

# 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`

2012/04/16 v1.5

## 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. `hyphenation`) must be plain text files encoded in UTF-8, 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`.<sup>1</sup>

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.

---

<sup>1</sup>It is assumed to be the first entry in `language.dat`.

## 2 Implementation

### 2.1 luatex-hyphen.lua

```
1 /*lua*/
```

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

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

Two functions for error and information reporting.

```
5 local function wlog(msg, ...)
6     texio.write_nl('log', 'luatex-hyphen: '..msg:format(...))
7 end
8 local function err(msg, ...)
9     error('luatex-hyphen: '..msg:format(...), 2)
10 end
```

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

```
11 local dbname = "language.dat.lua"
12 local language_dat
13 local dbfile = kpse.find_file(dbname)
14 if not dbfile then
15     err("file not found: "..dbname)
16 else
17     wlog('using data file: %s', dbfile)
18     language_dat = dofile(dbfile)
19 end
```

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

```
20 function lookupname(name)
21     if language_dat[name] then
22         return language_dat[name], name
23     else
24         for canon, data in pairs(language_dat) do
25             for _,syn in ipairs(data.synonyms) do
26                 if syn == name then
27                     return data, canon
28                 end
29             end
30         end
31     end
32 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.

```
33 function loadlanguage(lname, id)
34     if id == 0 then
35         return
```

```

36     end
37     local msg = "loading%s patterns and exceptions for: %s (\\"language%d)"
38     Lookup the language in the database.
39     local ldata, cname = lookupname(lname)
40     if not ldata then
41         err("no entry in %s for this language: %s", dbname, lname)
42     end
43
44     Handle special languages.
45     if ldata.special then
46         if ldata.special:find('^disabled:') then
47             err("language disabled by %s: %s (%s)", dbname, cname,
48                 ldata.special:gsub('^disabled:', ''))
49         elseif ldata.special == 'language0' then
50             err("\\"language0 should be dumped in the format")
51         else
52             err("bad entry in %s for language %s")
53         end
54     end
55
56     The generic case: load hyphenation patterns and exceptions from files given
57     by the language code.
58     wlog(msg, '', cname, id)
59     for _, item in ipairs{'patterns', 'hyphenation'} do
60         local filelist = ldata[item]
61         if filelist ~= nil and filelist ~= '' then
62             for _, file in ipairs(filelist:explode(',')) do
63                 local file = kpse.find_file(file) or err("file not found: %s", file)
64                 local fh = io.open(file, 'r')
65                 local data = fh:read('*a') or err("file not readable: %s", f)
66                 fh:close()
67                 lang[item](lang.new(id), data)
68             end
69         else
70             if item == 'hyphenation' then item = item..'.exceptions' end
71             wlog("info: no %s for this language", item)
72         end
73     end
74
75     function adddialect(dialect, language)
76         if dialect ~= '0' then
77             dialect = dialect:gsub('l0', '')
78             language = language:gsub('l0', '')
79             data = language_dat[language]
80             if data then
81                 data.synonyms[#data.synonyms+1] = dialect
82             end
83         end
84     end
85
86     end
87
88     end
89
90     end
91
92     end
93
94     end
95
96     end
97
98     end
99
100    end
101
102    end
103
104    end
105
106    end
107
108    end
109
110    end
111
112    end
113
114    end
115
116    end
117
118    end
119
120    end
121
122    end
123
124    end
125
126    end
127
128    end
129
130    end
131
132    end
133
134    end
135
136    end
137
138    end
139
140    end
141
142    end
143
144    end
145
146    end
147
148    end
149
150    end
151
152    end
153
154    end
155
156    end
157
158    end
159
160    end
161
162    end
163
164    end
165
166    end
167
168    end
169
170    end
171
172    end
173
174    end
175
176    end
177
178    end
179
180    end
181
182    end
183
184    end
185
186    end
187
188    end
189
190    end
191
192    end
193
194    end
195
196    end
197
198    end
199
200    end
201
202    end
203
204    end
205
206    end
207
208    end
209
210    end
211
212    end
213
214    end
215
216    end
217
218    end
219
220    end
221
222    end
223
224    end
225
226    end
227
228    end
229
230    end
231
232    end
233
234    end
235
236    end
237
238    end
239
240    end
241
242    end
243
244    end
245
246    end
247
248    end
249
250    end
251
252    end
253
254    end
255
256    end
257
258    end
259
260    end
261
262    end
263
264    end
265
266    end
267
268    end
269
270    end
271
272    end
273
274    end
275
276    end
277
278    end
279
280    end
281
282    end
283
284    end
285
286    end
287
288    end
289
290    end
291
292    end
293
294    end
295
296    end
297
298    end
299
300    end
301
302    end
303
304    end
305
306    end
307
308    end
309
310    end
311
312    end
313
314    end
315
316    end
317
318    end
319
320    end
321
322    end
323
324    end
325
326    end
327
328    end
329
330    end
331
332    end
333
334    end
335
336    end
337
338    end
339
340    end
341
342    end
343
344    end
345
346    end
347
348    end
349
350    end
351
352    end
353
354    end
355
356    end
357
358    end
359
360    end
361
362    end
363
364    end
365
366    end
367
368    end
369
370    end
371
372    end
373
374    end
375
376    end
377
378    end
379
380    end
381
382    end
383
384    end
385
386    end
387
388    end
389
390    end
391
392    end
393
394    end
395
396    end
397
398    end
399
400    end
401
402    end
403
404    end
405
406    end
407
408    end
409
410    end
411
412    end
413
414    end
415
416    end
417
418    end
419
420    end
421
422    end
423
424    end
425
426    end
427
428    end
429
430    end
431
432    end
433
434    end
435
436    end
437
438    end
439
440    end
441
442    end
443
444    end
445
446    end
447
448    end
449
450    end
451
452    end
453
454    end
455
456    end
457
458    end
459
460    end
461
462    end
463
464    end
465
466    end
467
468    end
469
470    end
471
472    end
473
474    end
475
476    end
477
478    end
479
480    end
481
482    end
483
484    end
485
486    end
487
488    end
489
490    end
491
492    end
493
494    end
495
496    end
497
498    end
499
500    end
501
502    end
503
504    end
505
506    end
507
508    end
509
510    end
511
512    end
513
514    end
515
516    end
517
518    end
519
520    end
521
522    end
523
524    end
525
526    end
527
528    end
529
530    end
531
532    end
533
534    end
535
536    end
537
538    end
539
540    end
541
542    end
543
544    end
545
546    end
547
548    end
549
550    end
551
552    end
553
554    end
555
556    end
557
558    end
559
560    end
561
562    end
563
564    end
565
566    end
567
568    end
569
570    end
571
572    end
573
574    end
575
576    end
577
578    end
579
580    end
581
582    end
583
584    end
585
586    end
587
588    end
589
589 </lua>

```

## 2.2 hyphen.cfg

