New CSplain of 2012

Petr Olšák

The CSplain package has existed since 1994 and it is a *gentle* extension of plain TEX to make using Czech and Slovak languages feasible. This was the case until October 2012, when the author carried out significant revisions and additions to CSplain. The basic change resulted from the decision to set the default input encoding of CSplain to UTF-8. In addition, CSplain got many other new features: the possibility of loading all available hyphenation patterns, the ability to cooperate with 16-bit TEX engines (LuaTEX, XƎTEX), more effective work with fonts including math, easy switching of the internal encoding (including Unicode), and the user-friendly macros OPmac.

In the default configuration, CSplain remains a gentle extension of plain TEX, backwards-compatible with previous versions. The new possibilities are easily accessed with `\input` and when they are used it is no longer correct to talk of a *gentle* extension. On the contrary, it is a strong competitor to all other macro systems based on TEX, even very large ones. CSplain has advantages in its simplicity, effectiveness, and ease of usage.

The new CSplain is available through CTAN and the usual TEX distributions, and its home on the web is `http://petr.olsak.net/csplain-e.html`.

Introduction

In October 2012, a discussion was held on the `cstex@` mailing list about the configuration of the input encoding of CSplain. It was shown that for many years CSplain used the wrong default input encoding on MS Windows: ISO 8859-2, which is foreign on this operating system. I was surprised.

Our old decision was that the input encoding of CSplain was to be set depending on the operating system in use. This is similar to the ASCII versus EBCDIC encodings on old systems, where TEX did reencoding of its input depending on its environment. It is essential that when the Czech and Slovak characters in the source file are shown correctly in the text editor then CSplain prints them correctly too. On the other hand, when we see bad characters in the text editor, we cannot wonder that CSplain produces broken output. Unfortunately, this idea was valid ten years ago, but not so much today. Nowadays there are text editors with special intelligence — they try to autodetect the encoding and they try to show anything properly. In such an environment, the above rule makes no sense. These modern editors handle the UTF-8 encoding, so we decided that this will be implicitly set as the input encoding of CSplain on all systems.

The conversion between UTF-8 input codes and the internal encoding (i.e. font encoding and hyphenation pattern encoding) must be done straightforwardly at the input processor level. No active characters are allowed for this purpose. When we do

```
\def\test#1#2%
   {the first character is #1, second is #2}
\test čř
```

then we expect the output "the first character is č, second is ř". Therefore, CSplain needs to activate the encTEX extension in 8-bit TEX engines (TEX, pdfTEX). The 16-bit TEX engines are more straightforwardly used for this case.

Format generation

The following lines show various methods to generate the format files `csplain` and `pdfcsplain`. The implicit output (DVI and PDF) is set by the name of generated format (`csplain` sets DVI output, while `pdfcsplain` sets PDF output).

```
pdftex -ini -enc "\let\enc=u \input csplain.ini"
pdftex -jobname csplain -ini -etex \
       -enc csplain-utf8.ini
pdftex -jobname pdfcsplain -ini -etex \
       -enc csplain-utf8.ini
xetex  -jobname pdfcsplain -etex -ini csplain.ini
luatex -jobname pdfcsplain -ini csplain.ini
```

CSplain — basic features

The basic behavior of CSplain is similar to plain TEX. The only difference is that the default `\hsize` and `\vsize` are set to create one inch margins in A4 paper format, not letter format. One can consider that the second difference is the presence of macros unknown in plain TEX:

```
\chyph      % Czech hyphenation patterns and
            % \frenchspacing initialised.
\shyph      % Slovak hyphenation patterns and
            % \frenchspacing initialised.
\csaccents  % redefines \' \v \^ \' \" \r
            % to expand to given internal slot.
```

You can return to the default behavior with:

```
\ehyph      % US hyphenation patterns and
            % \nonfrenchspacing.
\cmaccents  % \', \v etc. expand to
            % \accent primitive.
```

The implicit internal encoding and the implicit fonts are set to CSencoding/CSfonts in CSplain. It means that (for example) the font `csr10` is preloaded as `\tenrm` instead of `cmr10`. These `cs*` fonts keep the 7-bit half of the encoding table the same as their `cm*` counterparts, while Czech and Slovak letters are placed in the second part of encoding table, ordered by ISO-8859-2.

