☞ **Implementacje**

EncTEX—a little extension of TEX *

Petr Olšák

Motto: *Certainly, if I were a publishing house, if I were in the publishing business myself, I would have probably had ten different versions of TEX by now for ten different complicated projects that had come in. They would all look almost the same as TEX, but no one else would have this program—they wouldn't need it, they're not doing exactly the book that my publishing house was doing.*
Donald E. Knuth, Prague, March 1996

The article describes a simple change of TEX which gives the possibility to manipulate internal TEX vectors xord and xchr. These vectors are used to convert input encoding into TEX internal encoding and vice versa. For example, emTEX users (DOS or OS/2) know so called `tcp` tables which are used to set xord and xchr values. On the other hand UNIX users have no chance to reset these values when the TEX binary is compiled already. The below described modification of TEX enables something similar to emTEX `tcp` tables. It is independent on the operating system. You can implement its in all TEX systems where the TEX program is compiled from WEB source files. I have tested my modification on UNIX systems with web2c implementation of TEX.

There existed two options to work with reencoding on UNIX web2c implementations so far. The first one goes from Skarvada's patch [3]. This solution implements several reencoding tables directly into the TEX source and user cannot change these tables at run time of TEX. The encoding table is selected by environment variable and is not saved into generated formats during iniTEX. I mean, these solution is not so flexible.

The second option was implemented through `tcx` files by Karl Berry. It is commented out in current web2c sources with the following note: "`tcx` files are probably a bad idea, since they make TEX source documents unportable. Try the `inputenc` LaTEX package." I know the `xord`/`xchr` reencoding creates new problems, when the `inputenc` package is used, but nevertheless I disagree with this note. I have the following arguments:

- The `inputenc` package is a solution for LaTEX only, but TEX is used via other formats too.
- The `log` file and the terminal output are not legible if overfull/underfull box of Czech text is reported. The `^^`notation is absolutely funny. If the section 49 of `tex.web` is changed (via `tex.ch` of course) this way: `(k<" ") or (k=invalid_code)`, the `^^`notation does not occur any more and the text is legible. But if `inputenc` is used with different internal TEX encoding, the Czech sentences in `log` files are still not legible.
- The reencoding is the *implementation* problem, it is not the problem of a naive user, who must write the `\usepackage[bla]{inputenc}`. He or she has no knowledge about encoding used in his/her OS. He or she perceives the `\usepackage[bla]{inputenc}` as a very mystical one.
- If the LaTEX document is sent via e-mail with MIME (or similar methods of transport), the re-encoding is done by e-mail agents and the document is properly encoded for the OS of the receiver. The `\usepackage[bla]{inputenc}` is not changed in LaTEX header automatically, thus if the sender and the receiver work in different encoding—viola—Houston, we have a problem. I think, the reencoding must be solved in software for transports between different OS (e-mail agents or WWW servers/clients, for example) and this problem cannot be solved in LaTEX header.
- The `inputenc` package sets active `\catcode`-s to accented characters. So, the `Olšák` is expanded to `Ol\v s\'ak` therefore you cannot define the control sequence `\Olšák`. Accented

*: [Od Redakcji:] Wielu użytkowników emTEX-a od lat wykorzystuje niezwykle wygodną w praktyce możliwość przekodowania układu znaków w pliku źródłowym na wewnętrzny układ znaków TEX-a. Dużą niedogodnością jest dla nich brak takich mechanizmów w dystrybucjach TEX-a opartych o web2c (np. MikTEX, teTEX).

Zapewne nie wszyscy wiedzą, że takie mechanizmy są już zaimplementowane w web2c, ale celowo zostały usunięte (wstawione wewnątrz komentarza) na życzenie prominentnych osób związanych z projektem LaTEX2ε. Powodem tego jest możliwość tworzenia nieprzenośnych dokumentów przez niewprawnych użytkowników TEX-a. Jest to prawda, ale czyż TEX ma być ograniczany do potrzeb użytkowników o najmniejszych potrzebach?

Więcej na ten temat w następnych numerach Biuletynu. Dziś artykuł opisujący szerzej ten problem i jego – przyznajemy kontrowersyjne – rozwiązanie.

letters have \catcode=13 but \catcode=11 is needed.

