Second version of encTEX: UTF-8 support

Petr Olšák

Czech Technical University in Prague
Email: petr@olsak.net

Abstract: The UTF-8 encoding keeps the standard ASCII characters unchanged and encodes the accented letters of our alphabets in two bytes. The standard 8bit TEX is not ready for the UTF-8 input because it have to manage the single character as two tokens. It means you cannot set the `\catcode`, `\uccode`, etc. to these single characters and you cannot do `\futurelet` of the next character in normal sense. The second version of my encTEX solves these problems.

The encTEX is full backward compatible with the original TEX. It adds ten new primitives by which you can set or read the conversion tables used by input processor of TEX or used during output to the terminal, log and `\write` files.

The second version gives possibility to convert the multi-byte sequences to one byte or to control sequence. You can implement up to 256 UTF-8 codes as one byte and unlimited number of other UTF-8 codes as a control sequence. All internals in 8bit TEX are working in the same way as if "normal one byte encoding" of input files is used.

I think that the UTF-8 encoding will be used more common. In such situation, there is no another way than to modify the input processor of TEX otherwise the 8bit TEX will dead in short time.

1 What is encTEX?

EncTEX is a TEX extension which allows re-encoding of input stream on input processor of TEX (before tokenization) and backward re-encoding of output stream during `\write` and output to the terminal and log. It is implemented as the patch to the change file `tex.ch`. The patches are ready for web2c distribution on [1] and (may be) encTEX becomes as a standard web2c extension like mikTEX. Try to use the `-enc` option on command line to test if your TEX is equipped with this extension. If not, you can get and apply the patches and rebuild TEX binaries. The patches affect TEX, eTEX, pdfTEX and pdfeTEX programs. All these programs will dispose of this extension.

First version of encTEX was released in 1997. This version was able to do only byte to byte conversion by affecting the TEX's internal *xord* and *xchr* vectors. EncTEX introduced three primitives in its first version: `\xordcode` (reads or sets the values of *xord* vector for input re-encoding), `\xchrcode` (reads or sets

the values of *xchr* vector for output re-encoding) and `\xprncode` (reads or sets the values of newly introduced *xprn* vector which controls the "print-ability" of characters—it controls the possibility of the character conversion to `^^ab` form on output side). See my article [2] for more details.

The first version of encTEX was not widely used because the TCX tables was renovated in web2c distribution immediately after encTEX was released. Roughly speaking, the TCX tables do the same work as first version of my encTEX but less flexible. There was no reason to combine the TCX tables with encTEX.

The second version of encTEX was designed and prepared by me in December 2002 and released in January 2003. This version introduces seven more primitives in order to user can control the multi-byte input re-encoding and reverse output re-encoding. Groups of bytes on input stream can be converted to one byte or to control sequence. The conversion is done before tokenization but the control sequence generated by this conversion is not re-tokenized again and token processor does not go to "ignoring spaces" state after such control sequence. The backward conversion during `\write` allows you to convert one byte or control sequence to the original group of bytes.

The second version of encTEX is backward compatible with the first one, of course. The detail documentation is available on [1]. The very nice on-line html documentation written by David Nečas (Yeti) is available on [5]

2 Motivation

I am maintainer of a `csplain` format—the basic part of the CSTEX package (for Czech and Slovak users). The `csplain` is similar as very known plainTEX format (by Don Knuth, [4]). Moreover, `csplain` solves the processing of all letters from Czech and Slovak alphabets. It means that the CS-fonts (encoded by ISO-8859-2) is used by default instead of Computer Modern fonts, the hyphenation tables for Czech and Slovak languages are inputted in the same encoding and all Czech and Slovak letters have to be treated as single non-composite symbols. These symbols have `\catcode` set to 11 (letter), thus they can be used in control sequences too.

Czech and Slovak alphabets are encoded by many mutual incompatible standards and pseudo-standards in various operating systems and operating environments. All these encodings have to be converted to internal ISO-8859-2 in `csplain` at input processor level and they have to be converted back to the input encoding during `\write`, terminal and log output. Only this rule keeps the independence of the TEX processing on the operating system.

Note: if the source text of the Czech or Slovak document is transported from one environment to another, the re-encoding to the standard of the target environment is done automatically or by user manually. The main principle is that the Czech and Slovak characters in source text have to be displayed correctly by used operating environment before it is processed by `csplain`.

I have created the `cstrip` test in 1998 [3]. You can verify if you are really using the `csplain` format by this test. This test verifies if TEX's input processor

is set correctly depending on your operating environment: all Czech and Slovak characters have to be mapped into ISO-8859-2 and they have to be written back to the input encoding on terminal, log and `\write` files. The `^^ab` form is not permitted for Czech and Slovak letters.

We were able to set the input processor properly for `csplain` in old TEX distributions. For example emTEX have used TCP tables. On the other hand the web2c distribution have had disabled its TCX tables in 1997 thus users was not able to implement the `csplain` format correctly in operating environments where different encoding of our alphabets from ISO-8859-2 were used. This was the main motivation of encTEX extension of TEX.

Now, the new encoding standard derived from UNICODE and named UTF-8 is used very often. The non-ASCII characters are encoded in two or more bytes here. If this encoding standard is used in our operating environment then we need to be able to set multi-byte conversion in input processor of TEX. There is no other way to carry out the `cstrip` test. This was my motivation of second version of the encTEX.

3 Multi-byte re-encoding

The detail documentation is included in encTEX package. Thus, only a short overview of the principles is presented here.