CSplain defines control sequences which correspond to the special glyphs used in CSfonts.

```
\clqq       % left Czech double quote.
\crqq       % right Czech double quote.
\flqq       % left French double quote
            % (used at right side in Czech).
\frqq       % right French double quote
            % (used at left side in Czech).
\promile    % per mille character.
\uv         % quotation macro: \uv{text} gives
            % \clqq text\crqq.
\ogonek a   % Polish a-ogonek
            % (composed from components)
```

UTF-8 input encoding when encTEX is used

You can recognize the UTF-8 encoded CSplain with encTEX by the message:

```
The format: csplain <Nov. 2012>.
The cs-fonts are preloaded and A4 size
    implicitly defined.
The utf8->iso8859-2 re-encoding of Czech+Slovak
    alphabet activated by encTeX
```

Many thousands of character codes can occur in UTF-8 input, but by default, CSplain is able to read only characters from ASCII and the Czech and Slovak alphabets:

Á á Ä ä Č č Ď ď É é Ě ě Í í Ĺ ĺ Ľ ľ Ň ň Ó ó Ö ö Ô ô Ŕ ŕ Ř ř Š š Ť ť Ú ú Ů ů Ü ü Ý ý Ž ž.

These characters are mapped by encTEX to one byte (one slot) corresponding to the internal encoding. Moreover, the characters known from plain TEX are mapped to the control sequences:

```
plain: \ss ß, \l, \L, \ae æ, \oe œ, \AE Æ, \OE Œ,
       \o ø, \O Ø, \i ı, \j ȷ, \aa å, \AA Å,
       \S §, \P ¶, \copyright ©, \dots ...,
       \dag †, \ddag ‡.
csplain: \clqq, \crqq, \flqq, \frqq, \promile.
```

EncTEX is able to map the UTF-8 code to the internal 8-bit slot or to the control sequence. When such a mapped control sequence or internal 8-bit slot is processed by the `\write` primitive, it is converted back to the UTF-8 code. So, the 8-bit TEX engine can handle an unlimited number of UTF-8 codes. But by default, only the characters mentioned above are properly processed by CSplain. If another UTF-8 code occurs in the input, CSplain reports the following warning (the Ñ character is used in this example):

```
WARNING: unknown UTF-8 code: ‘Ñ = ^^c3^^91’
        (line: 42)
```

and users can add their own mapping and definition of such a character. For example:

```
\mubyte\Ntilde ^^c3^^91\endmubyte
                 % \UTF-8 code mapped to \Ntilde.
\def\Ntilde{\~N} % The \Ntilde is defined.
```

Now CSplain processes the Ñ character properly even though it is not included in the Czech or Slovak alphabets.

The distribution `enctex.tar.gz` contains these two files:

```
utf8lat1.tex % Latin1 Supplement U+0080-U+00FF
utf8lata.tex % Latin Extended-A  U+0100-U+017F
```

These files do the mapping of the abovementioned UTF-8 codes by encTEX and provide the definitions for the mapped control sequences. You can `\input` them to your document and/or create analogous files for your purposes.

Internal encoding

The internal encoding means the encoding of the fonts and hyphenation patterns that are used. By default, CSplain sets the internal encoding to the CS-encoding (as mentioned above). But you can change this encoding via `\input` at the beginning of your document. There are two possibilities:

```
\input t1code % the T1 internal encoding is set
\input ucode  % the Unicode internal encoding
              % is set (in 16-bit TeX engines)
```

These `\input` files do the following:

- Set the correct `\uccode`/`\lccode`.
- Reset the `\chyph` and `\shyph` macros, so they choose the hyphenation patterns in proper encoding.
- Remap the UTF-8 codes to the new slots, if encTEX is used.
- Redefine some character-like control sequences (`\ss`, etc.).
- Redefine `\csaccents`, so `\'x`, `\v x`, etc. expand to the right slots.