- Donald Knuth has implemented the xord/xchr vectors into TEX to separate the encodings used in OS and the internal TEX encoding (because text fonts used in TEX are independent on OS). Administrator must setup xord/xchr values during the TEX WEB source state. Why don't they do it? Because there are many TEX implementations with binary TEX only. Even if the TEX WEB source is available, setting xord/xchr is somewhat difficult for some administrators. But if the xord/xchr setting is possible during the iniTEX state, the administrators will be more flexible to choose the right setting for their OS.
- I think Donald Knuth did not take into the account the possibility to reencode during the expand processor state, as it is done by inputenc package. Just consider that the \uppercase, \lowercase primitives do their work on ⟨*balanced text*⟩ *before expanding* using \uccode-s and \lccode-s, which are used in hyphenation algorithm *after expanding*.

My solution of the reencoding problem is more general than the tcp tables or the tcx files, because the encTEX tables are read during iniTEX simply by using \input and are defined by TEX macros. I have implemented three new primitives into TEX: \xordcode, \xchrcode and \xprncode. They enable to set and read the values of xord and xchr vectors and to set the "printability" attribute of any character. A new quality is introduced: the xord/xchr vectors may be set independently. This opens great new possibilities.

The technical introduction

The xord vector is 256 bytes long and stores the reencoding information for inputting characters from text file into TEX. The xchr vector has the same length and stores the reencoding information for outputting characters from TEX to the terminal, to logs and to the files created through \write, but does not influence outputting to dvi files. These vectors are build into the program. All text information during input or output is reencoded by these vectors. If the input character has an external code x and the internal code y in TEX, the xord vector must be set the following way: xord[x] $= y$. The rules for the output characters are as follows: If the character with internal code y is not assumed to be "printable" then the ^^*code*y is outputted otherwise the character with code $x =$ xchr[y] is written.

The encTEX package

The installation of encTEX was tested on web2c version 7. If we have the WEB sources of this implementation of TEX then the command patch <enctex.patch in the directory with tech.ch installs the encTEX package. After that, the new compilation of TEX binary (make tex) is needed. For more details see the INSTALL.eng file.

The patch changes the tex.ch file only. No other files including the C libraries are changed. The make tex command runs the tangle on the main source file tex.web and on the changed change-file tech.ch. The Pascal source file tex.p is created and it is converted into C by the convert script and afterwards it is compiled into run time binary tex.

Different TEX implementations (than web2c) can have different tex.ch files, thus the simple patch command (for web2c) may not be applicable. For that reason the enctex.ch file is included. All encTEX specific changes are described in this file. You can modify your tex.ch file manually using information from this file.

The encTEX modification is independent on operating system and on TEX implementation because all the changes are done in WEB change file exclusively.

After encTEX is installed, you can read and set the values of xord and xchr vectors by new primitives \xordcode and \xchrcode. You can set the "printability" attribute of the character by new primitive \xprncode. The syntax of all three new primitives is the same as the syntax of \lccode or \uccode primitives. For example:

```
\xordcode"AB="CD
\xchrcode\xordcode"AB="AB
\the\xchrcode200
```

xord[0xAB]=0xCD and xchr[xord[0xAB]]=0xAB are set and the value of xchr[200] is printed in this example.

The new primitive \xprncode enables to set the "printability" attribute of the character. The character with internal code y is "printable" if and only if $y \in \{32\ldots126\}$ or \xprncode$\,y > 0$. For example, if we set \xprncode255=1, then the character with the code 255 is "printable" and it will be printed as a character with the code `xchr[255]`. On the other hand setting `\xprncode^` to zero does not cause "non-printability" because the code of the character "^" is in $\{32\dots126\}$. It is a kind of "self-defence instinct" against an unsound user who could set all characters to be "non-printable" and the printing ability of the program may be lost. The `\xprncode` primitive can take any value from the range $0\dots255$, but remember the rule—"printable" if `\xprncode` is positive.

There is an important difference between new encTEX primitives and well known primitives like `\lccode` or `\catcode`. The new primitives represent internal TEX registers and are *global* under any circumstances. You can set these values in a group and these settings are not changed at the group end. I turned down the possibility of local settings (via eqtb table) in order to achieve more efficient code.

The initial values, when the iniTEX starts, are following:

$\texttt{\textbackslash xordcode}\, i = i$ for $i \in \{128\dots255\}$,
$\texttt{\textbackslash xchrcode}\, i = i$ for $i \in \{128\dots255\}$,
$\texttt{\textbackslash xprncode}\, i = 0$ for $i \in \{0\dots31, 127\dots255\}$,
$\texttt{\textbackslash xprncode}\, i = 1$ for $i \in \{32\dots126\}$.

