

# The `luainputenc` package

Elie Roux

`elie.roux@telecom-bretagne.eu`

2010/05/10 v0.97

## Abstract

Input encoding management for Lua $\TeX$ . For an introduction on this package (among others), please refer to `luatex-reference.pdf`.

## Contents

<b>1</b>	<b>Documentation</b>	<b>1</b>
1.1	Introduction . . . . .	1
1.2	Overview of 8-bit mode . . . . .	2
1.3	Overview of UTF-8 mode . . . . .	2
1.3.1	legacy mode . . . . .	2
1.3.2	unicode font mode . . . . .	3
1.3.3	mixed mode . . . . .	3
<b>2</b>	<b>Accessing the encoding in lua</b>	<b>3</b>
<b>3</b>	<b>Files</b>	<b>3</b>
3.1	<code>inputenc.sty</code> patch . . . . .	3
3.2	<code>luainputenc.sty</code> . . . . .	4
3.3	<code>lutf8.def</code> . . . . .	10
3.4	<code>lutf8x.def</code> . . . . .	12
3.5	<code>luainputenc.lua</code> . . . . .	15

## 1 Documentation

### 1.1 Introduction

One the the most interesting new features of Lua $\TeX$  is the fact that it is (like Omega/Aleph) not limited to 256 characters, and can now understand Unicode. The problem is that it does not read input the way older engines (like pdf $\TeX$ ) do, and thus `inputenc` is totally broken with Lua $\TeX$ . This package aims at replacing `inputenc` for Lua $\TeX$ , by adapting the way Lua $\TeX$  handles input, and the way `inputenc` handles UTF-8. This package has two very distinct modes: 8-bit and UTF-8.

## 1.2 Overview of 8-bit mode

This package **does not** map 8-bit encodings to utf8. It allows LuaTeX to read 8-bit characters, by converting each byte into a unicode character with the same character number. The resulting unicode characters are not true UTF-8, they are what we will call “fake UTF-8”. For example the byte 225 will be converted into the unicode character with number 225 (two bytes long). It will be true UTF-8 only if the encoding is latin1.

Here is how it works: the 8-bit encodings are converted into fake UTF-8, so that the corresponding tokens are chars with the good numbers. Then (like `inputenc`) it reads the char numbers, and converts it into LICR (L<sup>A</sup>T<sub>E</sub>X Internal Character Representation), with the font encoding.

In LuaTeX version 0.43, a new callback called `process_output_buffer`, this callback allows to make LuaTeX write 8-bit instead of UTF-8, so the behaviour is the same as pdfTeX as this level. For versions prior to 0.43 though, we need to do more tricky things, described in the next paragraph. This machinery is disabled for LuaTeX version 0.43 and superior, so you can keep the default behaviour, which will be compatible with pdfTeX in most cases, but you can consider the machinery obsolete.

For these old versions, `luainputenc` only changes the input behaviour, it does not change the output behaviour (when files are written for example). The consequence is that files will still be written by LuaTeX in UTF-8 (fake UTF-8 in this case), even if the asked input encoding is a 8-bit encoding. In most cases it’s not a problem, as most files will be written in LICR, meaning ASCII, which is both 8-bit and UTF-8. The problem comes when characters with a number > 128 are written in a 8-bit encoding. This may happen if you use `\protect` in a section for example. In these cases, LuaTeX will write fake UTF-8, and try to read 8-bit encoding, so it will get confused.

The proposed solution is to unactivate the input conversion when we read certain files or extensions. This package should work with no change for most documents, but if you cook your own aux files with an unknown extension, you may have to force the package to read some files in UTF-8 instead of 8-bit. See comments in the `.sty` file to know the useful commands.

## 1.3 Overview of UTF-8 mode

The behaviour of `inputenc` in utf8 mode is to read the input byte by byte, and decide if the character we are in is 1, 2, 3 or 4 bytes long, and then read other bytes accordingly. This behaviour fails with LuaTeX because it reads input character by character (characters do not have a fixed number of bytes in unicode). The result is thus an error.

All characters recognized by TeX are active characters, that correspond to a LICR macro. Then `inputenc` reads the `*.dfu` files that contain the correspondance between these LICR macros and a character number in the fonts for different font encodings (T1, OT1, etc.).

### 1.3.1 legacy mode

`luainputenc` can get this behaviour (we will call it *legacy mode*, but another difference implied by the fact that LuaTeX can read more than 256 characters is that fonts can also have more than 256 characters. LuaTeX can thus read unicode fonts. If we want to use unicode fonts (OTF for example), we can’t use the *legacy mode* anymore, as it would mean that we would

have to rewrite a specially long `unicode.dfu` file, and it would be totally inefficient, as for instance `é` (unicode character number 233) would be mapped to `\'e`, and then mapped back to `\char 233`.

