyphen` project.

The modifications applied in this files are highlighted in the code documentation below, but here is a summary:

- not loading patterns in the format except for english
- loading patterns at runtime, except for english
- modified banner

This file 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 function warn (msg, ...)
7     texio.write_nl('luatex-hyphen: '..string.format(msg, ...))
8 end
9
10 luatexhyphen.languagesdat = {}
11
12 local languagesdatfile = kpse.find_file("languages.dat.lua")
13 if not languagesdatfile then
14     warn("file not found: languages.dat.lua")
15 else
16     luatexhyphen.languagesdat = dofile(languagesdatfile)
17 end
18
19 local function lookupname(l)
20     if luatexhyphen.languagesdat[l] then
21         return luatexhyphen.languagesdat[l], 1
22     else
23         for orig,lt in pairs(luatexhyphen.languagesdat) do
24             for _,syn in ipairs(lt.synonyms) do
25                 if syn == l then
26                     return lt, orig
27                 end
28             end
29         end
30     end
31     return nil
32 end
33
34 function luatexhyphen.loadpatterns(l, id)
35     local lt, orig = lookupname(l)
36     if not lt or not lt.code then
37         warn("no entry in languages.dat.lua for this language: %s", l)
38         return
39     end
40     local f = kpse.find_file(lt.code .. '.pat.txt')
41     if not f then
42         warn("file not found: %s", lt.code .. '.pat.txt')
```

```

43         return
44     end
45     f = io.open(f, 'r')
46     local data = f:read('*a')
47     f:close()
48     if not data then
49         warn("file not readable: %s", f)
50         return
51     end
52     local lobj = lang.new(id)
53     warn("loading patterns for: %s", orig)
54     lang.patterns(lobj, data)
55 end
56
57 function luatexhyphen.loadexceptions(l, id)
58     local lt, orig = lookupname(l)
59     if not lt or not lt.code then
60         warn("no entry in languages.dat.lua for this language: %s", l)
61         return
62     end
63     local f = kpse.find_file(lt.code .. '.hyp.txt')
64     if not f then
65         warn("file not found: %s", lt.code .. '.pat.txt')
66         return
67     end
68     f = io.open(f, 'r')
69     local data = f:read('*a')
70     f:close()
71     if not data then
72         warn("file not readable: %s", f)
73         return
74     end
75     local lobj = lang.new(id)
76     warn("loading exceptions for: %s", orig)
77     lang.hyphenation(lobj, data)
78 end
79

```

2.2 hyphen.cfg

```

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

    Use a modified banner for LuaTeX.
83     \ifx\directlua\@undefined
84       \toks8{Babel <#3> and hyphenation patterns for }%
85     \else
86       \toks8{LuaTeX adaptation of babel <#3>
87         and hyphenation patterns for }%
88     \fi

```

```

89 \let\ProvidesFile\@undefined
90 }
91 \def\ProvidesLanguage#1[#2 #3 #4]{%
92 \wlog{Language: #1 #4 #3 <#2>}%
93 }
94 \else
95 \let\bbl@tempa\ProvidesFile
96 \def\ProvidesFile#1[#2 #3 #4]{%

Same here.
97 \ifx\directlua\@undefined
98 \toks8{Babel <#3> and hyphenation patterns for }%
99 \else
100 \toks8{LuaTeX adaptation of babel <#3>
101 and hyphenation patterns for }%
102 \fi

103 \bbl@tempa#1[#2 #3 #4]%
104 \let\ProvidesFile\bbl@tempa}
105 \def\ProvidesLanguage#1{%
106 \begingroup
107 \catcode'\ 10 %
108 \@makeother\/%
109 \ifnextchar[%]
110 {\@provideslanguage{#1}}{\@provideslanguage{#1}[]}}
111 \def\@provideslanguage#1[#2]{%
112 \wlog{Language: #1 #2}%
113 \expandafter\xdef\csname ver@#1.ldf\endcsname{#2}%
114 \endgroup}
115 \fi
116

File identification is modified again.
117 \ProvidesFile{hyphen.cfg}
118 [2010/04/26 v3.81-luatex-1.3beta %
119 Language switching mechanism for LuaTeX, adapted from babel v3.81]

120 \ifx\AtBeginDocument\@undefined
121 \input plain.def\relax
122 \fi
123 \ifx\language\@undefined
124 \csname newcount\endcsname\language
125 \fi
126 \ifx\newlanguage\@undefined
127 \csname newcount\endcsname\last@language
128 \else
129 \countdef\last@language=19
130 \fi
131 \ifx\newlanguage\@undefined
132 \def\addlanguage#1{%
133 \global\advance\last@language \@ne
134 \ifnum\last@language<\@cclvi

```