```
80 <*hyphen>
81 \ifx\ProvidesFile\@undefined
82   \def\ProvidesFile#1[#2 #3 #4]{%
83     \wlog{File: #1 #4 #3 <#2>}%
84 
85     Use a modified banner for LuaTeX.
86     \ifx\directlua\@undefined
87       \toks8{Babel <#3> and hyphenation patterns for }%
88     \else
89       \toks8{LuaTeX adaptation of babel <#3>
90         and hyphenation patterns for }%
91     \fi
92 
93     \let\ProvidesFile\@undefined
94   }
95 \else
96   \let\bb@tempa\ProvidesFile
97   \def\ProvidesFile#1[#2 #3 #4]{%
98 
99     Same here.
100    \ifx\directlua\@undefined
101      \toks8{Babel <#3> and hyphenation patterns for }%
102    \else
103      \toks8{LuaTeX adaptation of babel <#3>
104        and hyphenation patterns for }%
105    \fi
106 
107    \bb@tempa#1[#2 #3 #4]%
108    \let\ProvidesFile\bb@tempa}
109    \def\ProvidesLanguage#1{%
110      \begingroup
111        \catcode`\ 10 %
112        \makeother`/%
113        \ifnextchar[%]
114          {\@provideslanguage{#1}}{\@provideslanguage{#1}[]}}
115      \def\@provideslanguage#1[#2]{%
116        \wlog{Language: #1 #2}%
117        \expandafter\xdef\csname ver@#1.ldf\endcsname{#2}%
118        \endgroup}
119    \fi
120 
121    File identification is modified again.
122    \ProvidesFile{hyphen.cfg}
123      [2012/04/16 v3.8m-luatex-1.5 %
124       Language switching mechanism for LuaTeX, adapted from babel v3.8m]
```