### 1.3.2 unicode font mode

To fix this, the most simple solution is to deactivate all activated characters, thus typing `é` will directly call `\char 233` in the unicode fonts, and produce a `é`. We will call this behaviour the *unicode font mode*. To enable this mode, you can use the option `unactivate` in `luainputenc`, and you must use the font encoding `EU2` provided by the `euenc` package. See documentation of `euenc` package for more details about `EU2`. To use this mode with `EU2`, you must be able to open OTF fonts. A simple way to do so it by using the package `luaotfload`.

### 1.3.3 mixed mode

But the *unicode font mode* has a strong limitation (that will certainly dissapear with time): it cannot use non-unicode fonts. If you want to mix unicode fonts and old fonts, you'll have to use the *mixed mode*. In this mode you can type some parts of your document in *legacy mode* and some in *unicode font mode*. The reason why we chose not to integrate this choice in the *legacy mode* is that we wanted to have a mode that preserved most of the backward compatibility, to safely compile old documents; the *mixed mode* introduces new things that may break old documents. To get the *mixed mode*, you must pass the option `lutf8x` to `luainputenc`. This mode is the most experimental.

## 2 Accessing the encoding in lua

In order to access the encoding and the package option in lua, two variables are set: `luainputenc.package_option` contains the option passed to the package, and `luainputenc.encoding` that contains the encoding (defaults to `utf8`, and is `utf8` even with the options `unactivate`, `utf8x`, etc.).

## 3 Files

This package contains a `.sty` file for both `LATEX` and Plain, a patch for `inputenc` to use `luainputenc` so that you can process old documents without changing anything, and the lua functions.

### 3.1 inputenc.sty patch

A good thing would be to patch `inputenc` to load `luainputenc` instead, so that you don't have to change your documents to load `luainputenc` especially. The `LATEX` team is extremely conservative and does not want this patch applied (maybe we will find a solution later). Here is a patch for `inputenc.sty`:

```

1
2   \ifnum\@tempcnta<'#2\relax
3       \advance\@tempcnta\@ne

```

```

4   \repeat}
5 +
6 +\begingroup\expandafter\expandafter\expandafter\endgroup
7 +\expandafter\ifx\csname XeTeXversion\endcsname\relax\else
8 + \RequirePackage{xetex-inputenc}
9 + \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{xetex-inputenc}}
10 + \ProcessOptions*
11 + \expandafter\endinput
12 +\fi
13 +\begingroup\expandafter\expandafter\expandafter\endgroup
14 +\expandafter\ifx\csname directlua\endcsname\relax\else
15 + \RequirePackage{luainputenc}
16 + \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{luainputenc}}
17 + \ProcessOptions*
18 + \expandafter\endinput
19 +\fi
20 +
21 \ProcessOptions
22 \endinput
23 %%
24

```

### 3.2 luainputenc.sty

This file has some code from `inputenc.sty`, but also provides new options, and new macros to convert from 8-bit to fake UTF-8.

```

25 %
26 %% This file was adapted from inputenc.sty, which copyright is:
27 %% Copyright 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004
28 %% 2005 2006 The LaTeX3 Project.
29 %%
30 %% inputenc.sty is under the lppl version 1.3c or later, and can be
31 %% found in the base LaTeX system.
32 %%
33 %% The lppl can be found at http://www.latex-project.org/lppl.txt
34 %%
35 %% The changes to inputenc.sty are Copyright 2009 Elie Roux, and are
36 %% under the CC0 license.
37 %%
38 %% The changes are LuaTeX support.
39 %%
40 %% This file is distributed under the CC0 license, with clause 6 of the
41 %% lppl as additional restrictions.
42

```

First we check if we are called with LuaTeX, (pdf)TeX or XeTeX. If we are called with pdfTeX, we default to `inputenc`, and to `xetex-inputenc` if we are called with XeTeX. We also remap the new options to `utf8` in these cases.

```

43
44 \RequirePackage{ifluatex}

```

```

45 \RequirePackage{ifxetex}
46
47 \ifxetex
48   \RequirePackage{xetex-inputenc}
49   \DeclareOption{unactivate}{\PassOptionsToPackage{utf8}{xetex-inputenc}}
50   \DeclareOption{lutf8}{\PassOptionsToPackage{utf8}{xetex-inputenc}}
51   \DeclareOption{lutf8x}{\PassOptionsToPackage{utf8}{xetex-inputenc}}
52   \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{xetex-inputenc}}
53   \ProcessOptions*
54   \expandafter\endinput
55 \fi
56
57 \ifluatex\else
58   \RequirePackage{inputenc}
59   \DeclareOption{unactivate}{\PassOptionsToPackage{utf8}{inputenc}}
60   \DeclareOption{lutf8}{\PassOptionsToPackage{utf8}{inputenc}}
61   \DeclareOption{lutf8x}{\PassOptionsToPackage{utf8}{inputenc}}
62   \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{inputenc}}
63   \ProcessOptions*
64   \expandafter\endinput
65 \fi
66

