Philology

Configuring TEX or LATEX for typesetting in several languages

Claudio Beccari

Abstract

Based on the frequent requests for help that I have received in the past months from users who want to configure TEX or LATEX for typesetting in several languages, it seems that this sort of information is not adequately covered in the documentation available to most do-it-yourself users. This tutorial is intended to give basic information on this topic. The related subject of hyphenation patterns will be addressed in a future tutorial.

1 Introduction

LATEX users are excused for not knowing how to set up their favorite typesetting program for different languages; Lamport [5] doesn't say a word on this subject; the new edition invites the reader to consult Goossens et al. [3] for what concerns multilanguage typesetting, and certainly the latter contains several hints — more than simple hints, since all of chapter 9 is dedicated to this problem.

TEX users should be more familiar with this problem via the main reference, Knuth [4], which covers the topic of hyphenation patterns. Appendix H stresses that TEX hyphenation capabilities are generated by the `\pattern` and the `\hyphenation` commands, the former being processable only by the initialization TEX program `initex`.

J. Braams prepared a complete system of style files and macros for use with LATEX, called babel; officially, it works only with standard document styles, but in practice, it is also valid for other styles and includes adequate means for setting text into type in some twenty languages. The only part that is lacking from the babel system is the set of hyphenation patterns for each of those languages, but this was done on purpose, I suppose, because pattern preparation, although essential for multilanguage typesetting, has almost nothing to do with the *style* of multilanguage typesetting.

This tutorial will attempt to explain to do-it-yourself (LA)TEX users how to configure their systems in order to set text into type in different languages at the same time. It is natural that this issue should be particularly interesting for non-English-speaking (LA)TEX users, but I have received requests for help from the United States as well, so I presume that across the ocean there are also some people who may benefit from these simple notes.

Typesetting text in several languages implies the following problems: (a) correct hyphenation, (b) correct "labels" in titles and captions ("Chapter", "Capitolo", "Chapitre", etc.), (c) special language-dependent typesetting rules (French vs. non-French spacing, quotation marks, etc.). Therefore the points to be covered include:

1. choosing the proper fonts and font encodings;
2. retrieving or creating hyphenation patterns;
3. initializing TEX or LATEX, taking into account the program's memory limitations;
4. retrieving or creating macros for switching languages;
5. language-dependent typesetting macros.

Here I will skip over the problem of typesetting with alphabets different from the extended Latin one; for Greek (modern and ancient) there is plenty of information (TEX and METAFONT files, the latter for generating the full set of characters and ligatures) in the CTAN directories

```
/tex-archive/languages/greek/levy
/tex-archive/languages/greek/yannis
```

The former contains the full set of 256-glyph fonts and TEX macros for handling them, the latter contains also the 128-glyph fonts, TEX macros and hyphenation patterns; both contain excellent documentation for using Greek fonts and for setting ancient and modern Greek.

The platforms (or boxes, as TEXies seem to call their computers) on which people set their texts are of widely different types, with different operating systems and, in particular, with different file systems. I'll address myself mainly to the DOS type of personal computer users, because apparently this category counts the highest number of do-it-yourself users; with minor modifications what I'll say is applicable also to other environments, with the caution that with VMS one must have system manager privileges and with UNIX, superuser privileges. In any case I assume that the do-it-yourself reader who is going to use this information in practice is sufficiently familiar with his/her computer to be able to use the available utilities for exploring and managing the hard disk(s) and the file system, for creating command (or batch or script) files, and so on.

As for myself, I use a VMS mainframe, a UNIX workstation and a DOS personal computer; the latter is a 486 type and has 8 Mb of RAM, so that, even though I don't use Windows, with a suitable

extended memory handler I can use a "big TEX" implementation of TEX based on the compilation of a C-source program; I recommend that any DOS user equip him/herself with a configuration of this sort. My LATEX is set up to deal with eight languages at a time: US-English, Italian, French, Spanish, Portuguese, Catalan, Romanian, and Latin (modern spelling). Except for US-English, I prepared all the hyphenation patterns for the remaining seven languages myself.

