RTFORMAT — an alternative way to add Cyrillics into LaTeX

Alexander S. Berdnikov
Institute of Analytical Instrumentation
St. Petersburg, Russia
berd@ianin.spb.su

Andrew V. Janishevsky
St. Petersburg, Russia
Andrew@p0.f209.n5030.z2.fidonet.org

Olga G. Lapko
MIR Publishing House
Moscow, Russia
olga@mir.msk.su

General remarks

Before we start further, it is a good idea to specify what we would like to achieve as an output. While currently the package T2ENC covers the requirements of 99% Cyrillic TeX users, there is 1% which are not quite happy with it. The problem is that in our case that 1% of TeX-users produces 99% of russian TeX-based documents ☺ — these users are the workers of a big publishing organization involved in preparing review magazines, etc., for the whole Russian Federation. As it could be foreseen, the speed of text typing is essential in this case (moreover, it is a very critical factor), and the necessity to switch the keyboard into English each time it is necessary to enter some macro makes this process more slow than it could be. Second, technology used in preparing rapid review publications involves a lot of specialized pre- and postprocessing utilities, and the necessity to analyze the cryptograms produced by T2ENC as the auxilliary files to some extend is the obstacle in this case.

What we need is to have in TeX russian letters as the ordinary letters (namely, to have the possibility to make macros with russian names) and to make input encoding, internal encoding and output encoding to be equivalent. Since T2ENC cannot satisfy these requirements, it is necessary to have the alternative way (although it may be less standard) of adding Cyrillics into TeX. And the package RTFORMAT described here just suites this purpose.

Although it is not obligatory, but since T2ENC enables to switch *dynamically* between several input encodings inside the same document, it is a good idea if RTFORMAT also can do it — although (may be) not so flexibely and effectively. And although it is not so easy (and, honestly saying, not so necessary) to perform dynamical switching of input encodings under the restrictions specified above, RTFORMAT can do it as well.

Now, after a brief foreword, let us proceed to the main part.

Multiple encodings

One problem with 'Russian TeX' is that different platforms use different encodings to represent russian letters. Moreover, different platforms use different cyrillic extensions (i.e., non-russian cyrillic letters) as their native encodings. The general LaTeX 2_ε approach implemented in T2enc-package solves this problem by considering *input encoding*, *internal presentation* and *output encoding* as separate black boxes interacting with each other through complex but rigorously defined interfaces. But the price is that 'russian letters' are no more *letters* from TeX point of view — they are active symbols which are mapped on some system-defined macros, which in turn are mapped into output font encoding. As a result the User cannot use russian letters for his/her macro names any more, but the unavoidable consequence of this approach is also that the output files (and the screen output as well) are some system-encoded cryptogramms which can hardly be read by human eyes without postprocessing by special utilities.

The alternative way is to let the input encoding, output font encoding and the internal encoding to be just the same, and it is just the approach which is used by RTFORMAT. The price is that now we should have independent set of fonts and hyphenation tables for each encoding (awful!), but as a bonus we can read native russian letters on screen, in .log-file and auxilliary files, and that we can use russian letters in user-defined macro names as well.

To increase the compatibility between various programs, the unified set of cyrillic letters common to all leading computer platforms was created and implemented as the unified set of encoding-dependent fonts. So, when you move some cyrillic text from one platform to another, it is possible that the native encoding of your new platform does not contain cyrillic letters existed for the former platform. But for TEX-fonts used for these platforms it is not so: they contain just the same subset of cyrillic letters although on different positions, and you do not loose something when you move from one platform to another.

To make life easier, for each russian (cyrillic) letter there is the corresponding macro similar to that used in T2ENC — as a result you can code laguage-dependent and encoding-dependent texts in your style files and macro packages without knowing the actual encoding of your platform.

Hyphenation patterns

The hyphenation patterns for Russian have some specific features absent in European latin-based languages. One of it is already described above: there are more than one encoding for Russian. As a result, if we would like to support several input encodings inside the same TEX format without re-encoding mechanism like TCX or T2ENC, we need to store the hyphenation table several times — one properly encoded copy per each input encoding. (It requires enormous memory expences and is not recommended without urgent needs: usually one input encoding is enough.)

The second is that there are *several* variants of hyphenation tables for russian language created by various authors. While only few of them are still supported by the developers, the User should have the possibility to select the desired hyphenation table. As a result RTFORMAT has the possibility of selection what hyphenation table by what author and in what input encoding should be used when the format file is generated.

Finally, English hyphenation table and Russian hyphenation table can be separated as two independent tables or can be joined as a single hyphenation table. Both variants have their own advantages and disadvantages (in particular, Russian and English uses different values for `\lefthyphenmin` and `\righthyphenmin`). So, the User should have the possibility to select what variant is desirable, and RTFORMAT has all necessary options to do it.