```

Here we know we are called with LuaT<sub>E</sub>X. We first require luatextra, then we load the lua file.

```

67
68 \RequirePackage{luatextra}
69
70 \luatexUseModule{luainputenc}
71

```

Here is some code from inputenc.

```

72
73 \def\DeclareInputMath#1{%
74   \@inpenc@test
75   \bgroup
76     \uccode'\~#1%
77     \uppercase{%
78       \egroup
79       \def~%
80     }%
81 }
82 \def\DeclareInputText#1#2{%
83   \def\reserved@a##1 ${}%
84   \def\reserved@b{#2}%
85   \ifcat_\expandafter\reserved@a\meaning\reserved@b$ $_%
86     \DeclareInputMath{#1}{#2}%
87   \else
88     \DeclareInputMath{#1}{\IeC{#2}}%
89   \fi

```

```

90 }
91 \def\IeC{%
92   \ifx\protect\@typeset@protect
93     \expandafter\@firstofone
94   \else
95     \noexpand\IeC
96   \fi
97 }

```

We changed a little the behaviour of this macro: we removed `\@inpenc@loop\^^?\^^ff`, because it made no sense in UTF-8 mode. We will call this line for 8-bit encodings.

Note that the code has been changed for `\endlinechar`, because in new versions (from v0.43) of LuaTeX the value cannot exceed 127. Thus, with the old version of `luainputenc`, when trying to add 10000, it fails silently, and when 10000 is subtracted, the new value is -1, resulting in no end of lines at all in the document.

```

98
99 \def\inputencoding#1{%
100   \the\inpenc@prehook
101   \gdef\@inpenc@test{\global\let\@inpenc@test\relax}%
102   \edef\@inpenc@undefined{\noexpand\@inpenc@undefined@{#1}}%
103   \edef\inputencodingname{#1}%
104   \@inpenc@loop\^^A\^^H%
105   \@inpenc@loop\^^K\^^K%
106   \@inpenc@loop\^^N\^^_%
107   \xdef\saved@endlinechar{\the\endlinechar }%
108   \endlinechar=-1
109   \xdef\saved@space@catcode{\the\catcode'\ }%
110   \catcode'\ 9\relax
111   \input{#1.def}%
112   \endlinechar=\saved@endlinechar{}%
113   \catcode'\ \saved@space@catcode\relax
114   \ifx\@inpenc@test\relax\else
115     \PackageWarning{inputenc}%
116       {No characters defined\MessageBreak
117        by input encoding change to '#1'\MessageBreak}%
118   \fi
119   \the\inpenc@posthook
120   \luadirect{luainputenc.set_option([[#1]])}
121 }
122 \newtoks\inpenc@prehook
123 \newtoks\inpenc@posthook
124 \def\@inpenc@undefined@#1{\PackageError{inputenc}%
125   {Keyboard character used is undefined\MessageBreak
126   in inputencoding '#1'}%
127   {You need to provide a definition with
128   \noexpand\DeclareInputText\MessageBreak or
129   \noexpand\DeclareInputMath before using this key.}}%
130 \def\@inpenc@loop#1#2{%
131   \@tempcnta'#1\relax
132   \loop

```

```

133 \catcode\@tempcnta\active
134 \bgroup
135 \uccode'\~\@tempcnta
136 \uppercase{%
137 \egroup
138 \let~\inpenc@undefined
139 }%
140 \ifnum\@tempcnta<#2\relax
141 \advance\@tempcnta\@ne
142 \repeat}
143

```

Here we declare our options. Note that we remap `utf8` to `lutf8`, because we use `outlutf8.def` instead of `inputenc's utf8.def`.

```

144
145 \DeclareOption{utf8}{%
146 \inputencoding{lutf8}%
147 }
148
149 \DeclareOption{lutf8}{%
150 \inputencoding{lutf8}%
151 }
152
153 \DeclareOption{utf8x}{%
154 \inputencoding{lutf8}%
155 }
156
157 \DeclareOption{lutf8x}{%
158 \inputencoding{lutf8x}%
159 }
160

```

For the `unactivate` option, for *unicode font mode*, we just don't do anything.

```

161
162 \DeclareOption{unactivate}{%
163 \edef\inputencodingname{unactivate}%
164 \luadirect{luainputenc.set_option([[unactivate]])}
165 }
166

```

All other options are 8-bit encodings, so we activate the translation into fake UTF-8, and we execute the loop we removes from `\inputencoding`.

```

167
168 \DeclareOption*{%
169 \lIE@activate %
170 \@inpenc@loop\^^?\^^ff%
171 \inputencoding{\CurrentOption}%
172 }
173

```

The rest of the file is only the machinery for LuaTeX versions without the callback `process_output_buffer`, so it will be deprecated after TeXLive 2009, you are not advised to use it.

```

174
175 \ifnum\luatexversion>42
176
177   \newcommand*\lIE@activate}[0]{%
178     \luadirect{luainputenc.register_callbacks()}%
179   }
180
181 \else
182

