The luainputenc package

Elie Roux
`elie.roux@telecom-bretagne.eu`

2009/09/23 v0.94

Abstract

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

Contents

1 Documentation

1.1 Introduction

One the the most interesting new features of LuaTeX 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 pdfTeX) do, and thus inputenc is totally broken with LuaTeX. This package aims at replacing inputenc for LuaTeX, by adapting the way LuaTeX 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 (LATEX Internal Character Representation), with the font encoding.

In LuaTEX version 0.43, a new callback called `process_output_buffer`, this callbacks 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 ouput 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 extentions. This package should work with no change for most documents, but if you cook your own aux files with an unknown extention, 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 desactivate 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 this package too. See section 2.5 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 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.

2.1 `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:

```

   \ifnum\@tempcnta<`#2\relax
      \advance\@tempcnta\@ne
   \repeat}
+
+\begingroup\expandafter\expandafter\expandafter\endgroup
+\expandafter\ifx\csname XeTeXversion\endcsname\relax\else
+  \RequirePackage{xetex-inputenc}
+  \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{xetex-inputenc}}
+  \ProcessOptions*
+  \expandafter\endinput
```

```
+\fi
+\begingroup\expandafter\expandafter\expandafter\endgroup
+\expandafter\ifx\csname directlua\endcsname\relax\else
+  \RequirePackage{luainputenc}
+  \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{luainputenc}}
+  \ProcessOptions*
+  \expandafter\endinput
+\fi
+
 \ProcessOptions
 \endinput
 %%

```

2.2 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.

```
%
%% This file was adapted from inputenc.sty, which copyright is:
%% Copyright 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004
%% 2005 2006 The LaTeX3 Project.
%%
%% inputenc.sty is under the lppl version 1.3c or later, and can be
%% found in the base LaTeX system.
%%
%% The lppl can be found at http://www.latex-project.org/lppl.txt
%%
%% The changes to inputenc.sty are Copyright 2009 Elie Roux, and are
%% under the CC0 license.
%%
%% The changes are LuaTeX support.
%%
%% This file is distributed under the CC0 license, with clause 6 of the
%% lppl as additional restrictions.

```

First we check if we are called with LuaTEX, (pdf)TEXor 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.

```

\RequirePackage{ifluatex}
\RequirePackage{ifxetex}

\ifxetex
  \RequirePackage{xetex-inputenc}
  \DeclareOption{unactivate}{\PassOptionsToPackage{utf8}{xetex-inputenc}}
  \DeclareOption{lutf8}{\PassOptionsToPackage{utf8}{xetex-inputenc}}
  \DeclareOption{lutf8x}{\PassOptionsToPackage{utf8}{xetex-inputenc}}
  \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{xetex-inputenc}}
  \ProcessOptions*
  \expandafter\endinput
\fi

\ifluatex\else
  \RequirePackage{inputenc}
  \DeclareOption{unactivate}{\PassOptionsToPackage{utf8}{inputenc}}
  \DeclareOption{lutf8}{\PassOptionsToPackage{utf8}{inputenc}}
  \DeclareOption{lutf8x}{\PassOptionsToPackage{utf8}{inputenc}}
  \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{inputenc}}
  \ProcessOptions*
  \expandafter\endinput
\fi

```

Here we know we are called with LuaTEX. We first require luatextra, then we load the `lua` file.

```

\RequirePackage{luatextra}

\luatexUseModule{luainputenc}

```

Here is some code from inputenc.

```

\def\DeclareInputMath#1{%
   \@inpenc@test
   \bgroup
      \uccode`\~#1%
      \uppercase{%
   \egroup
      \def~%
   }%
}
\def\DeclareInputText#1#2{%
   \def\reserved@a##1 ${}%
   \def\reserved@b{#2}%
   \ifcat_\expandafter\reserved@a\meaning\reserved@b$ $_%
      \DeclareInputMath{#1}{#2}%
   \else
      \DeclareInputMath{#1}{\IeC{#2}}%
   \fi
}
\def\IeC{%
  \ifx\protect\@typeset@protect
    \expandafter\@firstofone
  \else
    \noexpand\IeC
  \fi
}
```