The values `\xordcode` i and `\xchrcode` i for $i \in \{0\dots127\}$ depend on the operating system. If the system is using the ASCII standard (very common) then `\xordcode` $i = i$ and `\xchrcode` $i = i$ for all i. If the operating system is using another (obscure) encoding standard, then 95 printable ASCII internal codes from $\{32\dots126\}$ are mapped into appropriate codes through corresponding changes in `\xordcode` and `\xchrcode` initial values.

The `\xordcode`, `\xchrcode` and `\xprncode` values are stored in the format file and they are restored during the run of the production version of TEX.

About the ambiguous encoding

This subsection will describe some issues with `xord` and `xchr` resetting. Let us assume, we need to map the character á (code 129 in OS for example) to internal TEX code 128. So let `\xordcode129=128`, `\xchcode128=129` and `\xprncode128=1`. At the same time the input code 128 is not mapped because it is never used in Czech alphabet, for example. What if I get some file from Poland containing the character with the input code 128? This character is mapped to the code 128 (internal in TEX) but it is returned to `log` as the code 129, that means as the character á. TEX is never able to distinguish between codes 128 and 129 on the input.

We will describe this phenomenon more exactly. Let's use mathematical terminology. Let $X = \{0\dots255\}$ be a set of input codes and $Y = X$ be the same set (from mathematical point of view) but let's use this letter for a set of the internal codes in TEX. Let $Y_p \subseteq Y$ be a set of all printable characters. We claim:

$$Y_p = \{y; \texttt{\textbackslash xprncode}\, y > 0\} \cup \{32\dots126\}.$$

It is obvious that the values of `xchr` vector on the set $Y \backslash Y_p$ don't influence the behavior of the program output.

Let $I : X \to Y$ be the "input" function defined by `xord` vector and $O : Y_p \to X$ be the "output" function defined by `xchr` vector. The initial values of `xord` or `xchr` ensure that I is bijective and O is injective and $O = I^{-1}$ on Y_p. This feature gets lost after the first change of `xord` or `xchr` values. For example, let $x \neq y$ and $x \in X$, $y \in Y_p$. Let us make a simple transposition:

$$\mathrm{xord}[x] = y, \quad \mathrm{xchr}[y] = x. \tag{1}$$

Now, neither I function nor O function are injective! You can see, $\mathtt{xord[}x\mathtt{]} = \mathtt{xord[}y\mathtt{]}$ and similar equations hold for `xchr` vector. The following condition must be fulfilled so that our functions is injective after applying transposition (1) n-times. The sequence $x_0, \dots x_n$ must exist with these properties:

$$x_0 = x, \quad x_1 = y \quad \text{a} \quad \mathrm{xord}[x_i] = x_{i+1}$$

for all $i \in \{0\dots n-1\}$ and the equation $\mathtt{xord[}x_n\mathtt{]} = x$ holds. The similar properties must be valid for the `xchr` vector. The problem is, that we apply the transpositions (1) only on a certain subset of X (for example on the printable characters or on accessible characters in given encoding). Then we need not to be surprised that as a result of our settings neither I nor O are injective functions and therefore equations $O = I^{-1}$ or $I = O^{-1}$ are senseless. The inversion exists only if the function is injective.

The encoding tables. There are two types of encoding tables in encTEX. Both tables are TEX `\input` files with auxiliary macros. The files have common extension `tex`. It is recommended to use these tables (or to modify them to your needs). Don't use new primitives `\xordcode`, `\xchrcode` and `\xprncode` directly unless you exactly know what you are doing.

The first type of encoding tables

These tables are used during the iniTeX run. The values of xord/xchr are set symmetrically during the \input, the transposition (1) is used repeatedly for setting of the xord/xchr values. The resulting settings are stored in format file using \dump and used in production version of TeX.

These tables declare the relation of internal TeX encoding and the encoding used on the host operating system. For example, our system encoding is ISO8859-2 and internal TeX encoding is chosen by Cork standard (called T1 in LaTeX). In this case, the encoding table name is il2-t1.tex. It redefines the xord vector to map ISO8859-2 to T1 and xchr vector to map T1 back to ISO8859-2. A part of the il2-t1.tex table is shown in Appendix 1 at the end of this article.

The first thing every encoding table does is an input of the macro file encmacro.tex, which consequently defines macros \setcharcode, \expandto, \texmacro, \texaccent. See README.eng file in the encTeX package for detail documentation of these macros.

Secondly, the internal encoding specific macro is read. An input of the t1macro.tex file is performed in our example. The encoding specifics macros (such as accent definitions) are placed here. These macros solve the similar issues as the fd files for text fonts in LaTeX.