```

135     \else
136         \errmessage{No room for a new \string\language!}%
137     \fi
138     \global\chardef#1\last@language
139     \wlog{\string#1 = \string\language\the\last@language}}
140 \else
141     \def\addlanguage{\alloc@9\language\chardef\@cclvi}
142 \fi
143 \def\adddialect#1#2{%
144     \global\chardef#1#2\relax
145     \wlog{\string#1 = a dialect from \string\language#2}}
146 \def\iflanguage#1{%
147     \expandafter\ifx\csname l@#1\endcsname\relax
148         \@nolanerr{#1}%
149     \else
150         \bbl@afterfi{\ifnum\csname l@#1\endcsname=\language
151             \expandafter\@firstoftwo
152         \else
153             \expandafter\@secondoftwo
154         \fi}%
155     \fi}
156 \edef\selectlanguage{%
157     \noexpand\protect
158     \expandafter\noexpand\csname selectlanguage \endcsname
159 }
160 \ifx\@undefined\protect\let\protect\relax\fi
161 \ifx\documentclass\@undefined
162     \def\xstring{\string\string\string}
163 \else
164     \let\xstring\string
165 \fi
166 \xdef\bbl@language@stack{}
167 \def\bbl@push@language{%
168     \xdef\bbl@language@stack{\language\name+\bbl@language@stack}%
169 }
170 \def\bbl@pop@lang#1+#2-#3{%
171     \def\language\name{#1}\xdef#3{#2}%
172 }
173 \def\bbl@pop@language{%
174     \expandafter\bbl@pop@lang\bbl@language@stack-\bbl@language@stack
175     \expandafter\bbl@set@language\expandafter{\language\name}%
176 }
177 \expandafter\def\csname selectlanguage \endcsname#1{%
178     \bbl@push@language
179     \aftergroup\bbl@pop@language
180     \bbl@set@language{#1}}
181 \def\bbl@set@language#1{%
182     \edef\language\name{%
183         \ifnum\escapechar=\expandafter'\string#1\@empty
184         \else \string#1\@empty\fi}%

```

```

185 \select@language{\language}%
186 \if@filesw
187   \protected@write\@auxout{}\string\select@language{\language}}%
188   \addtocontents{toc}{\xstring\select@language{\language}}%
189   \addtocontents{lof}{\xstring\select@language{\language}}%
190   \addtocontents{lot}{\xstring\select@language{\language}}%
191 \fi}
192 \def\select@language#1{%
193   \expandafter\ifx\csname l@#1\endcsname\relax
194     \@nolanerr{#1}%
195   \else
196     \expandafter\ifx\csname date#1\endcsname\relax
197       \@noopterr{#1}%
198     \else
199       \bbl@patterns{\language}%
200       \originalTeX
201       \expandafter\def\expandafter\originalTeX
202         \expandafter{\csname noextras#1\endcsname
203           \let\originalTeX\@empty}%
204       \languageshorthands{none}%
205       \babel@beginsave
206       \csname captions#1\endcsname
207       \csname date#1\endcsname
208       \csname extras#1\endcsname\relax
209       \babel@savevariable\lefthyphenmin
210       \babel@savevariable\righthyphenmin
211       \expandafter\ifx\csname #1hyphenmins\endcsname\relax
212         \set@hyphenmins\tw@\thr@@\relax
213       \else
214         \expandafter\expandafter\expandafter\set@hyphenmins
215         \csname #1hyphenmins\endcsname\relax
216       \fi
217     \fi
218   \fi}
219 \long\def\otherlanguage#1{%
220   \csname selectlanguage \endcsname{#1}%
221   \ignorespaces
222 }
223 \long\def\endotherlanguage{%
224   \originalTeX
225   \global\@ignoretrue\ignorespaces
226 }
227 \expandafter\def\csname otherlanguage*\endcsname#1{%
228   \foreign@language{#1}%
229 }
230 \expandafter\def\csname endotherlanguage*\endcsname{%
231   \csname noextras\language\endcsname
232 }
233 \def\foreignlanguage{\protect\csname foreignlanguage \endcsname}
234 \expandafter\def\csname foreignlanguage \endcsname#1#2{%

```