```

121 \ifx\AtBeginDocument\@undefined
122   \input plain.def\relax
123 \fi
124 \ifx\language\@undefined
125   \csname newcount\endcsname\language
126 \fi
127 \ifx\newlanguage\@undefined
128   \csname newcount\endcsname\last@language
129 \else
130   \countdef\last@language=19
131 \fi
132 \ifx\newlanguage\@undefined
133   \def\addlanguage#1{%
134     \global\advance\last@language \one
135     \ifnum\last@language<\@ccvi
136     \else
137       \errmessage{No room for a new \string\language!}%
138     \fi
139     \global\chardef#1\last@language
140     \wlog{\string#1 = \string\language\the\last@language}}
141 \else
142   \def\addlanguage{\alloc@9\language\chardef\@ccvi}
143 \fi
144 \def\adddialect#1#2{%
145   \global\chardef#1#2\relax
146   \ifx\directlua\@undefined\else
147     \ifx\directlua\relax\else
148       \directlua{
149         if not luatexhyphen then
150           dofile(assert(kpse.find_file("luatex-hyphen.lua")))
151         end
152         luatexhyphen.adddialect("\string#1", "\string#2")
153       }%
154     \fi
155   \fi
156   \wlog{\string#1 = a dialect from \string\language#2}}
157 \def\iflanguage#1{%
158   \expandafter\ifx\csname l@#1\endcsname\relax
159     \@nolanerr{#1}%
160   \else
161     \bblobafterfi{\ifnum\csname l@#1\endcsname=\language
162       \expandafter\@firstoftwo
163     \else
164       \expandafter\@secondoftwo
165     \fi}%
166   \fi}
167 \edef\selectlanguage{%
168   \noexpand\protect
169   \expandafter\noexpand\csname selectlanguage \endcsname
170 }

```

```

171 \ifx\@undefined\protect\let\protect\relax\fi
172 \ifx\documentclass\@undefined
173   \def\xstring{\string\string\string}
174 \else
175   \let\xstring\string
176 \fi
177 \xdef\bb@language@stack{}
178 \def\bb@push@language{%
179   \xdef\bb@language@stack{\languagename+\bb@language@stack}%
180 }
181 \def\bb@pop@lang#1+#2-#3{%
182   \def\languagename{#1}\xdef#3{#2}%
183 }
184 \def\bb@pop@language{%
185   \expandafter\bb@pop@lang\bb@language@stack-\bb@language@stack
186   \expandafter\bb@set@language\expandafter{\languagename}%
187 }
188 \expandafter\def\csname selectlanguage \endcsname#1{%
189   \bb@push@language
190   \aftergroup\bb@pop@language
191   \bb@set@language{#1}}
192 \def\bb@set@language#1{%
193   \edef\languagename{%
194     \ifnum\escapechar=\expandafter`\string#1\@empty
195     \else \string#1\@empty\fi}%
196   \select@language{\languagename}%
197   \if@filesw
198     \protected@write\auxout{}{\string\select@language{\languagename}}%
199     \addtocontents{toc}{\xstring\select@language{\languagename}}%
200     \addtocontents{lof}{\xstring\select@language{\languagename}}%
201     \addtocontents{lot}{\xstring\select@language{\languagename}}%
202   \fi}
203 \def\select@language#1{%
204   \expandafter\ifx\csname l@#1\endcsname\relax
205     \c@nolanerr{#1}%
206   \else
207     \expandafter\ifx\csname date#1\endcsname\relax
208       \c@noopterr{#1}%
209     \else
210       \bb@patterns{\languagename}%
211       \originalTeX
212       \expandafter\def\expandafter\originalTeX
213         \expandafter{\csname noextras#1\endcsname
214           \let\originalTeX\@empty}%
215       \languageshorthands{none}%
216       \babel@beginsave
217       \csname captions#1\endcsname
218       \csname date#1\endcsname
219       \csname extras#1\endcsname\relax
220       \babel@savevariable\lefthyphenmin

```