```

`\lIE@setstarted` and `\lIE@setstopped` are called when the fake UTF-8 translation must be activated or deactivated. You can call them several successive times. They are called very often, even if the package is not activated (for example if it's loaded with the `utf8` option), but they act only if the package is activated.

```

183
184 \newcommand*\lIE@setstarted[0]{%
185   \ifnum\lIE@activated=1 %
186     \luadirect{luainputenc.setstarted()}%
187   \fi %
188 }
189
190 \newcommand*\lIE@setstopped[0]{%
191   \ifnum\lIE@activated=1 %
192     \luadirect{luainputenc.setstopped()}%
193   \fi %
194 }
195

```

The following 5 macros are made to declare a file that will have to be read in fake UTF-8 and not in 8-bit. These files are the ones that will be generated by T<sub>E</sub>X. In **no way** this means you can include true UTF-8 files, it means that you can include files that have been written by LuaT<sub>E</sub>X with `luainputenc`, which means files in fake UTF-8. The macros are very simple, when you call them with a file name (the same as the one you will use with “input”), it will read it with or without the fake UTF-8 translation. This package includes a whole bunch of extensions that will be read in fake UTF-8, so the occasions to use these macros will be rare, but if you use them, please report it to the package maintainer.

`\lIE@SetUtfFile` If you call this macro with a file name, each time you will input this file, it will be read in fake UTF-8. You can call it with a file that you generate with LuaT<sub>E</sub>X and that you want to include.

```

196
197 \newcommand*\lIE@SetUtfFile[1]{%
198   \luadirect{luainputenc.set_unicode_file([[#1]])}%
199 }
200

```



`\lIE@SetNonUtfFile` Same as the previous macro, except that the file will be read as 8-bit. This macro is useful if there is an exception in an extension (see further comments).

```
201
202 \newcommand*\lIE@SetNonUtfFile[1]{%
203   \luadirect{luainputenc.set_non_unicode_file([[#1]])}%
204 }
205
```

`\lIE@UnsetFile` This macro gives a file the default behaviour of its extension.

```
206
207 \newcommand*\lIE@UnsetFile[1]{%
208   \luadirect{luainputenc.unset_file([[#1]])}%
209 }
210
```

`\lIE@SetUtfExt` You can tell luainputenc to treat all files with a particular extension in a certain way. The way the file extension is checked is to compare the four last characters of the filename. So if your extension has only three letters, you must include the preceding dot. This macro tells luainputenc to read all files from an extension in fake UTF-8.

```
211
212 \newcommand*\lIE@SetUtfExt[1]{%
213   \luadirect{luainputenc.set_unicode_extention([[#1]])}%
214 }
215
```

`\lIE@SetUtfExt` Same as before, but the files will be read in 8-bit.

```
216
217 \newcommand*\lIE@SetNonUtfExt[1]{
218   \luadirect{luainputenc.set_non_unicode_extention([[#1]])}
219 }
220
```

`\lIE@InputUtfFile` This macro inputs a file in fake UTF-8. It has the “feature” to unset the behaviour on the file you will call, so to be safe, you must call them with files for which the behaviour has not been set.

```
221
222
223 \newcommand*\lIE@InputUtfFile[1]{%
224   \lIE@SetUtfFile{#1}%
225   \input #1%
226   \lIE@UnsetFile{#1}%
227 }
228
```

`\lIE@InputNonUtfFile` Same as before, but to read a file as 8-bit.

```
229
230 \newcommand*\lIE@InputNonUtfFile[1]{%
231   \lIE@SetNonUtfFile{#1}%

```

```

232 \input #1%
233 \lIE@UnsetFile{#1}%
234 }
235

```

Two definitions to put the previous two macros in the user space.

```

236
237 \newcommand*\InputUtfFile[1]{%
238 \lIE@InputUtfFile{#1}%
239 }
240
241 \newcommand*\InputNonUtfFile[1]{%
242 \lIE@InputNonUtfFile{#1}%
243 }
244
245 \newcount\lIE@activated
246
247 \newcommand*\lIE@activate}[0]{%
248 \lIE@activated=1 %
249 \lIE@setstarted %
250 }
251
252 \newcommand*\lIE@FromInputenc}[1]{%
253 \ifnum\lIE@activated=0 %
254 \lIE@activate %
255 \fi%
256 }
257
258 \fi
259
260 \ProcessOptions*
261

```

### 3.3 lutf8.def

```

262 %% This file was adapted from utf8.def, which copyright is:
263 %% Copyright 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003
264 %% 2004 2005 2006 The LaTeX3 Project.
265 %%
266 %% utf8.def is under the lppl version 1.3c or later, and can be found
267 %% in the base LaTeX system.
268 %%
269 %% The lppl can be found at http://www.latex-project.org/lppl.txt
270 %%
271 %% The changes to utf8.def are Copyright 2009 Elie Roux, and are under
272 %% the CC0 license.
273 %%
274 %% The changes are LuaTeX support.
275 %%
276 %% This file is distributed under the CC0 license, with clause 6 of the