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.

```

\def\inputencoding#1{%
  \the\inpenc@prehook
  \gdef\@inpenc@test{\global\let\@inpenc@test\relax}%
  \edef\@inpenc@undefined{\noexpand\@inpenc@undefined@{#1}}%
  \edef\inputencodingname{#1}%
  \@inpenc@loop\^^A\^^H%
  \@inpenc@loop\^^K\^^K%
  \@inpenc@loop\^^N\^^_%
  \advance\endlinechar\@M
  \xdef\saved@space@catcode{\the\catcode`\ }%
  \catcode`\ 9\relax
  \input{#1.def}%
  \advance\endlinechar-\@M
  \catcode`\ \saved@space@catcode\relax
  \ifx\@inpenc@test\relax\else
    \PackageWarning{inputenc}%
             {No characters defined\MessageBreak
              by input encoding change to `#1'\MessageBreak}%
  \fi
  \the\inpenc@posthook
}
\newtoks\inpenc@prehook
\newtoks\inpenc@posthook
\def\@inpenc@undefined@#1{\PackageError{inputenc}%
        {Keyboard character used is undefined\MessageBreak
         in inputencoding `#1'}%
       {You need to provide a  definition with
        \noexpand\DeclareInputText\MessageBreak or
        \noexpand\DeclareInputMath before using this key.}}%
\def\@inpenc@loop#1#2{%
  \@tempcnta`#1\relax
  \loop
    \catcode\@tempcnta\active
    \bgroup
      \uccode`\~\@tempcnta
      \uppercase{%
    \egroup
         \let~\inpenc@undefined
      }%
  \ifnum\@tempcnta<`#2\relax
     \advance\@tempcnta\@ne
  \repeat}

```

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

```

```

```
\DeclareOption{utf8}{%
  \inputencoding{lutf8}%
}

\DeclareOption{lutf8}{%
  \inputencoding{lutf8}%
}

\DeclareOption{utf8x}{%
  \inputencoding{lutf8}%
}

\DeclareOption{lutf8x}{%
  \inputencoding{lutf8x}%
}

```

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

```

\DeclareOption{unactivate}{%
  \edef\inputencodingname{unactivate}%
}

```

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`.

```

\DeclareOption*{%
  \lIE@activate %
  \@inpenc@loop\^^?\^^ff%
  \inputencoding{\CurrentOption}%
}

```

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.

```

\ifnum\luatexversion>42

    \newcommand*{\lIE@activate}[0]{%
      \luadirect{luainputenc.register_callbacks()}%
    }

\else

```

`\lIE@setstarted` and `\lIE@setstopped` are called when the fake UTF-8 translation must be activated or desactivated. 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.

```
\newcommand*\lIE@setstarted[0]{%
  \ifnum\lIE@activated=1 %
    \luadirect{luainputenc.setstarted()}%
  \fi %
}

\newcommand*\lIE@setstopped[0]{%
  \ifnum\lIE@activated=1 %
    \luadirect{luainputenc.setstopped()}%
  \fi %
}

```

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 TEX. In **no way** this means you can include true UTF-8 files, it means that you can include files that have been written by LuaTEX 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 extentions 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 LuaTEX and that you want to include.

```

\newcommand*\lIE@SetUtfFile[1]{%
  \luadirect{luainputenc.set_unicode_file([[#1]])}%
}

```

`\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 extention (see further comments).

```

\newcommand*\lIE@SetNonUtfFile[1]{%
  \luadirect{luainputenc.set_non_unicode_file([[#1]])}%
}

```

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

```

\newcommand*\lIE@UnsetFile[1]{%
  \luadirect{luainputenc.unset_file([[#1]])}%
}

```

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

```

\newcommand*\lIE@SetUtfExt[1]{%
  \luadirect{luainputenc.set_unicode_extention([[#1]])}%
}

```

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

```

\newcommand*\lIE@SetNonUtfExt[1]{
  \luadirect{luainputenc.set_non_unicode_extention([[#1]])}
}

```

`\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.

```


\newcommand*\lIE@InputUtfFile[1]{%
  \lIE@SetUtfFile{#1}%
  \input #1%
  \lIE@UnsetFile{#1}%
}