```

235 \begingroup
236   \originalTeX
237   \foreign@language{#1}%
238   #2%
239   \csname noextras#1\endcsname
240 \endgroup
241 }
242 \def\foreign@language#1{%
243   \def\language#1{%
244     \expandafter\ifx\csname l@#1\endcsname\relax
245       \nolater{#1}%
246     \else
247       \bbl@patterns{\language}%
248       \languageshortands{none}%
249       \csname extras#1\endcsname
250       \expandafter\ifx\csname #1hyphenmins\endcsname\relax
251         \set@hyphenmins\tw@\thr@@\relax
252       \else
253         \expandafter\expandafter\expandafter\set@hyphenmins
254         \csname #1hyphenmins\endcsname\relax
255       \fi
256     \fi
257   }
258 \def\bbl@patterns#1{%

```

With LuaTeX, load patterns and exceptions at runtime using functions from the supporting Lua module.

Remember which patterns have been loaded to avoid reloading patterns and exceptions every time the language is activated. This is done in TeX rather than in Lua so that the information about which patterns are loaded in the format is correctly remembered at runtime.¹

```

259 \ifx\directlua\@undefined\else
260   \unless\ifcsname bbl@luatex@#1@loaded\endcsname
261     \expandafter\gdef\csname bbl@luatex@#1@loaded\endcsname{%
262       \directlua{
263         if not luatexhyphen then
264           dofile(kpse.find_file("luatex-hyphen.lua"))
265         end
266         luatexhyphen.loadpatterns("\luatexluaescapestring{#1}",
267           \number\csname l@#1\endcsname)
268         luatexhyphen.loadexceptions("\luatexluaescapestring{#1}",
269           \number\csname l@#1\endcsname)
270       }%
271     \fi
272   \fi
273 \language=\expandafter\ifx\csname l@#1:\f@encoding\endcsname\relax

```

¹It is theoretically possible to do so in Lua too, saving things in a bytecode register and restoring if via `\everyjob`, but there is currently no standard mechanism for this, and the TeX works well anyway.

```

274 \csname l@#1\endcsname
275 \else
276 \csname l@#1:\f@encoding\endcsname
277 \fi\relax
278 }
279 \def\hyphenrules#1{%
280 \expandafter\ifx\csname l@#1\endcsname\@undefined
281 \@nolanerr{#1}%
282 \else
283 \bbl@patterns{#1}%
284 \languageshorthands{none}%
285 \expandafter\ifx\csname #1hyphenmins\endcsname\relax
286 \set@hyphenmins\tw@\thr@@\relax
287 \else
288 \expandafter\expandafter\expandafter\set@hyphenmins
289 \csname #1hyphenmins\endcsname\relax
290 \fi
291 \fi
292 }
293 \def\endhyphenrules{}
294 \def\providehyphenmins#1#2{%
295 \expandafter\ifx\csname #1hyphenmins\endcsname\relax
296 \@namedef{#1hyphenmins}{#2}%
297 \fi}
298 \def\set@hyphenmins#1#2{\lefthyphenmin#1\righthyphenmin#2}
299 \def\LdfInit{%
300 \chardef\atcatcode=\catcode'\@
301 \catcode'\@=11\relax
302 \input babel.def\relax
303 \catcode'\@=\atcatcode \let\atcatcode\relax
304 \LdfInit}
305 \ifx\originalTeX\@undefined\let\originalTeX\@empty\fi
306 \ifx\babel@beginsave\@undefined\let\babel@beginsave\relax\fi
307 \ifx\PackageError\@undefined
308 \def\@nolanerr#1{%
309 \errhelp{Your command will be ignored, type <return> to proceed}%
310 \errmessage{You haven't defined the language #1\space yet}}
311 \def\@nopatterns#1{%
312 \message{No hyphenation patterns were loaded for}%
313 \message{the language '#1'}%
314 \message{I will use the patterns loaded for \string\language=0
315 instead}}
316 \def\@noopterr#1{%
317 \errmessage{The option #1 was not specified in \string\usepackage}
318 \errhelp{You may continue, but expect unexpected results}}
319 \def\@activated#1{%
320 \wlog{Package babel Info: Making #1 an active character}}
321 \else
322 \newcommand*{\@nolanerr}[1]{%
323 \PackageError{babel}%

```