As you can see, these files don't reload the fonts with the proper encoding. This has to be done with the next `\input` in your document, for example `\input lmfonts` or `ctimes` or `cs-pagella`.

CSplain preloads the Czech and Slovak hyphenation patterns in CS-encoding, in T1 encoding and (if a 16-bit TeX engine is detected) in Unicode. The only thing the user need be concerned with is initializing the hyphenation patterns with `\chyph` or `\shyph` *after* the `\input t1code` or `\input ucode` is done. The section below, "More languages", describes how CSplain is able to load hyphenation patterns of another languages.

Font loading

The CSplain package provides the following ready-to-use files which load the given font family (typically `\rm`, `\it`, `\bf` and `\bi`):

```
lmfonts     % Latin Modern fonts
ctimes      % Times
chelvet     % Helvetica
cavantga    % AvantGarde
cncent      % NewCentury
cpalatin    % Palatino
cs-termes   % TeX-Gyre Termes (Times)
cs-heros    % TeX-Gyre Heros (Helvetica)
cs-cursor   % TeX-Gyre Cursor (Courier)
cs-adventor % TeX-Gyre Adventor (AvantGarde)
cs-bonum    % TeX-Gyre Bonum (Bookman)
cs-pagella  % TeX-Gyre Pagella (Palatino)
cs-schola   % TeX-Gyre Schola (NewCentury)
cs-antt     % Antykwa Torunska
cs-polta    % Antykwa Poltawskiego
cs-bera     % Bera
cs-arev     % ArevSans
cs-charter  % Charter
```

All of these font files include the switch to load the correct font for the chosen internal encoding (CS-encoding or T1 or Unicode). These font files simply load the fonts for the needed variants with the `\font` primitive, redefining the control sequences `\tenrm`, `\tenit`, `\tenbf`, `\tenbi` and `\tentt`. Again, users can easily create their own additional font files by using these as a model.

The font loading files do not deal with the various sizes of the fonts, because they do not need to. That is the subject of the next section.

Font handling

CSplain introduces a simple font-resizing principle. The main credo is: "power is in simplicity". That is the reason why I don't use NFSS, for example.

The command `\font\foo=something` declares *font selector* `\foo` which selects the font `something`. The terminology *font selector* in this section is used only for selectors declared by the `\font` primitive. This means that `\bf` (for example) isn't a font selector. It is a macro.

CSplain defines the following macros for font size handling.

- `\resizefont\foo` resizes the font represented by font selector `\foo`. More precisely, it declares (locally) `\foo` as the same font but with the size given in the macro `\sizespec`. The `\sizespec` macro can have the form `at`⟨*dimen*⟩ or `scale`⟨*factor*⟩.
- `\regfont\foo` registers the font selector `\foo` as a resizable font. By default CSplain declares the following selectors with `\regfont`: `\tenrm`, `\tenit`, `\tenbf`, `\tenbi` and `\tentt`. Users can declare more selectors.
- `\resizeall` resizes (locally) all registered font selectors to the size given by the `\sizespec` macro.
- `\letfont \foo=\bar at`⟨*dimen*⟩ or `\letfont \foo=\bar scaled`⟨*factor*⟩ declares a new font selector `\foo` as the same font as `\bar` with the given size. The `\bar` font selector is unchanged.

Here's an example:

```
\font\zapfchan=pzcmi8z  \regfont\zapfchan
\def\sizespec{at13.5pt} \resizeall \tenrm
\baselineskip=15pt

Here is the typesetting at size 13.5pt
including {\it italics}, {\bf bold} and
including the {\zapfchan Zapf Chancery font}.

\def\sizespec{at8pt} \resizeall \tenrm
Now all the typesetting is at the 8pt size.
```

Another example uses the font loading files:

```
\input chelvet % \tenrm, \tenit, etc. is now
               % the Helvetica family.
\letfont\titlefont = \tenbf at14.4pt
               % \titlefont is for titles:
               % Helvetica Bold at14,4pt.
\input ctimes  % \tenrm, etc. is Times Roman.
\def\sizespec{at11pt}\resizeall \tenrm
               % Normal text will be typeset
               % by Times Roman at11pt.
\def\small{\def\sizespec{at9pt}\resizeall \tenrm}
   % The \small macro switches the whole family
   % of Times Roman to the 9pt size,
   % e.g., for footnotes.
```

Note #1. The font selectors `\tenrm`, `\tenit`, etc. have the subword `ten` in its name but this is only for historical reasons. The current meaning of these selectors can be fonts at an arbitrary size.