Second version of encTEX introduces seven new TEX primitives to define and control re-encoding between multi-byte input/output and TEX internal representation. These are:

- `\mubyte` and `\endmubyte` defining the conversions,
- `\mubytein`, an integer register controlling input conversion,
- `\mubyteout`, an integer register controlling output conversion,
- `\mubytelog`, an integer register controlling output to terminal and log file,
- `\specialout`, an integer register controlling `\special` argument treatment, and
- `\noconvert`, a primitive suppressing output conversion.

The default values of all the new registers are such that encTEX behaves compatibly with unmodified TEX (incidentally, it means zeroes).

You can set the conversion table by the couple `\mubyte` and `\endmubyte`. Examples:

```
\mubyte ^^c1   ^^c3^^81\endmubyte % Aacute
\mubyte ^^c4   ^^c3^^84\endmubyte % Adieresis
...
```

It means that for example the group of two bytes `^^c3^^81` will be converted to one byte `^^c1` (if `\mubytein` is positive) and this byte is converted back to byte sequence `^^c3^^81` during `\write` (if `\mubyteout` is positive) and to log and terminal (if `\mubytelog` is positive).

If your operating environment uses UTF-8 encoding then the two bytes `^^c3^^81` are displayed as Á. You can do the "normal things" with this character in your text editor:

```
\catcode ‘Á=11  \def\myÁsequence{...}
...
\def\run{\futurelet \next \dotest}
\def\dotest{\ifx \next Á...}
\run Áha
...
\uccode‘Á=‘Á \lccode‘Á=‘á \sfcode‘Á=999
...
```

This behavior is very desirable for `csplain` format and `cstrip` test. You can convert your old `csplain` documents to the new UTF-8 encoding and you can process them by the `csplain` in operating environment with UTF-8 standard. You get the absolutely the same result as in old days. This backward compatibility is most important for me.

Next example:

```
\mubyte \Alpha     ^^ce^^91\endmubyte
\mubyte \Beta      ^^ce^^92\endmubyte
...
\mubyte \leftarrow ^^e2^^86^^90\endmubyte
\mubyte \uparrow   ^^e2^^86^^91\endmubyte
...
```

For instance, the group of three bytes `^^e2^^86^^90` is now converted to `\leftarrow` control sequence and this control sequence is converted back to `^^e2^^86^^90` during `\write` if `\mubyteout` ≥ 3. The UTF-8 encoding of math characters are implemented by this way, see `utf8raw.tex` file in encTEX distribution and `math-example.tex` for more complex example.

The UTF-8 encoding tables for encTEX was prepared by David Nečas [6]. He has made his own Python script which converts the `NamesList.txt` [7] with UNICODE declarations of characters to the `\mubyte`...`\endmubyte` tables. This script is included in encTEX distribution.

There is another way of declaration of math symbols:

```
\mubyte \utfAlpha   ^^ce^^91\endmubyte
\mubyte \utfBeta    ^^ce^^92\endmubyte
...
\def\uftAlpha{\ensuremathmode \Alpha}
\def\uftBeta{\ensuremathmode \Beta}
...
\def\ensuremathmode #1{\ifmmode #1\else $#1$\fi}
```

This second solution is more robust because you can write math symbol in UTF-8 encoding without a need to start the math mode explicitly. Note that these symbols are displayed as natural math symbols in your text editor. I did

not use this solution in my macros distributed with encTeX because this concept is not compatible with common TEX documents where all math mode switches are explicitly written.

4 More funny examples

You can use encTEX capability for another purposes than only for encoding. Look to the next simple example:

```
\mubyte \TeX        TeX\endmubyte
\mubyte \copyright  (C)\endmubyte
\mubyte \dots       ...\endmubyte
```

If you write "`TeX and friends`" (without backslash) then input processor of encTEX converts this stream to `\TeX`, ⟨*space*⟩, `a`, `n`, `d`, ⟨*space*⟩, `f`, `r`, etc. This is desired behavior. Moreover, if `\mubyteout` ≥ 3 then the `\TeX` control sequence is not expanded during `\write` and it is converted back to its input byte sequence "`TeX`". On the other hand, if you write `\LaTeX`, then the input is converted to two control sequences `\La\TeX` and it is not desired. You can solve this problem by defining the "`\La`" macro or you can declare:

```
\mubyte \LaTeX      LaTeX\endmubyte
\mubyte \LaTeXe     LaTeX2e\endmubyte
```

Note that both byte sequences in this example begin by the same text "`LaTeX`". If the two characters "`2e`" follow immediatelly then `\LaTeXe` control sequence is generated (by second line of this example) else `\LaTeX` control sequence is generated. The order of the lines in this example is unimportant.

What happens, if this setting is active and you write `\LaTeX` (including backslash)? Nothing bad. The empty control sequence before generated control sequence `\LaTeX` is suppressed by encTEX, it means that only `\LaTeX` control sequence is the result of the conversion.

I implemented program `vlna` adding tildes after Czech one-letter prepositions (v, k, s, u, o, z) entirely in encTEX using `\mubyte`. It correctly handles math mode (no tildes are added there). It's available in the encTEX distribution as an example of crazy application of encTEX in the file `vlna.tex`.

5 References

1. `http://www.olsak.net/enctex.html`, the main page of encTEX project.
2. Petr Olšák: *EncTEX—A little extension of TEX*, in: TUGboat, vol. 19/4, pp. 336–371.
3. `ftp://ftp.math.feld.cvut.cz/pub/cstex/base/cstrip.tar.gz`.
4. Donald Knuth: *The TEXbook*.
5. `http://www/trific.ath.cx/tex-mf/enctex/`
6. `http://www/trific.ath.cx/`, David Nečas – home page.
7. `http://www.unicode.org/Public/UNIDATA/NamesList.txt`