Adding Native Language Support to the CWEB package and the TEX program

Włodek Bzyl
Instytut Matematyki, Uniwersytet Gdański, Wita Stwosza 57, 80–952 Gdańsk, Poland
matwb@univ.gda.pl

Abstract

By adding National Language Support (NLS, for short) to literate programs I propose making such changes in their text via change files, which make modified programs aware of and able to support multiple languages. This paper describes how the GNU *libc* and *gettext* libraries were used to add NLS to the CWEB package and presents a possible way of bringing NLS to the TEX program.

Introduction

In 1984 D. E. Knuth wrote about the WEB system [Literate Programming. *The Computer Journal*, 27]: "I made a conscious decision not to design a language that would be suitable for everybody. My goal was to provide a tool for system programmers, not for high school students or for hobbyists. I don't have anything against high school students and hobbyists, but I don't believe every computer language should attempt to offer all things to all people."

Now, it can be said that WEB systems are used by a small elite of literate programmers who are able to combine their English verbal and programming skills. This is so, because all existing WEBs have been created with English conventions in mind, so these tools should not be expected to work well in non-English environments. Having realized that, I asked myself: does a WEB system exist that could be untied from the English conventions and tied anew to conventions of another language? At that time I was experimenting with FWEB, noweb, and CWEB. Of these systems only CWEB supported the use of non-Latin characters in literate programs. This feature made me believe that a CWEB adaptation to the Polish conventions and the conventions of other languages is possible. After some work I had a version of CWEB usable by any programmer literate in Polish.

When I was experimenting with the changed CWEB, new GNU *libc* and *gettext* libraries appeared. These libraries make it possible to write C programs that automatically adapt to local sets of conventions set up by the values of some environment variables. This forced me to rethink what I had done. New functionality offered by functions from these libraries makes it possible to have *one* CWEB that could be used for writing literate programs in English, Polish and many other languages.

If this idea is feasible, it would make literate programming accessible to programmers who like to write and to explain what they are doing in the language of their choice, or in the language appropriate for the audience to whom they are going to present their concepts and ideas. Moreover, the World Wide Web would not be populated with slightly different CWEBs and the CWEB system of Levy and Knuth will stay open for improvements by everyone.

Programming interface for NLS — the ISO C model

In the ISO C model, NLS works by means of *locales* divided into six categories, to be selected and activated independently. Each category specifies a collection of conventions — one set of conventions for each category. Here is the list of all categories.

`LC_CTYPE` — specifies the character set
`LC_COLLATE` — specifies the conventions for sorting order
`LC_MESSAGES` — specifies the language for messages
`LC_MONETARY` — specifies the formatting of monetary quantities
`LC_NUMERIC` — specifies the formatting of numbers
`LC_TIME` — specifies the formatting of dates and times

Each category name is both a macro name to be used in C code and an environment variable that a user can set. There is also a special C macro `LC_ALL` used to select all sets of conventions and there are two special environment variables.

`LC_ALL` — if defined, its value specifies the locale to use for all purposes

`LANG` — if defined, its value specifies the locale to use for all purposes except as overridden by any of the variables above.

In C code, the `setlocale` function is the main means for specifying the categories to be used. It does not change the program behavior directly. Rather, the selected locale data is used by some functions from the C library. For example, the functions `strcoll` and `strxfrm` will use the sorting order defined in the Polish locale whenever the value of environment variable `LC_COLLATE` is `pl`. If the `LC_MESSAGES` value is `pl`, then the users will see Polish messages on their screen, supposing that a catalog of messages with the translations of the messages into Polish can be found.

According to the authors of the *gettext* manual, bringing NLS to a C program is an easy two step process.

1. [Internationalization.] Parameterize the program code so that it does not include specific cultural conventions in its output code and in its message strings.
2. [Localization.] Specify for each locality of users the set of cultural conventions and the catalog of message strings to be used by the program output code.

Although it could not be warranted that these two steps could be successfully performed on existing code, in the next section it will be shown how NLS can be added to the CWEB package. The paper concludes with remarks on a possible way of bringing NLS to the TeX program.

Adding NLS to the CWEB package

Look for special macro packages designed for CWEB *users in your language; or, if you are brave, write one yourself.*
— CWEB user manual

The purpose of the current section is to propose a possible way of bringing NLS to the programs `ctangle`, `cweave`, and to the TeX macro file `cwebmac.tex`; i.e., to the main components of the CWEB package. Out of several possible ways of doing that, I decided to use the ISO C model because of the existing support in the newest GNU *libc* and *gettext* libraries.