2 Fonts

Except for English (both US and UK), which does not use diacritics (actually it does when assimilated foreign words are used), and textual Latin (i.e. excluding prosodic Latin), almost all the languages that were examined by Sojka and Ševeček [6] use diacritics. (LA)TEX has no problem in setting suitable diacritics over or under any letter, but (LA)TEX has real problems in hyphenating words that contain the control symbols or control words that are used for setting such diacritics. In fact, TEX "words" are not the same as the words of a language. Loosely speaking, for TEX a "word" is a sequence of characters of category code 11 (letter) or 12 (other), with a non-zero `\lccode`, set in the same font, that is preceded by a punctuation mark (parenthesis, quotation marks, etc.) or by space or glue, and ends with anything different from the above-mentioned characters. Any command that interrupts this sequence interrupts the word, unless it expands to one or more characters with the proper characteristics. For a precise definition of what TEX thinks a word is, see Appendix H of *The TEXbook*.

As examples, the French word `\'ecole` = *école* is not a word at all for TEX because the accent command `\'` expands to `\accent 19`, unless ... Even the English word `{\it school}` = *school* is not a word for TEX, because there is no space or glue between the command `\it` and the word 'school'; this is why (LA)TEX often reports overfull hboxes when you emphasize some text.

Unless ... yes, there is a way around this: extended fonts and a simple little macro, `\hz`, for inserting glue where there should be no extra space:

```
\def\hz{\nobreak\hskip0pt \relax}
```

In fact, `\hz` inserts an unbreakable glob of glue of zero width, stretch and shrink, but nevertheless it is glue, so that after this glob a real TEX word may begin, a word that TEX can hyphenate properly. Use this little macro and you'll get rid of many hyphenation problems. For example, if you type

```
\emph{\hz electricity}
```

you get *electricity* properly hyphenated even though it is emphasized. If you want to typeset some French words within an English text, and you only have the standard cm fonts, besides inserting a lot of discretionary breaks `\-` at every syllable (if you don't have the French hyphenation patterns), you have the possibility of inserting `\hz` in order to convince TEX to deal with word fragments; for example, you can type

```
\'e\hz lectricit\'e
```

and TEX will try to hyphenate the fragment 'lectricit', probably finding some correct hyphen points even by using English patterns.

But the real solution lies in using the extended fonts. There are two flavors: the real dc fonts and the virtual em fonts. The former are complete sets of 256 glyphs that conform to the "double Cork" encoding, while the latter are virtual fonts that are made up with pieces taken from real 128-glyph cm fonts. Although em fonts lack some glyphs, compared with real dc fonts, they sometimes are more flexible than dc fonts since by editing the virtual property list file it is possible to modify them very easily. Moreover the `.pk` files for the dc fonts in the customary 300, 329, ..., 746 dots-per-inch sizes occupy approximately 7 Mb of disk space, which might not be available on the smaller platforms or which might be used in a different way.

If you already have the dc `.tfm` and pixel files on your computer disk, or if you are willing to copy them from a CTAN archive, skip the next section; otherwise, you might find it interesting to create the full set of em fonts yourself, as explained below.

2.1 Creating standard virtual em fonts

Examine your file system and find out if you already have `.tfm` files whose names start with `em` and if you have files with the same name but extension `.vf`; if you do, skip to the next section.

If you don't, you might be in trouble, but before giving up examine your file system and find out if you have executable files (extension `.exe`) with the following names: `tftovp`, `vftovp`, `vptovf`, `pltotf`, `tftopl`. If you do, they probably came with your screen and/or printer driver, unless you are the type of hacker who copies everything in the hope that it might become useful at some future date.