```

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

```

\newcommand*\lIE@InputNonUtfFile[1]{%
  \lIE@SetNonUtfFile{#1}%
  \input #1%
  \lIE@UnsetFile{#1}%
}

```

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

```

\newcommand*\InputUtfFile[1]{%
  \lIE@InputUtfFile{#1}%
}

\newcommand*\InputNonUtfFile[1]{%
  \lIE@InputNonUtfFile{#1}%
}

\newcount\lIE@activated

```

```
\newcommand*{\lIE@activate}[0]{%
  \lIE@activated=1 %
  \lIE@setstarted %
}

\newcommand*{\lIE@FromInputenc}[1]{%
  \ifnum\lIE@activated=0 %
    \lIE@activate %
  \fi%
}

\fi

\ProcessOptions*

```

2.3 lutf8.def

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

```

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.

```


\ProvidesFile{lutf8.def}
  [2009/09/23 v0.94 UTF-8 support for luainputenc]

\makeatletter
\catcode`\ \saved@space@catcode

\@inpenc@test

\ifx\@begindocumenthook\@undefined
  \makeatother
  \endinput \fi
```

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`.

```

\gdef\DeclareUnicodeCharacter#1#2{%
 \@tempcnta"#1%
 \catcode\@tempcnta\active %
 \DeclareInputText{\the\@tempcnta}{#2}%
}

\@onlypreamble\DeclareUnicodeCharacter

\def\cdp@elt#1#2#3#4{%
  \wlog{Now handling font encoding #1 ...}%
  \lowercase{%
      \InputIfFileExists{#1enc.dfu}}%
         {\wlog{... processing UTF-8 mapping file for font encoding
                #1}%
          \catcode`\ 9\relax}%
        {\wlog{... no UTF-8 mapping file for font encoding #1}}%
}
\cdp@list

\def\DeclareFontEncoding@#1#2#3{%
  \expandafter %
  \ifx\csname T@#1\endcsname\relax %
    \def\cdp@elt{\noexpand\cdp@elt}%
    \xdef\cdp@list{\cdp@list\cdp@elt{#1}%
                    {\default@family}{\default@series}%
                    {\default@shape}}%
    \expandafter\let\csname#1-cmd\endcsname\@changed@cmd %
    \begingroup %
      \wlog{Now handling font encoding #1 ...}%
      \lowercase{%
        \InputIfFileExists{#1enc.dfu}}%
           {\wlog{... processing UTF-8 mapping file for font encoding #1}}%
           {\wlog{... no UTF-8 mapping file for font encoding #1}}%
    \endgroup
  \else
     \@font@info{Redeclaring font encoding #1}%
  \fi
  \global\@namedef{T@#1}{#2}%
  \global\@namedef{M@#1}{\default@M#3}%
  \xdef\LastDeclaredEncoding{#1}%
}

\DeclareUnicodeCharacter{00A9}{\textcopyright}
\DeclareUnicodeCharacter{00AA}{\textordfeminine}
```

```
\DeclareUnicodeCharacter{00AE}{\textregistered}
\DeclareUnicodeCharacter{00BA}{\textordmasculine}
\DeclareUnicodeCharacter{02C6}{\textasciicircum}
\DeclareUnicodeCharacter{02DC}{\textasciitilde}
\DeclareUnicodeCharacter{200C}{\textcompwordmark}
\DeclareUnicodeCharacter{2026}{\textellipsis}
\DeclareUnicodeCharacter{2122}{\texttrademark}
\DeclareUnicodeCharacter{2423}{\textvisiblespace}

```

2.4 lutf8x.def

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

```

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`.

```

\ProvidesFile{lutf8x.def}
  [2009/09/23 v0.94 UTF-8 support for luainputenc]

\makeatletter
\catcode`\ \saved@space@catcode

\@inpenc@test

\ifx\@begindocumenthook\@undefined
 \makeatother
 \endinput \fi