Mathematics in Russian TEX

It is a long-term discussion should we use russian letters (or, more generally, arbitrary cyrillic letters and symbols) in mathmatical mode or not. Both poits of view have their adherents and advocates, and the best way is to allow the User to make the final selection up to his/her choice and taste. So, RTFORMAT has the possibility to control the way how cyrillic letters can be used in mathematical mode:

- rigorous — it is not allowed to use russian letters or some other cyrillic letters or symbols at all (which does not mean that you cannot insert it, for example, inside `\hbox` in mathematical mode),
- standard — it is allowed to use russian uppercase and lowercase letters (except Ë/ë), but additional cyrillic letters and cyrillic symbols are not allowed,
- free — all cyrillic letters and cyrillic symbols are legal in mathematical mode.

The mode how cyrillic letters are interpreted in mathematics is controlled dynamically inside the document and the User can change this mode by special macros. (The default mode is selected when the format file is generated.) The ‘prohibited’ letters are substituted by circles in .dvi-file and they are clearly visible when the document is looked through.

Russian fonts

While there are several russian fonts on CTAN, only LHFONTS by Olga Lapko are suitable for our purpose. Namely, this package contains all additional cyrillic letters which are necessary to achieve the compatibility between various platforms (see section *Multiple encodings*), and it also has a very flexible tool to change the font encoding according to our wishes. (More about it can be found in documentation supplementary to LHFONTS.) So, the unified set of fonts based on the unified encodings characteristic to leading computer platforms (see section *Multiple encodings*) was created using LHFONTS as the basis. It will be uploaded on CTAN as soon as α-testing stage is passed.

While we have the possibility to change *dynamically* input encoding, it means that font encoding (output encoding) is also changed dynamically. It is assumed that all fonts corresponding to various (and different) encodings follow the scheme standard for CM-fonts, and that the user-defined prefix is the only thing which is different for these font sets. As a result all the fonts in RTFORMAT are defined not as constants, but by a special *cyrillic prefix*, and at any moment by calling corresponding macro the User can redefine the whole font system used by TeX from the very beginning.

TeX formats supported by *RTFORMAT*

At this moment RTFORMAT is used to add Cyrillics (i.e., Russian) to the following TeX formats: *Plain* TeX, AMSTeX, *Eplain*, LaTeX 2.09, LaTeX 2_ε, any format based on *Plain* TeX, all previous formats combined with BABEL.

All cases except LaTeX 2.09 and LaTeX 2_ε follows just the same scheme: the command `\dump` is redefined in such a way that except dumping it loads the file rtgener.tex which in turn adds Russian hyphenations, Russian encodinds, additional commands specific to Russian TeX, etc. At the beginning of the procedure the file rtformat.ini is loaded — it defines desired encodings, cyrillic fonts, mathematical fonts (if necessary), how to treat cyrillic characters in mathematical mode (section *Mathematics in Russian TeX*), additional changes defined by the User. (Such approach enables to specify necessary parameters for all formats simultaneously.) After it a small file rtcorr.tex is loaded — it defines in advance some *cyrillic* fonts instead of standard CM-fonts because there are some formats, style files, macro packages (*Eplain*, *XEplain*, BTXMAK.TEX, TEXNAMES.STY) which define them as pure latin fonts. Next the main format file is loaded and processed — for *Plain TeX* it differs from the original only in section which defines the fonts (see earlier section *Russian fonts*). If necessary, additional format file (for example, eplain.tex or amstex.tex) is loaded after this *plain* kernel. Finally the command `\dump` is executed which adds necessary corrections to already defined 'latin-based' format and dumps TeX memory as the predefined format file.

For TeX formats with BABEL this scheme is a little bit different: BABEL itself modifies the command `\input` in such a way that it loads the file hyphen.cfg instead of original file hyphen.tex. While in BABEL this substitution produces strange results if the main file hyphen.cfg does not exist, in RTFORMAT the User gets self-explanatory message informing what is wrong in his/her system. Except these BABEL-specific corrections the modified command `\dump` also redefines the format name (`\fmtname`) in such a way that it correctly reflects that now you are working with *combined* TeX format.

As usually, LaTeX 2.09 and LaTeX 2_ε requires much more work — it is described in the next two sections.

Tricks specific for LaTeX 2.09

The font manager in LaTeX 2.09 is quite different from that used in *Plain TeX*. Hopefully Leslie Lamport organized this part of his package in a smart way: all font-specific commands are placed in a special file lfonts.tex with comments enabling the User to modify them up to his/her wish. It is not so easy to force LaTeX 2.09 to load your own file instead of lfonts.tex — the command `\input` is redefined inside LaTeX 2.09, and the trick which works with BABEL fails. Since all known TeX compilers consider the files in the current directory as that to be loaded first, we can put the modified file lfonts.tex in current directory and hence force LaTeX 2.09 to load our own file. Not to create several files lfonts.tex with different content the procedure (extbat-file) which generates LaTeX 2.09 format file copies the file rtlfonts.tex under that name before TeX-compiler starts, and destroys it immediately when the work is finished.