This point is crucial; in fact, if you got these files with your drivers, you can be almost certain that your drivers handle virtual fonts. Check the driver documentation; if your drivers actually handle virtual fonts, keep reading this section; otherwise, go to the next subsection. There is no sense

	’0	’1	’2	’3	’4	’5	’6	’7	
’20	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ	"8
’21	Ĺ	L’	Ł	Ń	Ň		Ő	Ŕ	
’22	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů	"9
’23	Ÿ	Ź	Ž	Ż	IJ	İ	đ	§	
’24	ă	ą	ć	č	d’	ě	ę	ğ	"A
’25	Í	l’	ł	ń	ň		ő	ŕ	
’26	ř	ś	š	ş	t’	ţ	ű	ů	"B
’27	ÿ	ź	ž	ż	ij	¡	¿		
’30	À	Á	Â	Ã	Ä	Å	Æ	Ç	"C
’31	È	É	Ê	Ë	Ì	Í	Î	Ï	
’32		Ñ	Ò	Ó	Ô	Õ	Ö	Œ	"D
’33	Ø	Ù	Ú	Û	Ü	Ý		SS	
’34	à	á	â	ã	ä	å	æ	ç	"E
’35	è	é	ê	ë	ì	í	î	ï	
’36		ñ	ò	ó	ô	õ	ö	œ	"F
’37	ø	ù	ú	û	ü	ý		ß	
	"8	"9	"A	"B	"C	"D	"E	"F	

Table 1: Font layout for em fonts with character codes above 127. The empty positions should contain the lower- and uppercase versions of ‘eth’ and ‘thorn’, the ‘nj’ ligature, and a non-slanting version of the pound sterling sign; the dc fonts have these positions filled up.

in creating virtual fonts if your drivers can’t handle them.

If you check in *The TEXbook*, Appendix F, you may realize that the roman, italic and typewriter fonts have different layouts; therefore, when you create virtual fonts you must give your programs this kind of information. Just to give an example let’s create the virtual font `emr10` starting from the standard real font `cmr10`:

1. Run `tftovp` by issuing the following command

   ```
   tftovp -rm cmr10.tfm emr10.vpl
   ```

   after having set things up so that the necessary `.tfm` files are in the default directory; the best thing to do is to issue the command in the directory where you have all the `.tfm` files.

   This action creates a virtual property list file, with extension `.vpl`, that contains all the information on the size of every character, the ligatures, the glyphs that are made by superposition of glyphs taken from several other real fonts. Of course, for other cases you might be obliged to use the command `tftovp` in its full glory with all the other options fully spelled out, but this simple example is sufficient for giving the idea of the whole procedure.[1]

2. Now run `vptotf` in this way

   ```
   vptotf emr10
   ```

   obtaining the `.tfm` file (LA)TEX needs for its font selection and the virtual font file (extension `.vf`) that the driver needs for using the virtual font; move this latter file into the directory where the driver(s) expect to find virtual files.

3. The `.vpl` file is no longer needed, so it can be deleted.

Of course you might automate this simple procedure by writing a command (or script) file that performs all these operations with a minimum of human intervention.

If you have `.afm` files for PostScript fonts, you can do similar operations in order to use such outline fonts, but maybe leave that for when you have more experience.

2.2 Getting along with ordinary cm fonts

If you do not have the programs mentioned in the previous subsection and/or your drivers do not handle virtual fonts or do not handle fonts with more than 128 characters, or if you have decided that you do not want to use virtual fonts (for example for portability reasons), do not give up! It is still possible to redefine the accent macros so as to make them a little smarter, i.e. so that they introduce automatically an `\hz` command before and/or after the letter they operate upon, depending on the nature of the following character.