```

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

```

\gdef\DeclareUnicodeCharacter#1#2{%
 \@tempcnta"#1%
 \luadirect{luainputenc.declare_character('\the\@tempcnta')}%
 \catcode\@tempcnta\active %
 \DeclareInputText{\the\@tempcnta}{#2}%
}

\@onlypreamble\DeclareUnicodeCharacter

\def\cdp@elt#1#2#3#4{%
  \wlog{Now handling font encoding #1 ...}%
  \lowercase{%
      \InputIfFileExists{#1enc.dfu}}%
         {\wlog{... processing UTF-8 mapping file for font encoding
                #1}%
          \catcode`\ 9\relax}%
        {\wlog{... no UTF-8 mapping file for font encoding #1}}%
}
\cdp@list

```

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

```

\def\lIE@ActivateUnicodeCatcodes{%
\luadirect{luainputenc.activate_characters()}%
}

\def\lIE@DesactivateUnicodeCatcodes{%
\luadirect{luainputenc.desactivate_characters()}%
}

\def\lIE@CharactersActivated{%
\luadirect{luainputenc.force_characters_activated()}
}

\edef\lIE@EU{EU2}

```

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 TEXLive.

```

\def\DeclareFontEncoding@#1#2#3{%
  \edef\lIE@test{#1}%
  \ifx\lIE@test\lIE@EU %
    \ifx\LastDeclaredEncoding\lIE@EU\else %
      \lIE@CharactersActivated %
      \lIE@DesactivateUnicodeCatcodes %
    \fi
    \gdef\@@enc@update{%
      \edef\lIE@test{#1}%
      \ifx\f@encoding\lIE@EU %
      \lIE@DesactivateUnicodeCatcodes %
     \else %
      \lIE@ActivateUnicodeCatcodes %
    \fi
    \expandafter\let\csname\cf@encoding-cmd\endcsname\@changed@cmd
    \expandafter\let\csname\f@encoding-cmd\endcsname\@current@cmd
    \default@T
    \csname T@\f@encoding\endcsname
    \csname D@\f@encoding\endcsname
    \let\enc@update\relax
    \let\cf@encoding\f@encoding
  }
 \else %
   \expandafter %
   \ifx\csname T@#1\endcsname\relax %
     \def\cdp@elt{\noexpand\cdp@elt}%
     \xdef\cdp@list{\cdp@list\cdp@elt{#1}%
                    {\default@family}{\default@series}%
                    {\default@shape}}%
     \expandafter\let\csname#1-cmd\endcsname\@changed@cmd %
     \begingroup %
       \wlog{Now handling font encoding #1 ...}%
       \lowercase{%
         \InputIfFileExists{#1enc.dfu}}%
            {\wlog{... processing UTF-8 mapping file for font encoding #1}}%
            {\wlog{... no UTF-8 mapping file for font encoding #1}}%
     \endgroup
   \else
      \@font@info{Redeclaring font encoding #1}%
   \fi
 \fi %
 \global\@namedef{T@#1}{#2}%
 \global\@namedef{M@#1}{\default@M#3}%
 \xdef\LastDeclaredEncoding{#1}%
}

\DeclareUnicodeCharacter{00A9}{\textcopyright}
\DeclareUnicodeCharacter{00AA}{\textordfeminine}
\DeclareUnicodeCharacter{00AE}{\textregistered}
\DeclareUnicodeCharacter{00BA}{\textordmasculine}
\DeclareUnicodeCharacter{02C6}{\textasciicircum}
\DeclareUnicodeCharacter{02DC}{\textasciitilde}
\DeclareUnicodeCharacter{200C}{\textcompwordmark}
\DeclareUnicodeCharacter{2026}{\textellipsis}
\DeclareUnicodeCharacter{2122}{\texttrademark}
\DeclareUnicodeCharacter{2423}{\textvisiblespace}

```

2.5 eu2enc.def

This file is extremely short. It just declares the encoding, with the default font. The default font here is lmr, which means that LaTeX will read `eu2lmr.fd`. The problem is that all unicode fonts are OTF fonts, so `eu2lmr.fd` will call OTF fonts. Thus, to use `EU2`, you need to be able to read OTF fonts. The package luaotfload is a good choice to be able to do so.

```

\ProvidesFile{eu2enc.def}[2009/09/23 v0.1 a unicode font encoding for LuaTeX.]
\DeclareFontEncoding{EU2}{}{}
\DeclareErrorFont{EU2}{lmr}{m}{n}{10}
\DeclareFontSubstitution{EU2}{lmr}{m}{n}