```

324         {You haven't defined the language #1\space yet}%
325     {Your command will be ignored, type <return> to proceed}}
326 \newcommand*{\@nopatterns}[1]{%
327     \PackageWarningNoLine{babel}%
328     {No hyphenation patterns were loaded for\MessageBreak
329     the language '#1'\MessageBreak
330     I will use the patterns loaded for \string\language=0
331     instead}}
332 \newcommand*{\@noopterr}[1]{%
333     \PackageError{babel}%
334     {You haven't loaded the option #1\space yet}%
335     {You may proceed, but expect unexpected results}}
336 \newcommand*{\@activated}[1]{%
337     \PackageInfo{babel}{%
338     Making #1 an active character}}
339 \fi
340 \def\process@line#1#2 #3/{%
341     \ifx=#1
342     \process@synonym#2 /
343     \else
344     \process@language#1#2 #3/%
345     \fi
346 }
347 \toks@{}
348 \def\process@synonym#1 /{%
349     \ifnum\last@language=\m@ne
350     \expandafter\chardef\csname l@#1\endcsname0\relax
351     \wlog{\string\l@#1=\string\language0}
352     \toks@\expandafter{\the\toks@
353     \expandafter\let\csname #1hyphenmins\expandafter\endcsname
354     \csname\language\hyphenmins\endcsname}%
355     \else
356     \expandafter\chardef\csname l@#1\endcsname\last@language
357     \wlog{\string\l@#1=\string\language\the\last@language}
358     \expandafter\let\csname #1hyphenmins\expandafter\endcsname
359     \csname\language\hyphenmins\endcsname
360     \fi
361 }
362 \def\process@language#1 #2 #3/{%
363     \expandafter\addlanguage\csname l@#1\endcsname
364     \expandafter\language\csname l@#1\endcsname
365     \def\language{#1}%

```

Yet another banner modification. See below why the test makes sense.

```

366 \ifx\directlua@\undefined
367     \global\toks8\expandafter{\the\toks8#1, }%
368 \else
369     \unless\ifcsname bbl@luatex@english@loaded\endcsname
370     \global\toks8\expandafter{\the\toks8#1 }%
371 \fi

```

```

372 \fi
373 \begingroup
374 \bbl@get@enc#1:\@@@
375 \ifx\bbl@hyph@enc\@empty
376 \else
377 \fontencoding{\bbl@hyph@enc}\selectfont
378 \fi
379 \lefthyphenmin\m@ne

```

Assume the first (that is, zeroth) language in `language.dat` is English. This assumption is very reasonable, since otherwise it would break compatibility with frozen `TEX` by not providing Knuth's original 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. But once it is done, we don't want to load any other language.

```

380 \ifx\directlua\@undefined
381 \input #2\relax
382 \else
383 \unless\ifcsname bbl@luatex@english@loaded\endcsname
384 \gdef\bbl@luatex@english@loaded{1}%
385 \input #2\relax
386 \fi
387 \fi
388 \ifnum\lefthyphenmin=\m@ne
389 \else
390 \expandafter\xdef\csname #1hyphenmins\endcsname{%
391 \the\lefthyphenmin\the\righthyphenmin}%
392 \fi
393 \endgroup
394 \ifnum\the\language=\z@
395 \expandafter\ifx\csname #1hyphenmins\endcsname\relax
396 \set@hyphenmins\tw@\thr@@\relax
397 \else
398 \expandafter\expandafter\expandafter\set@hyphenmins
399 \csname #1hyphenmins\endcsname
400 \fi
401 \the\toks@
402 \fi
403 \toks@{}%
404 \def\bbl@tempa{#3}%
405 \ifx\bbl@tempa\@empty
406 \else
407 \ifx\bbl@tempa\space
408 \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.

```

409 \ifx\directlua\@undefined
410 \input #3\relax
411 \fi
412 \fi
413 \fi
414 }
415 \def\bbl@get@enc#1:#2\@@@{%
416 \def\bbl@tempa{#1}%
417 \def\bbl@tempb{#2}%
418 \ifx\bbl@tempb\@empty
419 \let\bbl@hyph@enc\@empty
420 \else
421 \bbl@get@enc#2\@@@
422 \edef\bbl@hyph@enc{\bbl@tempa}%
423 \fi}
424 \openin1 = language.dat
425 \ifeof1
426 \message{I couldn't find the file language.dat,\space
427 I will try the file hyphen.tex}
428 \input hyphen.tex\relax
429 \else
430 \last@language\m@ne
431 \loop
432 \endlinechar\m@ne
433 \read1 to \bbl@line
434 \endlinechar'\^M
435 \ifx\bbl@line\@empty
436 \else
437 \edef\bbl@line{\bbl@line\space/}%
438 \expandafter\process@line\bbl@line
439 \fi
440 \iftrue \csname fi\endcsname
441 \csname if\ifeof1 false\else true\fi\endcsname
442 \repeat
443 \language=0
444 \fi
445 \closein1
446 \let\process@language\@undefined
447 \let\process@synonym\@undefined
448 \let\process@line\@undefined
449 \let\bbl@tempa\@undefined
450 \let\bbl@tempb\@undefined
451 \let\bbl@eq\@undefined
452 \let\bbl@line\@undefined
453 \let\bbl@get@enc\@undefined
454 \ifx\addto@hook\@undefined
455 \else
456 \expandafter\addto@hook\expandafter\everyjob\expandafter{%

```

```
457 \expandafter\typeout\expandafter{\the\toks8 loaded.}}
458 \fi
```