```

```

277 %% lpp1 as additional restrictions.
278

```

Most of the file is taken from `utf8.def`, the main changes are commented. A lot of code was removed, especially the codes that analysed the unicode characters byte by byte.

```

279
280
281 \ProvidesFile{lutf8.def}
282   [2010/05/10 v0.97 UTF-8 support for luainputenc]
283
284 \makeatletter
285 \catcode'\ \saved@space@catcode
286
287 \@inpenc@test
288
289 \ifx\@begindocumenthook\undefined
290   \makeatother
291   \endinput \fi
292

```

This function is changed a lot. Its aim is to map the character (first argument) to a macro (second argument). In `utf8.def` it was complicated as unicode was analyzed byte by byte. With LuaTeX it is extremely simple, we just have to activate the character, and call a traditional `\DeclareInputTeXt`.

```

293
294 \gdef\DeclareUnicodeCharacter#1#2{%
295   \@tempcnta"#1%
296   \catcode\@tempcnta\active %
297   \DeclareInputText{\the\@tempcnta}{#2}%
298 }
299
300 \@onlypreamble\DeclareUnicodeCharacter
301
302 \def\cdp@elt#1#2#3#4{%
303   \wlog{Now handling font encoding #1 ...}%
304   \lowercase{%
305     \InputIfFileExists{#1enc.dfu}}%
306     {\wlog{... processing UTF-8 mapping file for font encoding
307       #1}%
308       \catcode'\ 9\relax}%
309     {\wlog{... no UTF-8 mapping file for font encoding #1}}%
310 }
311 \cdp@list
312
313 \def\DeclareFontEncoding@#1#2#3{%
314   \expandafter %
315   \ifx\csname T@#1\endcsname\relax %
316     \def\cdp@elt{\noexpand\cdp@elt}%
317     \xdef\cdp@list{\cdp@list\cdp@elt{#1}%
318       {\default@family}{\default@series}%
319       {\default@shape}}%

```

```

320 \expandafter\let\csname#1-cmd\endcsname\@changed@cmd %
321 \begingroup %
322 \wlog{Now handling font encoding #1 ...}%
323 \lowercase{%
324 \InputIfFileExists{#1enc.dfu}}%
325 {\wlog{... processing UTF-8 mapping file for font encoding #1}}%
326 {\wlog{... no UTF-8 mapping file for font encoding #1}}%
327 \endgroup
328 \else
329 \@font@info{Redefining font encoding #1}%
330 \fi
331 \global\@namedef{T@#1}{#2}%
332 \global\@namedef{M@#1}{\default@M#3}%
333 \xdef\LastDeclaredEncoding{#1}%
334 }
335
336 \DeclareUnicodeCharacter{00A9}{\textcopyright}
337 \DeclareUnicodeCharacter{00AA}{\textordfeminine}
338 \DeclareUnicodeCharacter{00AE}{\textregistered}
339 \DeclareUnicodeCharacter{00BA}{\textordmasculine}
340 \DeclareUnicodeCharacter{02C6}{\textasciicircum}
341 \DeclareUnicodeCharacter{02DC}{\textasciitilde}
342 \DeclareUnicodeCharacter{200C}{\textcompwordmark}
343 \DeclareUnicodeCharacter{2026}{\textellipsis}
344 \DeclareUnicodeCharacter{2122}{\texttrademark}
345 \DeclareUnicodeCharacter{2423}{\textvisiblespace}
346

```

### 3.4 lutf8x.def

```

347 %% This file was adapted from utf8.def, which copyright is:
348 %% Copyright 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003
349 %% 2004 2005 2006 The LaTeX3 Project.
350 %%
351 %% utf8.def is under the lppl version 1.3c or later, and can be found
352 %% in the base LaTeX system.
353 %%
354 %% The lppl can be found at http://www.latex-project.org/lppl.txt
355 %%
356 %% The changes to utf8.def are Copyright 2009 Elie Roux, and are under
357 %% the CC0 license.
358 %%
359 %% The changes are LuaTeX support.
360 %%
361 %% This file is distributed under the CC0 license, with clause 6 of the
362 %% lppl as additional restrictions.
363

```

This file is mostly the code from `lutf.def`, but it adds mechanisms to pass from *legacy mode* to *unicode font mode*. The trick is to put in a lua table all characters that are activated by the *legacy mode*, and to unactivate them when we switch to *unicode font mode*. This is

made (almost) entirely in lua. The difficult part is the changes in `\DeclareFontEncoding`.

```

364
365 \ProvidesFile{lua8x.def}
366 [2010/05/10 v0.97 UTF-8 support for luainputenc]
367
368 \makeatletter
369 \catcode'\ \saved@space@catcode
370
371 \@inpenc@test
372
373 \ifx\@begindocumenthook\@undefined
374 \makeatother
375 \endinput \fi
376