But except font definitions LaTeX 2.09 contains more language-specific commands. Namely, these are the commands

> `\prefacename`, `\refname`, `\abstractname`, `\bibname`, `\chaptername`, `\appendixname`, `\contentsname`, `\listfigurename`, `\listtablename`, `\indexname`, `\figurename`, `\tablename`, `\partname`, `\enclname`, `\ccname`, `\headtoname`, `\pagename`, `\seename`, `\alsoname`, `\proofname`

defined inside LaTeX 2.09 and its standard style files. RTFORMAT corrects it by defining the commands `\russiancaptions` and `\englishcaptions` which switch the definitions of the language-specific commands (they are called automatically when the language is switched). To make it encoding-independent (see section *Multiple encodings*), all russian names are defined by macro names (i.e., `\CYRA`, `\CYRB`, etc.), not by explicit characters or character codes. These language-specific commands are defined inside the file rtltx209.tex loaded by the modified command `\dump` just after it loads the main file rtgener.tex.

Some troubles are also caused by russian letters in mathematics (see section *Mathematics in Russian TeX*) because switching LaTeX 2.09 font definitions (see section *Russian fonts*) is not reflected automatically in definitions of mathematical font families. Hopefully, this problem is solved very easy — it is enough to change the font size (say, by the command `\large\normalsize` to update mathematical families. Side effect is that the font size becomes `\normalsize` each time you change the encoding, but it seems that the User can survive even with this feature — honestly saying, it is a rare chance that somebody changes the *input encoding* in his/her file.

Tricks specific for LaTeX 2_ε

LaTeX 2_ε requires much more efforts. First of all, its font manager (i.e., *NFSS* tables and *NFSS* internal commands) are similar to 'black boxes' without any documentation how it works in real system. Second, the authors of *NFSS* do not destine it for dynamical redefinition in *online*-style — so, although most characteristics are defined when your document is processed, not when the format file is generated, they may be defined only once per each LaTeX run. Third, there are many non-documented tricks and features in *NFSS* which makes it very difficult to interrupt its internal work and to correct the operations intrinsic for LaTeX 2_ε kernel. So, more corrections are necessary for LaTeX 2_ε as compared with LaTeX 2.09. (All these corrections are placed in a special file `rtltxe.tex` which is loaded by the modified command `\dump` just after it loads the files `rtgener.tex` and `rtltx209.tex`.)

To skip the whole history, let us consider the *final* solution used by RTFORMAT. The command `\get@external@font` is redefined: it calls original `\get@external@font`, analyzes its result (say, '`cmr10 at 12.0pt`') and substitutes the original CM-fonts by corresponding cyrillic fonts — say, in our case it returns as a result '`lhr10 at 12.0pt`' when LH-fonts are used and CP866 is the input encoding. (It is worth to note that the analysis of the result returned by the original `\get@external@font` is not too simple: original macro returns the answer where even the *letters* like like '`cmr`' have `\catcode`=12.)

But it is not enough to make the whole system works. As it was already told, *NFSS* tables are the static objects — namely, they are defined only once and are not redefined more and more again even if you think that they should do it. For example, suppose that there is the font with encoding `OT1`, family `cmr`, weight `m`, shape `n` and design size `10pt` — its internal name is '`\OT1/cmr/m/n/10 `', where the final space is an essential component of the name. As soon as it happens, the macro `\get@external@font` is not called any more when you need the font with these characteristics once again. Not to produce total garbage when the User changes the input encoding (see section *Multiple encodings*), the list of already defined fonts is stored in a special array and all these fonts become undefined when the input encoding is changed.

Do you think it is so? — No, definitely not, because except *text fonts* there are also *math fonts*, and the latter are controlled by *NFSS* quite differently. Direct tracing of LaTeX 2_ε commands (when mathematical mode is activated) results to a long and comprehensive list of expanded macros. It is nearly impossible to understand what happens inside *NFSS* if you are not the TeX-guru, not the member of *LaTeX Project Team*, do not have transparent documentation how *NFSS* works internally (the latter does not exist at all with big probability ☺). Hopefully, there is the initial trigger which initializes the whole mathematical system: if you define `\glb@currsize` as empty macro (`\gdef\glb@currsize{}`) and execute the command `\let\ifmath@fonts=\iftrue`, LaTeX 2_ε thinks that mathematical system is not initialized and initializes it from the very beginning calling `\get@external@font` each time it needs new mathematical font. (It is definitely not effective, but at least it works.)

Acknowledgements

We express our warmest thanks to *CyrTUG* and *CyrTeX* activists who supported our work by their careful attention and valuable critics. Special thanks are to Irina Makhovaya who was the leader of *CyrTUG* for nearly 10 years and whose impart on the development of TeX in Russia and former Soviet Union states can scarcely be exaggerated.

Alexander Berdnikov would like to thank separately Dr. A. Compagner from Delft University of Technology for his long-term friendship and patient attention as a teacher — not necessarily relating to some TeX or LaTeX joint works.

◇ Alexander S. Berdnikov
`berd@ianin.spb.su`

◇ Andrew V. Janishevsky
`Andrew@p0.f209.n5030.z2.fidonet.org`

◇ Olga G. Lapko
`olga@mir.msk.su`