See Appendix 2 for the overview of all tables of the first type included in encTeX. You can list these files by

```
ls *-csf.tex *-t1.tex
```

EncTeX contains many prepared files for iniTeX for plain-variant formats. For example, the command

```
initex plain-il2-dc
```

generates the plain format with ISO8859-2 as the input encoding. This format name is plain-il2-dc, it includes the hyphenation table in T1 and uses dc fonts for text. The content of the plain-il2-dc.tex file is shown in Appendix 3. The \input of the original Knuth's plain.tex and the table il2-t1.tex is performed here.

The second type of encoding tables. The second type of tables performs reencoding only on the input side of TeX, so the xord values are changed but xchr values are not. The name convention identifies these tables: the symbols t1 or csf are missing in the name, because this type of tables solves reencoding from one operating system standard to another and is not related to TeX internal code.

For example, the table 1250-il2.tex maps the input characters from CP1250 to ISO8859-2. The CP1250 encoding becomes a new input encoding and we assume that the first type of encoding table il2-*.tex was used in iniTeX.

We can use the table of the second type, when a part of the input document (or the whole document) has a different encoding from the encoding used by our operating system. The table of the second type has to be in relation with the input encoding declared in the table of the first type.

For example, the il2-t1.tex table was used in iniTeX and we have obtained the document in CP1250. We can write:

```
\input 1250-il2
\input document
\restoreinputencoding
The ISO8859-2 is restored here.
```

The double reencoding is active when the document.tex is read. Firstly from CP1250 to ISO8859-2 and secondly from ISO8859-2 to internal TeX T1 encoding. The text is outputted to log, to terminal and to \write files in ISO8859-2 encoding. The ISO8859-2 input encoding is restored after \restoreinputencoding command.

Attention: it is impossible to re-read the \write files when the table of second type is active. If, for instance, the document.tex includes some \write activities (for index, table of contents and so on), we have to read these auxiliary files *before* \input 1250-il2 or *after* \restoreinputencoding. That is the reason why the format \dump is senseless when table of the second type is active.

About compatibility. The encTeX extension successfully passes the TRIP test with two exceptions: 1. The banner is changed, 2. The number of multiletter control sequences is greater than in original TeX by three.

All changes of TeX which do not change the behavior of original TeX and only add some new primitives are backward compatible with the original Knuth's TeX. It means that if a document is written for original TeX and is processed through extended TeX we will get the same results. Here is one exception though: the macro construction of the type \ifx\xordcode\undefined has to be missing in such a document. But, I guess, the probability of existence of such constructions in documents for standard TEX is equal to zero.

We have to say that it is possible to write new macros and documents in extended TEX which are not backward compatible with original TEX. This is a disadvantage of all extensions of TEX. We face this situation both if the extension adds new primitives directly (as in encTEX or pdfTEX, for example), as well as if the access to new primitives is hidden and may be initialized by some trick at the format generation time (e-TEX or MLTEX). The issue is, how many users will use new primitives and who will be a supervisor for standardization of these primitives.

In case of my encTEX package, I have no claim to standardize its primitives into newly developed TEXs. I have made this extension for my needs and if somebody likes it, he/she can use it realizing that his/her documents may not be backward compatible if he/she uses the new primitives directly. On the other hand, I meant my primitives to be used primarly while formats generating and not to be used directly in real documents. Thus the documents can still be backward compatible.

If an administrator of a multi user system installs TEX format using encTEX, he/she can call some table of the first type and prohibit usage of new primitives before `\dump`:

```
\let\xordcode=\undefined
\let\xchrcode=\undefined
\let\xprncode=\undefined
```

Thus users can't access the backward incompatible extension of encTEX. I recommend this setup for public sites. If encTEX is used this way exclusively, then the `xord` and `xchr` vectors work as was meant by the author of TEX: They filter operating system specific encodings into internal TEX encoding.

I suspect that the author of TEX Donald Knuth wanted the administrator to set the `xord`/`xchr` values using WEB change-file before the TEX compilation into run time binary. Unfortunately, this procedure is not very common for UNIX `web2c` implementations because an extra work has to be done by the administrator (not only `make tex`). The new primitives of encTEX postpone the `xord`/`xchr` setting from the WEB state to the format initialization state. This is more understandable for many administrators (the similar solution is used in emTEX via `tcp` tables, for instance). The every TEX implementor has to set `xord`/`xchr` values. The only work I have done is that I have solved the `xord`/`xchr` setting at WEB state the most flexible way.