```

We change it a little to add the activated character in the lua table.

```

377
378 \gdef\DeclareUnicodeCharacter#1#2{%
379 \tempcnta"#1%
380 \luadirect{luainputenc.declare_character('\the\tempcnta')}%
381 \catcode\tempcnta\active %
382 \DeclareInputText{\the\tempcnta}{#2}%
383 }
384
385 \@onlypreamble\DeclareUnicodeCharacter
386
387 \def\cdp@elt#1#2#3#4{%
388 \wlog{Now handling font encoding #1 ...}%
389 \lowercase{%
390 \InputIfFileExists{#1enc.dfu}}%
391 {\wlog{... processing UTF-8 mapping file for font encoding
392 #1}%
393 \catcode'\ 9\relax}%
394 {\wlog{... no UTF-8 mapping file for font encoding #1}}%
395 }
396 \cdp@list
397

```

The macros to change from/to *legacy mode* to/from *unicode font mode*.

```

398
399 \def\lIE@ActivateUnicodeCatcodes{%
400 \luadirect{luainputenc.activate_characters()}%
401 }
402
403 \def\lIE@DeactivateUnicodeCatcodes{%
404 \luadirect{luainputenc.deactivate_characters()}%
405 }
406
407 \def\lIE@CharactersActivated{%
408 \luadirect{luainputenc.force_characters_activated()}
409 }

```

```

410
411 \edef\lIE@EU{EU2}
412

```

We add some code to automatically activate or unactivate characters according to the encoding changes. Note that we override `\@enc@update`, which may pose some problems if a package of yours does it too. Fortunately this package is the only one that does it in  $\text{\TeX}$ Live.

```

413
414 \def\DeclareFontEncoding@#1#2#3{%
415   \edef\lIE@test{#1}%
416   \ifx\lIE@test\lIE@EU %
417     \ifx\LastDeclaredEncoding\lIE@EU\else %
418       \lIE@CharactersActivated %
419       \lIE@DesactivateUnicodeCatcodes %
420     \fi
421   \gdef\@enc@update{%
422     \edef\lIE@test{#1}%
423     \ifx\f@encoding\lIE@EU %
424       \lIE@DesactivateUnicodeCatcodes %
425     \else %
426       \lIE@ActivateUnicodeCatcodes %
427     \fi
428     \expandafter\let\csname\cf@encoding-cmd\endcsname\@changed@cmd
429     \expandafter\let\csname\f@encoding-cmd\endcsname\@current@cmd
430     \default@T
431     \csname T@\f@encoding\endcsname
432     \csname D@\f@encoding\endcsname
433     \let\enc@update\relax
434     \let\cf@encoding\f@encoding
435   }
436 \else %
437   \expandafter %
438   \ifx\csname T@#1\endcsname\relax %
439     \def\cdp@elt{\noexpand\cdp@elt}%
440     \xdef\cdp@list{\cdp@list\cdp@elt{#1}%
441                   {\default@family}{\default@series}%
442                   {\default@shape}}%
443   \expandafter\let\csname#1-cmd\endcsname\@changed@cmd %
444   \begingroup %
445     \wlog{Now handling font encoding #1 ...}%
446     \lowercase{%
447       \InputIfFileExists{#1enc.dfu}}%
448     {\wlog{... processing UTF-8 mapping file for font encoding #1}}%
449     {\wlog{... no UTF-8 mapping file for font encoding #1}}%
450   \endgroup
451 \else
452   \@font@info{Redeclaring font encoding #1}%
453 \fi
454 \fi %
455 \global\@namedef{T@#1}{#2}%

```

```

456 \global\@namedef{M@#1}{\default@M#3}%
457 \xdef\LastDeclaredEncoding{#1}%
458 }
459
460 \DeclareUnicodeCharacter{00A9}{\textcopyright}
461 \DeclareUnicodeCharacter{00AA}{\textordfeminine}
462 \DeclareUnicodeCharacter{00AE}{\textregistered}
463 \DeclareUnicodeCharacter{00BA}{\textordmasculine}
464 \DeclareUnicodeCharacter{02C6}{\textasciicircum}
465 \DeclareUnicodeCharacter{02DC}{\textasciitilde}
466 \DeclareUnicodeCharacter{200C}{\textcompwordmark}
467 \DeclareUnicodeCharacter{2026}{\textellipsis}
468 \DeclareUnicodeCharacter{2122}{\texttrademark}
469 \DeclareUnicodeCharacter{2423}{\textvisiblespace}
470

```

### 3.5 luainputenc.lua

First the `inputenc` module is registered as a Lua<sub>TeX</sub> module, with some informations.

```

471
472 module('luainputenc', package.seeall)
473
474 luainputenc.module = {
475     name      = "luainputenc",
476     version   = 0.97,
477     date      = "2010/05/10",
478     description = "Lua simple inputenc package.",
479     author    = "Elie Roux",
480     copyright  = "Elie Roux",
481     license    = "CC0",
482 }
483
484 luatexbase.provides_module(luainputenc.module)
485
486 local format = string.format
487
488 luainputenc.log = luainputenc.log or function(...)
489     luatexbase.module_log('luainputenc', format(...))
490 end
491