Clearly, the ISO C model could be used with the literate CWEB programs, because they are essentially C text. Therefore, a project of bringing NLS to the CWEB package appears to be feasible, but a more detailed analysis is necessary.

Let us remember what the `ctangle`/`cweave` pair of programs actually does. In literate programing, `ctangle` creates a C program and `cweave` creates a `.tex` file. The first line of the produced `.tex` file tells TeX to input the file `cwebmac.tex` with macros defining CWEB's documentation conventions. Finally `cweave` will generate a sorted cross-reference identifier index, alphabetized lists of the section names, and a table of contents. In case of errors, both programs send various clues about errors to the computer screen.

Recasting the above description in terms of the ISO model we get the following TODO list:

`LC_COLLATE` — the code responsible for sorting [§228–§239, `cweave.w`] should be changed. In particular, the collation mapping, based on the data read from the `LC_COLLATE` locale, should be created at runtime. A closer inspection of the code [§235, `cweave.w`] reveals that `LC_CTYPE` data should be used too, as the original collation mapping does not differentiate between uppercase and lowercase characters. This implies that the environment variables `LC_COLLATE` and `LC_CTYPE` should have the same value. Otherwise, we end up with a corrupted collation table with lowercase characters not mapped onto their uppercase equivalents. Therefore, we only read the value of the `LC_COLLATE` environment variable and use the value to read the `LC_CTYPE` locale data.

`LC_MESSAGES` — strings to be translated should be marked. Here, for each literate program, a separate change file should be created, with a code that initializes locale data and with strings to be translated being marked. Next, everything should be ctangled and the `xgettext` tool should be used to create an initial message catalog from the produced C sources.

`LC_MONETARY, LC_NUMERIC` — nothing to be done for these categories.

`LC_TIME` — the `\today` macro should be redefined. Otherwise, the file created by `cweave` and typeset by TeX will contain English month names. This raises the following question: how to make the expansion of the `\today` macro depend on the values of environment variables, as they are at the time when the file is being made? This question is a particular case of a more general one: how to characterize the file produced by an internationalized `cweave`?

An admissible answer could be: the created file should be able to instruct TeX which format to use for typesetting. Additionally, the file should redefine the `cwebmac.tex` macros that output the English text together with the extra fonts being used.

It is a little known fact that the current Web2C implementation of TeX makes it possible to choose the format at runtime with a `%&` line.[1] This feature appears to provide us with a way of producing such a file. It suffices that the first and second line of the file created by `cweave` are:

```
%&⟨LC_COLLATE variable value⟩
\input cweb-⟨value of LC_COLLATE variable⟩.tex
```

Unfortunately, the first line could not be created automatically, because names of format files do not reflect the language supported by the format. For example, the name `mex` does not tell us that the `mex.fmt` is the adaptation of the `plain` format to Polish conventions. Even if it were possible to create the first line automatically, the file would be typeset incorrectly — almost all characters with codes from the 128–255 range would be missing, or they would be replaced improperly. This is due to the fact that the encoding used for writing down a file differs from the internal encoding of fonts used for typesetting. For example, to typeset correctly a Latin-2 encoded file written in Polish, the file should be presented to TeX as PL-encoded, which is the internal font encoding used by the Polish Computer Modern fonts.

It should be noted, that some TeX implementors have approached the problem of such a change of encoding. For example, Eberhard Mattes in his emTeX enables the user to save re-encoding mapping in format files. The EncTeX package (the Extension of TeX for the Reencoding of the Input) by Petr Olšák provides new primitives to be used for creating re-encoding mapping to be saved in a generated format.

These non-standard extensions proliferate format files and make them depend on the file encoding. This is not necessary, because there are other ways of making re-encoding superfluous; for example, by instructing the current shell to do re-encoding. Alas, none of the shells known to the author allow such re-encoding.

Another possibility, based on a commented out TCX-code in Web2C, would be to add the following option to the `%&` line

```
-translate-file=⟨LC_COLLATE variable value⟩
```

[1] This notation is analogous to the `#!` notation used in shell scripts to tell the kernel which shell runs the script.

which tells `tex` to do input re-encoding defined in the translate file. Unfortunately, this line would be ignored by the `tex` program, as the user is not allowed to put anything on the `%&` line except just the format name.