```

2.6 eu2lmr.fd

This file simply describes the default (lmr) font of the `EU2` encoding. It loads the otf fonts with some default features enabled. This file may change, don't rely on it too much.

```

\ProvidesFile{eu2lmr.fd}
    [2009/09/23 v0.2 Font defs for Latin Modern for LuaTeX's EU2 encoding]
\DeclareFontFamily{EU2}{lmr}{}
\DeclareFontShape{EU2}{lmr}{m}{n}%
     {<-5.5>    "lmroman5-regular:+tlig;+tsub;+liga;+rlig;"
  <5.5-6.5> "lmroman6-regular:+tlig;+tsub;+liga;+rlig;"
      <6.5-7.5> "lmroman7-regular:+tlig;+tsub;+liga;+rlig;"
  <7.5-8.5> "lmroman8-regular:+tlig;+tsub;+liga;+rlig;"
      <8.5-9.5> "lmroman9-regular:+tlig;+tsub;+liga;+rlig;"
  <9.5-11>  "lmroman10-regular:+tlig;+tsub;+liga;+rlig;"
      <11-15>   "lmroman12-regular:+tlig;+tsub;+liga;+rlig;"
      <15->     "lmroman17-regular:+tlig;+tsub;+liga;+rlig;"
      }{}
\DeclareFontShape{EU2}{lmr}{m}{sl}%
     {<-8.5>    "lmroman8-oblique:+tlig;+tsub;+liga;+rlig;"
  <8.5-9.5> "lmroman9-oblique:+tlig;+tsub;+liga;+rlig;"
      <9.5-11>  "lmroman10-oblique:+tlig;+tsub;+liga;+rlig;"
  <11-15>    "lmroman12-oblique:+tlig;+tsub;+liga;+rlig;"
      <15->     "lmroman17-oblique:+tlig;+tsub;+liga;+rlig;"
      }{}
\DeclareFontShape{EU2}{lmr}{m}{it}%
     {<-7.5>    "lmroman7-italic:+tlig;+tsub;+liga;+rlig;"
      <7.5-8.5> "lmroman8-italic:+tlig;+tsub;+liga;+rlig;"
  <8.5-9.5> "lmroman9-italic:+tlig;+tsub;+liga;+rlig;"
      <9.5-11>  "lmroman10-italic:+tlig;+tsub;+liga;+rlig;"
  <11->     "lmroman12-italic:+tlig;+tsub;+liga;+rlig;"
      }{}
\DeclareFontShape{EU2}{lmr}{m}{sc}%
     {<-> "lmroman10-capsregular:+tlig;+tsub;+liga;+rlig;"}{}
%
% Is this the right 'shape'?:
```

```
\DeclareFontShape{EU2}{lmr}{m}{scsl}%
     {<-> "lmroman10-capsoblique:+tlig;+tsub;+liga;+rlig;"}{}
%%%%%%% bold series
\DeclareFontShape{EU2}{lmr}{b}{n}
     {<-> "lmroman10-demi:+tlig;+tsub;+liga;+rlig;"}{}
\DeclareFontShape{EU2}{lmr}{b}{sl}
     {<-> "lmroman10-demioblique:+tlig;+tsub;+liga;+rlig;"}{}
%%%%%%%% bold extended series
\DeclareFontShape{EU2}{lmr}{bx}{n}
     {<-5.5>    "lmroman5-bold:+tlig;+tsub;+liga;+rlig;"
  <5.5-6.5> "lmroman6-bold:+tlig;+tsub;+liga;+rlig;"
      <6.5-7.5> "lmroman7-bold:+tlig;+tsub;+liga;+rlig;"
  <7.5-8.5> "lmroman8-bold:+tlig;+tsub;+liga;+rlig;"
      <8.5-9.5> "lmroman9-bold:+tlig;+tsub;+liga;+rlig;"
  <9.5-11>  "lmroman10-bold:+tlig;+tsub;+liga;+rlig;"
      <11->     "lmroman12-bold:+tlig;+tsub;+liga;+rlig;"
      }{}
\DeclareFontShape{EU2}{lmr}{bx}{it}
     {<-> "lmroman10-bolditalic:+tlig;+tsub;+liga;+rlig;"}{}
\DeclareFontShape{EU2}{lmr}{bx}{sl}
     {<-> "lmroman10-boldoblique:+tlig;+tsub;+liga;+rlig;"}{}