```

We keep the option and the true encoding in two variables.

```

492
493 luainputenc.encoding = "utf8"
494 luainputenc.package_option = nil
495
496 function luainputenc.set_option(option)
497     luainputenc.package_option = option
498     if option == "lutf8" or option == "lutf8x" or option == "utf8x" or option == "unactivate" then
499         luainputenc.encoding = "utf8"
500     else

```

```

501     luainputenc.encoding = option
502 end
503 end
504
    Some local declarations.

505
506 local char, utfchar, byte, format, gsub, utfbyte, utfgsub =
507 string.char, unicode.utf8.char, string.byte, string.format, string.gsub, unicode.utf8.byte, unicod
508
    The function to transform a 8-bit character in the corresponding fake UTF-8 character.

509
510 function luainputenc.byte_to_utf(ch)
511     return utfchar(byte(ch))
512 end
513
    The function that will be registered in the process_input_buffer callback when needed.

514
515 function luainputenc.fake_utf_read(buf)
516     return gsub(buf,"(.)", luainputenc.byte_to_utf)
517 end
518
    The function to transform a fake utf8 character in the corresponding 8-bit character.

519
520 function luainputenc.utf_to_byte(ch)
521     return char(utfbyte(ch))
522 end
523
    The function that will be registered in the process_output_buffer callback if it exists.

524
525 function luainputenc.fake_utf_write(buf)
526     return utfgsub(buf,"(.)", luainputenc.utf_to_byte)
527 end
528
    Here we register the two callbacks, and the behaviour is the same as in pdfTeX. The
    next part of the file is only the machinery for LuaTeX versions without the callback
    process_output_buffer, so it will be deprecated after TeXLive 2009, you are not advised
    to use it.

529
530 if tex.luaTeXversion > 42 then
531
532     function luainputenc.register_callbacks()
533         luatexbase.add_to_callback('process_output_buffer', luainputenc.fake_utf_write, 'luainputenc.fake_utf_write')
534         luatexbase.add_to_callback('process_input_buffer', luainputenc.fake_utf_read, 'luainputenc.fake_utf_read')
535     end

```



```

536
537 else
538

```

`start()` and `stop()` are the functions that register or unregister the function in the callback. When the function is registered, LuaTeX reads the input in fake UTF-8.

```

539     local started, stopped = 1, 0
540
541     luainputenc.state = stopped
542
543     function luainputenc.setstate(state)
544         if state == luainputenc.state then
545             return
546         elseif state == started then
547             luainputenc.start()
548         else
549             luainputenc.stop()
550         end
551     end
552
553     function luainputenc.setstarted()
554         luainputenc.setstate(started)
555     end
556
557     function luainputenc.setstopped()
558         luainputenc.setstate(stopped)
559     end
560
561     function luainputenc.start()
562         luatexbase.add_to_callback('process_input_buffer', luainputenc.fake_utf_read,
563             'luainputenc.fake_utf_read')
564         luainputenc.state = started
565         if luainputenc.callback_registered == 0 then
566             luainputenc.register_callback()
567         end
568     end
569
570     function luainputenc.stop()
571         luatexbase.remove_from_callback('process_input_buffer', 'luainputenc.fake_utf_read')
572         luainputenc.state = stopped
573         return
574     end
575
576

```

Here is a list of all file extensions for which we consider that the files have been written by LuaTeX, and thus must be read in fake UTF-8. I may have forgotten things in the list. If you find a new extension, please report the maintainer.

```

577
578     luainputenc.unicode_extensions = {

```

```

579     ['.aux'] = 1, -- basic files
580     ['.toc'] = 1,
581     ['.gls'] = 1,
582     ['.ind'] = 1,
583     ['.idx'] = 1,
584     ['.vrb'] = 1, -- beamer and powerdot
585     ['.nav'] = 1, -- other beamer extentions
586     ['.sol'] = 1,
587     ['.qsl'] = 1,
588     ['.snm'] = 1,
589     ['.pgn'] = 1, -- pagereference
590     ['.cpg'] = 1, -- AlProTeX
591     ['.pst'] = 1, -- pst-tree
592     ['.tmp'] = 1, -- sauerj/collect
593     ['.sym'] = 1, -- listofsymbols
594     ['.sub'] = 1, -- listofsymbols
595     ['.lof'] = 1, -- preprint
596     ['.lot'] = 1, -- preprint
597     ['.mtc1'] = 1, -- minitoc
598     ['.ovr'] = 1, -- thumbss
599     ['.fff'] = 1, -- endplate
600     ['.sbb'] = 1, -- splitbib
601     ['.bbl'] = 1, -- latex
602     ['.ain'] = 1, -- authorindex
603     ['.abb'] = 1, -- juraabbrev
604     ['.ent'] = 1, -- endnotes
605     ['.end'] = 1, -- fn2end
606     ['.thm'] = 1, -- ntheorem
607     ['.xtr'] = 1, -- extract
608     ['.han'] = 1, -- linguho
609     ['.bnd'] = 1, -- bibref
610     ['.bbl'] = 1, -- bibref
611     ['.col'] = 1, -- mwrite
612     ['.ttt'] = 1, -- endfloat
613     ['.fax'] = 1, -- lettre
614     ['.tns'] = 1, -- lettre
615     ['.odt'] = 1, -- lettre
616     ['.etq'] = 1, -- lettre
617     ['.emd'] = 1, -- poemscol
618     ['.emx'] = 1, -- poemscol
619     ['.ctn'] = 1, -- poemscol
620     ['.hst'] = 1, -- vhistory
621     ['.acr'] = 1, -- crosswrd
622     ['.dwn'] = 1, -- crosswrd
623     ['.ttc'] = 1, -- talk
624     -- ['.txt'] = 1, -- coverpage, but not sure it's safe to include it...
625     ['.eve'] = 1, -- calend0
626     ['.scn'] = 1, -- cwebmac
627 }
628