As I am generally against the unnecessary proliferation of format files, yet another approach has been chosen. This approach does not yield, as described above, a self-contained file. The format and the re-encoding name must be written whenever `tex` is run. For example, the following command

```
tex -fmt=mex -translate-file=pl foo
```

initiates the typesetting of `foo.tex` with character codes re-mapped by a table read from the `pl.tcx` translate file. This command should be used to invoke `tex` on a Latin-2 encoded file written in Polish.

To get the behavior just described, it suffices to output the second line (from the two lines displayed on the left and above), where the file being input consists of two lines:

```
\input cwebmac.tex
\input ⟨LC_COLLATE variable value⟩-cweb.tex
```

Here, the second input file should redefine the macros that output English text. Moreover, this file could be used to add and to redefine extra fonts, because all text fonts being used share the same internal encoding.

Literate programmers should have the option of using any 8-bit character code, even in identifiers of the C program. Because there is no internationalized C compiler around, which, by definition, would allow the use of all character codes as identifiers, the authors of CWEB made `ctangle` to recognize character codes from the forbidden range 128–255 and to replace them by strings read from a default, or user-constructed, transliteration table.

All the described changes have been implemented. The result is the CWEB-NLS package. The change files of the package could be used for converting `ctangle` and `cweave` into programs that easily adapt to local conventions. Switching between different languages is achieved by setting the `LANG` environment variable to the appropriate language prior to using internationalized programs. For example, let's presume that the Polish language is requested. At the shell prompt, or from the users' startup files the following command should be executed

```
export LANG=pl
```

(in bash, or an equivalent command in other shells). If users prefer to see English messages on their screen, they should execute

```
export LC_COLLATE=pl
```

and all NLS magic will happen automatically.

Let's conclude this section with some statistics. The CWEB-NLS package consists of 9 orthogonal change files, where each file implements a different functionality. These change files could be applied with other change files extending `ctangle` and `cweave` in other ways. The total number of changed sections, from the total of 406, is 110 (around 20, if not counting the trivial changes). The 112 different messages output by `ctangle` and `cweave` have been translated into Polish. Two TeX macro files have been written to make possible a mechanical creation of localized macro files. Finally, 36 different strings output by the `cwebmac.tex` macros were translated and 7 fonts were changed too.

TeX and NLS

In this section I am going to concentrate on the possible ways of implementing message catalogs. The proposed implementation applies also to METAFONT and METAPOST. This section will conclude with a list of proposed changes to the Web2C code to make it more NLS friendly.

"Free software is going international! The Free Translation Project is a way to get maintainers of free software, translators, and users all together, so that they will gradually become able to speak many languages." — this is how the `ABOUT-NLS` file begins. I consider this GNU project very important for the reasons explained above and I am glad to see many GNU packages already speaking in Polish (see Appendix B for the current state of the project). But it came as a surprise to find traces of NLS support in Web2C. In the file `texmfmp.c`, which is a part of `tex`, `etex`, `pdftex`, `omega`, `mf`, `metapost` code, I found the statement `setlocale(LC_CTYPE,"")`. Unfortunately, this statement makes programs depend on the values of the `LC_CTYPE`, `LANG`, `LC_ALL` environment variables. In particular, the following C functions are affected: `isgraph`, `isspace`, `isprint`, `islower`, `isdigit`. Another unpleasant surprise came, when I executed the following command

```
LANG=pl tex -format-file=mex foo
```

It showed the following strange output

```
tex: nieznana opcja '-format-file=mex'
Try 'tex -help' for more information.
```

— a mixture of Polish and English. Generally, I do not like to be surprised by software in this way. Therefore I decided to examine the code. TeX is assembled from literate sources combined with various change files and hand-coded routines spread among several C source files. These files are not localized and the non-localized code is responsible for the appearance of the untranslated messages.

Unfortunately, the literate part of TeX does all of its string processing by "home-grown methods" [§38, `tex.web`] and string handling in TeX could not be localized with the GNU tools because format files play a rôle of message catalogs. To see this, let's recall the relevant facts. All the strings output by `tex` are contained in the `tex.pool` file with a check sum appended at the end. The check sum replaces the place-holder `$@` in `tex.p` — the tangled Pascal source of `tex`. When the `tex` program is preparing itself to dump a format file it reads the strings from the pool file and writes these strings on the format file. The `tex` program reads these strings from format files. Whenever `tex` is run it examines the check sums and gives up when the check sums do not match. The above description shows that removing the messages from format files requires considerable changes in the code. For that reason, I propose another approach for how to handle the translated strings.