```

2.7 luainputenc.lua

First the `inputenc` module is registered as a LuaTEX module, with some informations.

```

luainputenc = { }

luainputenc.module = {
    name          = "luainputenc",
    version       =  0.94,
    date          = "2009/09/23",
    description   = "Lua simple inputenc package.",
    author        = "Elie Roux",
    copyright     = "Elie Roux",
    license       = "CC0",
}

luatextra.provides_module(luainputenc.module)

local format = string.format

luainputenc.log = luainputenc.log or function(...)
  luatextra.module_log('luainputenc', format(...))
end

local char, utfchar, byte, format, gsub, utfbyte, utfgsub =
string.char, unicode.utf8.char, string.byte, string.format, string.gsub, unicode.utf8.byte, unicod

```

The function to transform a 8-bit character in the corresponding fake UTF-8 character.

```

function luainputenc.byte_to_utf(ch)
    return utfchar(byte(ch))
end

```

The function that will be registered in the process_input_buffer callback when needed.

```

function luainputenc.fake_utf_read(buf)
    return gsub(buf,"(.)", luainputenc.byte_to_utf)
end

```

The function to transform a fake utf8 character in the corresponding 8-bit character.

```

function luainputenc.utf_to_byte(ch)
    return char(utfbyte(ch))
end

```

The function that will be registered in the process_output_buffer callback if it exists.

```

function luainputenc.fake_utf_write(buf)
    return utfgsub(buf,"(.)", luainputenc.utf_to_byte)
end

```

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.

```

if tex.luatexversion > 42 then

    function luainputenc.register_callbacks()
        callback.add('process_output_buffer', luainputenc.fake_utf_write, 'luainputenc.fake_utf_wr
        callback.add('process_input_buffer', luainputenc.fake_utf_read, 'luainputenc.fake_utf_read
    end

else

```

`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.

```

    local started, stopped = 1, 0

    luainputenc.state = stopped

```

```
function luainputenc.setstate(state)
    if state == luainputenc.state then
        return
    elseif state == started then
        luainputenc.start()
    else
        luainputenc.stop()
    end
end

function luainputenc.setstarted()
    luainputenc.setstate(started)
end

function luainputenc.setstopped()
    luainputenc.setstate(stopped)
end

function luainputenc.start()
    callback.add('process_input_buffer', luainputenc.fake_utf_read,
        'luainputenc.fake_utf_read')
    luainputenc.state = started
    if luainputenc.callback_registered == 0 then
        luainputenc.register_callback()
    end
end

function luainputenc.stop()
    callback.remove('process_input_buffer', 'luainputenc.fake_utf_read')
    luainputenc.state = stopped
    return
end

```

Here is a list of all file extentions 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 extention, please report the maintainer.

```

luainputenc.unicode_extentions = {
  ['.aux'] = 1, -- basic files
  ['.toc'] = 1,
  ['.gls'] = 1,
  ['.ind'] = 1,
  ['.idx'] = 1,
  ['.vrb'] = 1, -- beamer and powerdot
  ['.nav'] = 1, -- other beamer extentions
  ['.sol'] = 1,
  ['.qsl'] = 1,
  ['.snm'] = 1,
  ['.pgn'] = 1, -- pagereference
```

```
    ['.cpg'] = 1, -- AlProTeX
    ['.pst'] = 1, -- pst-tree
    ['.tmp'] = 1, -- sauerj/collect
    ['.sym'] = 1, -- listofsymbols
    ['.sub'] = 1, -- listofsymbols
    ['.lof'] = 1, -- preprint
    ['.lot'] = 1, -- preprint
    ['mtc1'] = 1, -- minitoc
    ['.ovr'] = 1, -- thumbss
    ['.fff'] = 1, -- endplate
    ['.sbb'] = 1, -- splitbib
    ['.bbl'] = 1, -- latex
    ['.ain'] = 1, -- authorindex
    ['.abb'] = 1, -- juraabbrev
    ['.ent'] = 1, -- endnotes
    ['.end'] = 1, -- fn2end
    ['.thm'] = 1, -- ntheorem
    ['.xtr'] = 1, -- extract
    ['.han'] = 1, -- linguho
    ['.bnd'] = 1, -- bibref
    ['.bbl'] = 1, -- bibref
    ['.col'] = 1, -- mwrite
    ['.ttt'] = 1, -- endfloat
    ['.fax'] = 1, -- lettre
    ['.tns'] = 1, -- lettre
    ['.odt'] = 1, -- lettre
    ['.etq'] = 1, -- lettre
    ['.emd'] = 1, -- poemscol
    ['.emx'] = 1, -- poemscol
    ['.ctn'] = 1, -- poemscol
    ['.hst'] = 1, -- vhistory
    ['.acr'] = 1, -- crosswrd
    ['.dwn'] = 1, -- crosswrd
    ['.ttc'] = 1, -- talk
    -- ['.txt'] = 1, -- coverpage, but not sure it's safe to include it...
    ['.eve'] = 1, -- calend0
    ['.scn'] = 1, -- cwebmac
    }