```

The code to define a specific behaviour for certain files.

```
629
630     luainputenc.unicode_files = {}
631
632     luainputenc.non_unicode_files = {}
633
634     function luainputenc.set_unicode_file(filename)
635         if luainputenc.non_unicode_files[filename] == 1 then
636             luainputenc.non_unicode_files[filename] = nil
637         end
638         luainputenc.unicode_files[filename] = 1
639     end
640
641     function luainputenc.set_non_unicode_file(filename)
642         if luainputenc.unicode_files[filename] == 1 then
643             luainputenc.unicode_files[filename] = nil
644         end
645         luainputenc.non_unicode_files[filename] = 1
646     end
647
648     function luainputenc.set_unicode_extention(ext)
649         luainputenc.unicode_extention[ext] = 1
650     end
651
652     function luainputenc.set_non_unicode_extention(ext)
653         if luainputenc.unicode_extentions[ext] == 1 then
654             luainputenc.unicode_extentions[ext] = nil
655         end
656     end
657
658     function luainputenc.unset_file(filename)
659         if luainputenc.unicode_files[filename] == 1 then
660             luainputenc.unicode_files[filename] = nil
661         elseif luainputenc.non_unicode_files[filename] == 1 then
662             luainputenc.non_unicode_files[filename] = nil
663         end
664     end
665
666     local unicode, non_unicode = stopped, started
667
668     function luainputenc.find_state(filename)
669         if luainputenc.unicode_files[filename] == 1 then
670             return unicode
671         elseif luainputenc.non_unicode_files[filename] == 1 then
672             return non_unicode
673         else
674             local ext = filename:sub(-4)
675             if luainputenc.unicode_extentions[ext] == 1 then
676                 return unicode
677             else
```

```

678             return non_unicode
679         end
680     end
681 end
682

```

We register the functions to stop or start the fake UTF-8 translation in the appropriate callbacks if necessary.

```

683
684     function luainputenc.pre_read_file(env)
685         if not env.path then
686             return
687         end
688         local currentstate = luainputenc.state
689         luainputenc.setstate(luainputenc.find_state(env.filename))
690         env.previousstate = currentstate
691     end
692
693     function luainputenc.close(env)
694         luainputenc.setstate(env.previousstate)
695     end
696
697     luainputenc.callback_registered = 0
698
699     function luainputenc.register_callback()
700         if luainputenc.callback_registered == 0 then
701             luatexbase.add_to_callback('pre_read_file', luainputenc.pre_read_file,
702                 'luainputenc.pre_read_file')
703             luatexbase.add_to_callback('file_close', luainputenc.close, 'luainputenc.close')
704             luainputenc.callback_registered = 1
705         end
706     end
707
708 end
709

```

Finally we provide some functions to activate or deactivate the catcodes of the non-ASCII characters.

```

710
711
712 luainputenc.activated_characters = {}
713 luainputenc.characters_are_activated = false
714
715 function luainputenc.declare_character(c)
716     luainputenc.activated_characters[tonumber(c)] = true
717 end
718
719 function luainputenc.force_characters_activated ()
720     luainputenc.characters_are_activated = true
721 end

```

```

722
723 function luainputenc.activate_characters()
724     if not luainputenc.characters_are_activated then
725         for n, _ in pairs(luainputenc.activated_characters) do
726             tex.sprint(string.format('\catcode %d\active',n))
727         end
728         luainputenc.characters_are_activated = true
729     end
730 end
731
732 function luainputenc.desactivate_characters()
733     if luainputenc.characters_are_activated then
734         for n, _ in pairs(luainputenc.activated_characters) do
735             tex.sprint(string.format('\catcode %d=11',n))
736         end
737         luainputenc.characters_are_activated = false
738     end
739 end
740

```