The question in my mind is why Donald Knuth did not introduce primitives similar to mine. He probably wanted that all TEX macros behave the same way on various implementations. In this case the direct access to `xord`/`xchr` values was not acceptable for him. We can use the condition `\ifnum\xordcode`@=`@` thus our macro processing depends on wether or not the operating system adopts the ASCII standard.

The same behavior of macros is not exactly reached in standard TEX neither. We can `\write` a character into a file and we can re-read this character in the next run. If we set `\catcode`^=12` before re-reading then we can conditionally continue processing based on the “printability” attribute of this character in given operating system.

Conclusion

Everybody can modify the TEX source for his needs (see the quotation from the author of TEX at the beginning of my article). Modifying the TEX source is simplier than what it looks like. In my case, I have perused [1] in the evening and I reconsidered all problems. Next day in the morning, I have implemented my ideas into computer and I have performed a couple of tests. And I have written this article in the afternoon (in Czech language; the English version took me considerably more time:-). The goal was reached quickly due to very well documented program TEX.

Reference

1. Donald Knuth. *TEX: The program*, volume B of *Computers & Typesetting*. Addison-Wesley, Reading, MA, USA, 1986.
2. Petr Olšák. *The encTEX package*, `ftp://math.feld.cvut.cz/pub/olsak/enctex`.
3. Libor Škarvada. *The patch for* `web2c` *TEX*, `ftp://ftp.muni.cz/pub/tex/local/cstug/skarvada`.

◇ Petr Olšák
olsak@math.feld.cvut.cz

Appendixes

Appendix 1: The part of the table of the first type `il2-t1.tex`.

```
%%% The encoding table, v.Sep.1997 (C) Petr Ol\v s\'ak
%%% input: ISO-8859-2, internal TeX: Cork

\input encmacro  \input t1macro

% (1) Czech/Slovak alphabet
%            input TeX  lc   uc    sf cat prn       sequence
\setcharcode  "C1  "C1  "E1  "C1   999  11  1  \texaccent \'A
\setcharcode  "E1  "E1  "E1  "C1  1000  11  1  \texaccent \'a
\setcharcode  "C4  "C4  "E4  "C4   999  11  1  \texaccent \"A
\setcharcode  "E4  "E4  "E4  "C4  1000  11  1  \texaccent \"a
\setcharcode  "C8  "83  "A3  "83   999  11  1  \texaccent \v C
\setcharcode  "E8  "A3  "A3  "83  1000  11  1  \texaccent \v c
...
\setcharcode  "A4  "1F   0    0     0  15  0  % =o=, not accesible
\setcharcode  "A7  "9F   0    0     0  12  1  \texmacro \S
\setcharcode  "D7  "03   0    0     0  13  0  \expandto {$\times$}
\setcharcode  "F7  "07   0    0     0  13  0  \expandto {$\div$}
```

Appendix 2: The list of tables of the first type.

File name	input encoding	internal TeX encoding
il2-csf.tex	ISO8859-2	CS-font
kam-csf.tex	Kamenických	CS-font
1250-csf.tex	CP1250, MS-Windows	CS-font
852-csf.tex	CP852, PC Latin2	CS-font
mac-csf.tex	MAC-CZ, Macintosh	CS-font
il2-t1.tex	ISO8859-2	T1 alias Cork
kam-t1.tex	Kamenických	T1 alias Cork
1250-t1.tex	CP1250, MS-Windows	T1 alias Cork
852-t1.tex	CP852, PC Latin2	T1 alias Cork
mac-t1.tex	MAC-CZ, Macintosh	T1 alias Cork

The CS-font encoding and T1 are commonly used as internal encoding of TeX for our language. The CP1250 is commonly used in MS Windows systems, the ISO8859-2 in UNIX, the CP852 or Kamenických encodings are used in DOS and the MAC-CZ is used in Macintosh systems.

Appendix 3: The content of `plain-il2-dc` file.

```
\input noprefnt % re-defines \font: \font\preloaded is ignored
\input plain    % format Plain
\restorefont    % original meaning of primitive \font
\input dcfonts  % loads text-style dc fonts
\input il2-t1   % input: ISO8859-2, internal TeX: Cork
\input hyphen.lan  % czech / slovak hyphenation pattern
\input plaina4  % \hsize and \vsize for A4
\everyjob=\expandafter{\the\everyjob
   \message{The format: plain-il2-dc <Sep. 1997>.}
   \message{The cm+dc-fonts are preloaded and A4 size predefined.}}
\dump
```