If we ignore the check sums, then it suffices to translate all the 1309 strings from the `tex.pool` file and to update the string lengths. Now, `tex.pool` with translated strings could be used to build a format with the command[2]

```
tex -ini -translate-file=pl mex
```

It is essential to use the translation file and to recode the strings into the internal encoding of the fonts being used. Otherwise, these strings would not be re-encoded correctly when written back onto the computer screen. It should be noted that the produced format does not depend on the encoding of the TeX file. For example, it could be used to typeset CP852 encoded TeX sources written in Polish. The actual command to be used should begin with

```
tex -fmt=mex -translate-file=cp852
```

where we assume that `cp852.tcx` file contains an appropriate translation table.

It is not particularly difficult to extend the above approach to preserve the examining of check

[2] Actually, due to the way TCX files are implemented, the pool file has to be translated to the internal font encoding by other means.

sums. This would require changes in three sections of `tex.web` (§53, §1307, §1308).

Let's conclude with the promised WISH LIST:

- extend the syntax of the `%&` line to allow including user options
- localize all C sources in the Web2C directory
- implement the mapping file, analogously to `texfonts.map`, which will allow one to rename format files; the file will allow automatic generation of fully internationalized files as described above.
- translate `tex.pool`

References

Drepper, Ulrich. "Internationalization in the GNU project", 1997.

Drepper, Ulrich, Jim Meyering and François Pinard. "GNU gettext tools", May 1998.

Knuth, Donald E. "Literate Programming", *The Computer Journal* **27** (1984), pp. 97–111.

Knuth, Donald E., and Silvio Levy. "The CWEB System of Structured Documentation", version 3.0. `cwebman.tex` file from the CWEB package.

`ABOUT-NLS`, file available on most GNU archive sites.

Appendix A

Someone writes in `tex.ch`: ,,TCX files are probably a bad idea, since they make TEX source documents unportable. Try the `inputenc` LATEX package."

I think that `inputenc` package does not provide the functionality offered by TCX files for the following reasons:

- the `\uppercase` primitive does not work
- commands names which use diacritical characters could not be defined
- the `inputenc` package is usable for LATEX only — other formats are not supported (there are already 42 different formats on the TEX Live 3 CD-ROM)
- the log file and terminal are not readable because unreadable '`^^`' notation is used — see the example below.

```
Niewypelnione \hbox (lichość 10000)
   znaleziono w linii 2
\rm kość
\hbox(6.88889+0.0)x44.0
.\rm k
.\kern-0.27779
.\rm o
.\rm ś
.\rm ć

[1] )
Wyjście zapisane do foo.dvi
   (1 strona, 212 bajtów)
```

```
Niewype^^c2nione \hbox (licho^^c9^^b8 10000)
   znaleziono w linii 2
\tenrm ko^^b1^^a2
\hbox(6.88889+0.0)x44.0
.\tenrm k
.\kern-0.27779
.\tenrm o
.\tenrm ^^b1
.\tenrm ^^a2

[1] )
Wyj^^c9cie zapisane do foo.dvi
   (1 strona, 212 bajt^^f3w)
```

Appendix B

The following matrix shows, for several countries, the current state of internationalization in the GNU project, as of May 1998. The following matrix shows, with regard to each package listed, the languages in which message catalogs have already been submitted.

Ready PO files	cs	de	fr	nl	no	pl	ru	sl	sv
bash		▪	▪	▪					
bison		▪	▪	▪					
clisp		▪	▪						
cpio		▪	▪	▪		▪			
diffutils		▪	▪			▪			▪
enscript		▪	▪	▪				▪	
fileutils	▪	▪	▪	▪		▪		▪	▪
findutils		▪	▪	▪		▪	▪		▪
flex			▪						▪
gcal		▪	▪	▪		▪			▪
gettext		▪	▪	▪	▪	▪		▪	▪
grep		▪	▪	▪	▪	▪	▪	▪	▪
hello		▪	▪	▪	▪	▪		▪	▪
id-utils		▪	▪			▪			
indent		▪				▪	▪		
libc		▪	▪	▪		▪			▪
m4		▪	▪	▪			▪		▪
make		▪	▪	▪		▪			
music			▪			▪			
ptx		▪	▪	▪	▪	▪			▪
recode		▪	▪	▪		▪		▪	▪
sh-utils		▪	▪	▪	▪	▪			▪
sharutils	▪	▪	▪	▪					▪
tar	▪	▪	▪	▪	▪	▪		▪	▪
texinfo	▪	▪	▪						
textutils	▪	▪	▪	▪	▪	▪			▪
wdiff	▪	▪	▪	▪	▪	▪			▪