```

221      \babel@savevariable\righthyphenmin
222      \expandafter\ifx\csname #1hyphenmins\endcsname\relax
223          \set@hyphenmins\tw@thr@@\relax
224      \else
225          \expandafter\expandafter\expandafter\set@hyphenmins
226              \csname #1hyphenmins\endcsname\relax
227      \fi
228  \fi
229 \fi}
230 \long\def\otherlanguage#1{%
231   \csname selectlanguage \endcsname{#1}%
232   \ignorespaces
233 }
234 \long\def\endootherlanguage{%
235   \originalTeX
236   \global\@ignoretrue\ignorespaces
237 }
238 \expandafter\def\csname otherlanguage*\endcsname{%
239   \foreign@language{#1}%
240 }
241 \expandafter\def\csname endotherlanguage*\endcsname{%
242   \csname noextras\languagename\endcsname
243 }
244 \def\foreignlanguage{\protect\csname foreignlanguage \endcsname}
245 \expandafter\def\csname foreignlanguage \endcsname{#1#2}{%
246   \begingroup
247     \originalTeX
248     \foreign@language{#1}%
249     #2%
250     \csname noextras#1\endcsname
251   \endgroup
252 }
253 \def\foreign@language#1{%
254   \def\languagename{#1}%
255   \expandafter\ifx\csname l@#1\endcsname\relax
256       \nolanerr{#1}%
257   \else
258       \bb@patterns{\languagename}%
259       \languageshorthands{none}%
260       \csname extras#1\endcsname
261       \expandafter\ifx\csname #1hyphenmins\endcsname\relax
262           \set@hyphenmins\tw@thr@@\relax
263       \else
264           \expandafter\expandafter\expandafter\set@hyphenmins
265               \csname #1hyphenmins\endcsname\relax
266       \fi
267   \fi
268 }
269 \def\bb@patterns#1{%
270   \language=\expandafter\ifx\csname l@#1:\f@encoding\endcsname\relax

```

```

271     \csname l@#1\endcsname
272 \else
273     \csname l@#1:\f@encoding\endcsname
274 \fi\relax

```

With  $\text{\LaTeX}$ , 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  $\text{\LaTeX}$ .

```

275 \ifx\directlua\@undefined\else
276   \ifx\directlua\relax\else
277     \ifcsname lu@texhyphen@loaded@\the\language\endcsname \else
278       \global\@namedef{lu@texhyphen@loaded@\the\language}{}%
279     \directlua{
280       if not luatexhyphen then
281         dofile(assert(kpse.find_file("luatex-hyphen.lua")))
282       end
283       luatexhyphen.loadlanguage("\luatexluaescapestring{#1}",
284         \the\language)}%
285   \fi
286   \fi
287 \fi
288 }
289 \def\hyphenrules#1{%
290   \expandafter\ifx\csname l@#1\endcsname\@undefined
291     \@nolanerr{#1}%
292   \else
293     \bb@patterns{#1}%
294     \languageshorthands{none}%
295     \expandafter\ifx\csname #1hyphenmins\endcsname\relax
296       \set@hyphenmins\tw@thr@@\relax
297     \else
298       \expandafter\expandafter\expandafter\set@hyphenmins
299         \csname #1hyphenmins\endcsname\relax
300     \fi
301   \fi
302 }
303 \def\endhyphenrules{}
304 \def\providehyphenmins#1#2{%
305   \expandafter\ifx\csname #1hyphenmins\endcsname\relax
306     \@namedef{#1hyphenmins}{#2}%
307   \fi}
308 \def\set@hyphenmins#1#2{\lefthyphenmin#1\righthyphenmin#2}
309 \def\LdfInit{%
310   \chardef\atcatcode=\catcode`\@
311   \catcode`\@=11\relax
312   \input babel.def\relax
313   \catcode`\@=\atcatcode \let\atcatcode\relax
314 \LdfInit}

```

```

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

```

```

365  \else
366    \expandafter\chardef\csname 1@#1\endcsname\last@language
367    \wlog{\string\l0#1=\string\language\the\last@language}
368    \expandafter\let\csname #1hyphenmins\expandafter\endcsname
369      \csname\languagename hyphenmins\endcsname
370  \fi
371 }
372 \def\process@language#1 #2 #3/{%
373   \expandafter\addlanguage\csname 1@#1\endcsname
374   \expandafter\language\csname 1@#1\endcsname
375   \def\languagename{#1}%

```

In the Lua<sub>TEX</sub> case, we have to decide whether 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 <sub>TEX</sub> macros in order to survive the format dumping-loading cycle, which would not be as straightforward using Lua objects.