You should be able to perform an anonymous `ftp` to my site:

```
ftp ftp.polito.it
Username:  anonymous
Password:  your e-mail address
cd /pub/tex/polito/hyphens
```

where you can fetch the three files `polyglot.tex`, `polyglot.sty` and `polyglot.doc`. You can use the

[1] Apparently nobody noticed or took care that none of the options available for the `tftovp` command handles the caps-and-small-caps fonts; for such fonts, some editing of the ASCII virtual property list file is necessary before going to the next step, but you’ll produce virtual caps-and-small-caps fonts when you have gained a little experience with virtual fonts. It’s a good idea to use your editor to explore the property list files: you get to understand a lot of things about TEX that are not written in any book, or are presented in such a way that the reader can’t really understand them.

second one as a LaTeX option, so that you can process the third one as a LaTeX document and get all the information about the use and/or initialization of your (La)TeX programs with the `polyglot` macros.

In the same directory you will find the "poor man" hyphenation patterns for several languages; these are labeled "poor man" because they do not consider accented letters and leave the task of hyphenating to the "intelligent" accent macros. Such macros do their best, but of course they cannot perform as well as a (La)TeX system correctly set up with dc or em fonts for handling multiple languages. Nevertheless, I did use such a "poor man" implementation for a while, and I have typeset several documents in different languages with almost no human intervention while getting error-free justified text with hyphenated words, in particular in French and in Catalan.[2]

3 Using em or dc fonts

3.1 Extended fonts and TeX

If you use TeX, and you want to use em or dc fonts, you should do the following:

1. copy the file `plain.tex` to another file, and call it `emplain.tex`;
2. edit `emplain.tex` so that it also preloads the roman, italic, etc., em or dc fonts corresponding to the roman, italic, etc., fonts already loaded; for example, add the lines:

   ```
   \font\etenrm = dcr10
   \font\etensl = dcsl10
   ...
   ```

   or

   ```
   \font\etenrm = emr10
   \font\etensl = emsl10
   ...
   ```

3. modify the definitions for `\rm`, `\sl`, `\it`

   ```
   \def\rm{\fam\z@\etenrm}
   \def\sl{\fam\slfam\etensl}
   \def\it{\fam\itfam\etenit}
   ...
   ```

3.2 Extended fonts and LaTeX 2ε

LaTeX 2ε is already pre-set for use with dc fonts, although the default encoding scheme is the "old" 128-glyph one. Therefore, before setting any text with dc fonts it is necessary to declare the extended `T1` encoding by means of the declaration

```
\renewcommand{\encodingdefault}{T1}
```

in the preamble of your document.

But the above operation is suitable only when you run LaTeX 2ε as a regular program that has already been initialized. Its ability to handle hyphenation patterns in one or more languages derives from its initialization which must be executed according to the procedure described for your particular implementation of TeX; in the following sections, an example is given, but your particular implementation might require some variations.

Regardless of the implementation, though, you need to set up or modify a file named `hyphen.cfg` which is intended for loading several hyphenation pattern files for the corresponding languages. As far as my experience is concerned, the LaTeX 2ε base package obtainable from CTAN states that this task is reserved for TeXperts and a special documentation file is offered to such experts for the task. Unfortunately this file does not say much and several actions must be invented by the user. Two points are to be followed with special care:

1. The command `\newlanguage` is defined to be an `\outer` one so that it cannot be used within the main argument of the command

   ```
   \InputIfFileExists{<A>}{<B>}{<C>}
   ```

   where `<A>` is the file name containing the patterns for a particular language, `<B>` the actions to be taken before inputting the pattern file, and `<C>` the actions to be taken if the pattern file does not exist or can't be found along the default or the specified paths. Therefore, in order to load French patterns, for example, it is necessary to spell out the loading commands:

   ```
   \newlanguage\l@french
   \InputIfFileExists{frhyph}{%
   \language\l@french
   \lccode`\'=`\'
   % etcetera
   }{%
   \errhelp{The configuration for
       hyphenation is incorrectly
       installed.^^J%
       If you don't understand
       this error message you need
       to seek^^Jexpert advice.}%
   \errmessage{OOPS! I can't find
       any hyphenation patterns for
       French.^^J \space Think of
       getting some, or the latex2e
   ```

[2] I mention these two languages (of the eight that my box can handle) because they were used to typeset formal and official documents undersigned by the Rectors/Directors of the Polytechnics with which we made agreements.

```
    setup will never succeed.}%
    \@@end
}
```

(The actions to be taken for a nonexistent or unreachable French pattern file are copied [and slightly edited] from the sample `hyphen.cfg` file that is included in the base package.)