Note #2. These macros do not solve the resizing of math fonts. This is the subject of the following section.

Note #3. The selection of the proper design size (`cmr5` or `cmr7` or ... or `cmr17`) is not solved by default. But the math font macros solve this and you can simply redefine `\resizefont` so that the proper design size is selected.

Math fonts

The CSplain package provides two macro files for math fonts: `ams-math.tex` and `tx-math.tex`. The first one loads AMS fonts and declares hundreds of math symbols and operators like AMSTeX. The second macro file does the same but loads the `tx` fonts which are visually compatible with Times Roman and similar designs.

By default, neither of these macro files are read. But you can load `ams-math.tex` explicitly, or the proper macro file is loaded implicitly with `\input ctimes`, `lmfonts`, etc.

These files provide the macro:

`\setmathsizes[`⟨*text*⟩`/`⟨*script*⟩`/`⟨*scriptscript*⟩`]`

in which the user can set the sizes of basic text, script and superscript. The parameters have to be written without unit (the unit pt is used). For example `\setmathsizes[10/7/5]` is the default from plain TeX.

The following math alphabets are available after `ams-math.tex` or `tx-math.tex` is loaded:

```
\mit      % mathematical variables
\rm, \it  % text fonts in math
\bf, \bi  % bold sans fonts (might be
          %    different than text fonts)
\cal      % normal calligraphic
\script   % script
\frak     % fraktur
\bbchar   % double stroked letters
```

The `ams-math.tex` defines the `\regtfm` macro to declare the mapping from a desired size to the list of design sizes represented by names of the metric files. For more information about this, see the file `ams-math.tex`, where `\regtfm` is defined and used. Once this mapping is set, you can redefine the internal subpart of the `\resizefont` macro in the following way:

```
\def\resizefontskipat#1 #2\relax
       {\whichtfm{#1} \sizespec\relax}
```

Now `\resizefont` chooses the right metrics if `\sizespec` and `\dgsize` are properly set. This complexity can be hidden from the user, if he or she uses the `\typosize` and `\typoscale` macros from OPmac.

The following example shows how to set the font for a title that includes math formulas:

```
\def\titlefont{\def{at14pt}\resizefont\tenbf
   \tenbf \setmathsizes[14/9.8/7]\boldmath}
\def\title#1\par{\centerline{\titlefont #1}}

\title More about $\int_x^\infty f(t){\rm d}t$
```

The `\boldmath` command selects the alternative set of all math families more compatible with **bold** fonts usually used in titles.

Unicode fonts

Historically, CSplain worked with 8-bit TeX engines where Unicode fonts are impossible. So, all the font handling mentioned so far is primarily intended for 8-bit fonts. The Unicode support for text fonts in CSplain is only experimental, and Unicode math isn't solved in CSplain at all.

The 16-bit TeX engines expect the UTF-8 input encoding and work in Unicode internally. So T1-encoded fonts cannot be used because Czech and Slovak alphabets are unfortunately not in the intersection of T1 and Unicode encodings. On the other hand, colleagues writing in German or French can use T1-encoded 8-bit fonts in 16-bit TeX engines because their whole alphabet is in this intersection.

XeTeX has a font loader linked with system libraries and it extends the syntax of the `\font` primitive. For example:

`\font\foo="[`⟨*filename*⟩`]:`⟨*fontfeatures*⟩`"` ⟨*sizespec*⟩

where ⟨*filename*⟩ is the file name without the `.otf` suffix and the ⟨*sizespec*⟩ is `at`⟨*dimen*⟩ or `scaled` ⟨*factor*⟩. The ⟨*fontfeatures*⟩ are font modifiers separated by semicolon. You have to know which features are implemented in the font and which in the font loader. For example, XeTeX's font loader provides the feature `mapping=tex-text` which activates the usual TeX ligatures like `--` → –. The normal ligatures (e.g., 'fi') are activated implicitly.