```

The code to define a specific behaviour for certain files.

```

  luainputenc.unicode_files = {}

  luainputenc.non_unicode_files = {}

  function luainputenc.set_unicode_file(filename)
      if luainputenc.non_unicode_files[filename] == 1 then
          luainputenc.non_unicode_files[filename] = nil
      end
      luainputenc.unicode_files[filename] = 1
```

```
end

function luainputenc.set_non_unicode_file(filename)
    if luainputenc.unicode_files[filename] == 1 then
        luainputenc.unicode_files[filename] = nil
    end
    luainputenc.non_unicode_files[filename] = 1
end

function luainputenc.set_unicode_extention(ext)
    luainputenc.unicode_extention[ext] = 1
end

function luainputenc.set_non_unicode_extention(ext)
    if luainputenc.unicode_extentions[ext] == 1 then
        luainputenc.unicode_extentions[ext] = nil
    end
end

function luainputenc.unset_file(filename)
    if luainputenc.unicode_files[filename] == 1 then
        luainputenc.unicode_files[filename] = nil
    elseif luainputenc.non_unicode_files[filename] == 1 then
        luainputenc.non_unicode_files[filename] = nil
    end
end

local unicode, non_unicode = stopped, started

function luainputenc.find_state(filename)
    if luainputenc.unicode_files[filename] == 1 then
        return unicode
    elseif luainputenc.non_unicode_files[filename] == 1 then
        return non_unicode
    else
        local ext = filename:sub(-4)
         if luainputenc.unicode_extentions[ext] == 1 then
              return unicode
         else
            return non_unicode
         end
    end
end

```

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

```

function luainputenc.pre_read_file(env)
    if not env.path then
```

```
            return
        end
        local currentstate = luainputenc.state
        luainputenc.setstate(luainputenc.find_state(env.filename))
        env.previousstate = currentstate
    end

    function luainputenc.close(env)
        luainputenc.setstate(env.previousstate)
    end

    luainputenc.callback_registered = 0

    function luainputenc.register_callback()
        if luainputenc.callback_registered == 0 then
            callback.add('pre_read_file', luainputenc.pre_read_file,
                'luainputenc.pre_read_file')
            callback.add('file_close', luainputenc.close, 'luainputenc.close')
            luainputenc.callback_registered = 1
        end
    end

end

```

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

```


luainputenc.activated_characters = {}
luainputenc.characters_are_activated = false

function luainputenc.declare_character(c)
    luainputenc.activated_characters[tonumber(c)] = true
end

function luainputenc.force_characters_activated ()
    luainputenc.characters_are_activated = true
end

function luainputenc.activate_characters()
    if not luainputenc.characters_are_activated then
        for n, _ in pairs(luainputenc.activated_characters) do
            tex.sprint(string.format('\\catcode %d\\active',n))
        end
        luainputenc.characters_are_activated = true
    end
end

function luainputenc.desactivate_characters()
```

```
    if luainputenc.characters_are_activated then
        for n, _ in pairs(luainputenc.activated_characters) do
            tex.sprint(string.format('\\catcode %d=11',n))
        end
        luainputenc.characters_are_activated = false
    end
end

```