```

376  \ifx\directlua\@undefined
377    \global\toks8\expandafter{\the\toks8#1, }%
378  \else
379    \directlua{
380      if not luatexhyphen then
381        dofile(assert(kpse.find_file("luatex-hyphen.lua")))
382      end
383      processnow = (tex.language == 0) or
384        (luatexhyphen.lookupname("\luatexluascapestring{#1}") == nil)}%
385  \ifnum0=\directlua{tex.sprint(processnow and "0" or "1")}\relax
386    \global\toks8\expandafter{\the\toks8#1, }%
387    \global\@namedef{lu@texhyphen@loaded@\the\language}{}%
388  \fi
389 \fi
390 \begingroup
391   \bblob@get@enc#1:\@@@
392   \ifx\bblob@hyph@enc\@empty
393   \else
394     \fontencoding{\bblob@hyph@enc}\selectfont
395   \fi
396   \lefthyphenmin\m@ne

```

Conditionally input the patterns file.

```

397  \ifx\directlua\@undefined
398    \input #2\relax
399  \else
400    \ifnum0=\directlua{tex.sprint(processnow and "0" or "1")}\relax
401      \input #2\relax
402    \fi
403  \fi
404  \ifnum\lefthyphenmin=\m@ne
405  \else
406    \expandafter\xdef\csname #1hyphenmins\endcsname{%

```

```

407      \the\lefthyphenmin\the\righthypenmin}%
408      \fi
409  \endgroup
410  \ifnum\the\language=\z@%
411    \expandafter\ifx\csname #1hyphenmins\endcsname\relax
412      \set@hyphenmins\tw@\thr@@\relax
413    \else
414      \expandafter\expandafter\expandafter\set@hyphenmins
415      \csname #1hyphenmins\endcsname
416    \fi
417    \the\toks@
418  \fi
419  \toks@{}%
420  \def\bb@tempa{\#3}%
421  \ifx\bb@tempa\empty
422  \else
423    \ifx\bb@tempa\space
424  \else
425    \ifx\directlua{\undefined}
426      \input #3\relax
427    \else
428      \ifnum0=\directlua{tex.sprint(processnow and "0" or "1")}\relax
429        \input #3\relax
430      \fi
431      \directlua{processnow = nil}%
432    \fi
433    \fi
434  \fi
435 }
436 \def\bb@get@enc#1:#2\@@@{%
437   \def\bb@tempa{\#1}%
438   \def\bb@tempb{\#2}%
439   \ifx\bb@tempb\empty
440     \let\bb@hyph@enc\empty
441   \else
442     \bb@get@enc#2\@@@
443     \edef\bb@hyph@enc{\bb@tempa}%
444   \fi}
445 \openin1 = language.dat
446 \ifeof1
447   \message{I couldn't find the file language.dat,\space
448             I will try the file hyphen.tex}
449   \input hyphen.tex\relax
450   \def\l@english{0}%
451   \def\languagename{english}%
452 \else
453   \last@language\m@ne
454   \loop

```

```

455   \endlinechar\m@ne
456   \read1 to \bb@line
457   \endlinechar`^^M
458   \ifx\bb@line\empty
459   \else
460     \edef\bb@line{\bb@line\space/}%
461     \expandafter\process@line\bb@line
462     \ifx\bb@defaultlanguage\@undefined
463       \let\bb@defaultlanguage\language
464     \fi
465   \fi
466   \iftrue \csname fi\endcsname
467   \csname if\ifeof1 false\else true\fi\endcsname
468   \repeat
469   \language=0
470   \let\language\bb@defaultlanguage
471   \let\bb@defaultlanguage\@undefined
472 \fi
473 \closein1
474 \let\process@language\@undefined
475 \let\process@synonym\@undefined
476 \let\process@line\@undefined
477 \let\bb@tempa\@undefined
478 \let\bb@tempb\@undefined
479 \let\bb@eq@\@undefined
480 \let\bb@line\@undefined
481 \let\bb@get@enc\@undefined
482 \ifx\addto@hook\@undefined
483 \else
484   \expandafter\addto@hook\expandafter\everyjob\expandafter{%
485     \expandafter\typeout\expandafter{\the\toks8 loaded.}}
486 \fi
487 ⟨/hyphen⟩

```