On the other hand, LuaTeX implements its extension of the font loader by Lua code. I have extracted the core of this code (from `luaotfload.sty`) for CSplain, in a file `luafonts.tex`. Its stability can't be guaranteed because the Lua functions from the LuaTeX distribution are called, and they may change in the future. If LuaTeX is being used, the files `lmfonts.tex`, `cs-termes.tex`, `cs-heros.tex`, etc. input `luafonts.tex` before the first usage of the extended `\font` primitive.

The extension of the `\font` primitive seems to have the same syntax in XeTeX and LuaTeX. But, unfortunately, the font features are different. By default, no ligatures are activated in Unicode fonts in LuaTeX. Users must use `script=latn` to activate the fi-ligatures and `+tlig` to activate the TeX special ligatures. Users can define the `\fontfeatures` macro for special needs of features. If this macro isn't defined, CSplain's font-loading macros make the following default:

```
\def\fontfeatures
       {mapping=tex-text;script=latn;+tlig}
```

which works in both XƎTEX and LuaTEX.

More languages

The following hyphenation patterns are preloaded in CSplain by default:

- `\USenglish=0` ... default US hyphenation patterns from plain TEX, ASCII encoding.
- `\czILtwo=5` ... Czech patterns, ISO-8859-2.
- `\skILtwo=6` ... Slovak patterns, ISO-8859-2.
- `\czCork=15` ... Czech patterns, T1 encoding.
- `\skCork=16` ... Slovak patterns, T1 encoding.
- `\czUnicode=115` ... Czech patterns, Unicode (only for 16-bit TEX engines).
- `\skUnicode=116` ... Slovak patterns, Unicode (only for 16-bit TEX engine).

Hyphenation patterns are selected with `\uslang`, `\czlang` and `\sklang`, which are equivalent to the old selectors `\ehyph`, `\chyph` and `\shyph`. The proper encoding is used if the command `\input t1code` or `\input ucode` precedes the patterns selector.

Since 2012, CSplain is able to load hyphenation patterns of other languages (ca. 50 languages). If the patterns use a subset of T1 encoding, they can be loaded in T1 (alias Cork) and/or in Unicode. Otherwise, only the Unicode encoding for the patterns is allowed. Unicode patterns can be loaded only in 16-bit TEX engines.

The loading of extra hyphenation patterns can be done on the command line when format is generated. Examples follow:

```
pdftex -ini -enc \
   "\let\plCork=y \let\enc=u \input csplain.ini"
pdftex -ini -enc "\let\allpatterns=y
   \let\enc=u \input csplain.ini"
luatex -jobname pdfcsplain -ini \
   "\let\ruUnicode=y \input csplain.ini"
luatex -jobname pdfcsplain -ini \
   "\let\allpatterns=y \input csplain.ini"
```

The first line adds Polish hyphenation patterns in the T1 encoding to CSplain. The second line loads all available hyphenation patterns for 8-bit TEX engines (i.e. Czech&Slovak in ISO-8859-2 and T1, and others, ca. 30 languages, in T1). The third line loads the Russian hyphenation patterns in Unicode. Finally, the last line loads all available hyphenation patterns (in T1 and in Unicode). The pattern selectors have the form `\`⟨*twoletters*⟩`lang`, for example `\pllang`, `\delang`, `\itlang`, `\rulang` etc. Please read the `hyphen.lan` file for more information.

The OPmac macro package

The OPmac (Olsak's Plain macros) package is part of CSplain. It provides more LATEX-like features in plain TEX: font size changing, automatic creation of tables of contents and indexes, working with bibliography databases, tables, references including hyperlinks options, etc. For more information about this macro package, see the companion article in this same issue of *TUGboat*.

◇ Petr Olšák
Czech Technical University
in Prague
Czech Republic