2. Before inputting any pattern file that contains special characters, it is necessary to map the accent macros and their arguments to the charcodes of the corresponding special characters and to define their lowercase codes.

   In fact, the ability of LaTeX 2ε to deal with extended codes through sophisticated accent macros is applicable only during regular typesetting runs, not during the manipulation and digestion of hyphenation patterns. For the latter, simple and direct macros must be designed such as the following:

   (a) For the special characters for which standard commands are available, such as `\o` for ø and `\ss` for ß, it is sufficient to prepare declarations of the following form:

```
\def\ss{^^ff}\lccode"FF="FF
\def\SS{^^df}\lccode"DF="FF
\def\ae{^^e6}\lccode"E6="E6
\def\AE{^^c6}\lccode"C6="E6
\def\oe{^^f7}\lccode"F7="F7
\def\OE{^^d7}\lccode"D7="F7
\def\o{^^f8} \lccode"F8="F8
\def\O{^^d8} \lccode"D8="F8
\def\i{^^19} \lccode"19="19
\def\j{^^1a} \lccode"1A="1A
\def\aa{^^e5}\lccode"E5="E5
\def\AA{^^c5}\lccode"C5="E5
\def\l{^^aa} \lccode"AA="AA
\def\L{^^8a} \lccode"8A="AA
```

   (b) For the other characters — those that carry a diacritical mark — it is better to resort to intermediate macros, some of which map the accent macro and character to a single control word, and some for defining the meaning of such control words. The whole trick is accomplished as follows: first you define the control word mappings

```
\def\'#1{\csname @ac@#1\endcsname}
\def\`#1{\csname @gr@#1\endcsname}
...
```

   and so on, with the prefixes that are listed in Table 2 for the other diacritical marks. Then, for each of the approximately one hundred diacriticized characters, you must set up declarations such as these:

```
\catcode`\^^a0=11 % letter  \u{a}
\lccode"A0="A0    % lc code
\def\@u@a{^^a0}   % ctrl word
%
\catcode`\^^80=11 % letter  \u{A}
\lccode"80="A0    % lc code \u{a}
\def\@u@A{^^80}   % ctrl word
%
...
```

   and so on for the remaining special characters whose codes can be deduced from Table 1.

   (c) Most important, steps 1 and 2 must be confined within a group together with the pattern file input command, so as to keep such declarations local to the sole group where patterns are being manipulated and digested.

3. After closing the group mentioned in the above item, it is wise to declare the default language, the default encoding (so that you do not need to declare it in every preamble of every document), and the relevant parameters for the leftmost and rightmost word fragments:

```
\language=0
\lefhyphenmin=2
\righthyphenmin=3
\def\encodingdefault{T1}
```

   This done, you are ready to run the initializer.

3.3 Extended fonts and LaTeX2.09

If you are still using LaTeX2.09 or you want to have a LaTeX2.09-compatible version, do the following:

1. copy the file `lplain.tex` to a new file, say, `emlplain.tex`;
2. edit `emlplain.tex` by replacing the line `\input lfonts` with `\input emlfonts`
3. copy `lfonts.tex` into `emlfonts.tex`;
4. edit `emlfonts.tex` by adding a line that loads a dc or an em font for every text font already loaded; as shown here (original lines are marked with `<--`):

```
\font\fivrm =cmr5  % roman   <--
\font\efivrm=dcr5  % ex. roman
...
\font\elvrm =cmr10\@halfmag % roman <--
\font\eelvrm=dcr10\@halfmag % ex. roman
...
\font\frtnrm =cmr10\@magscale2 % rom <--
\font\efrtnrm=dcr10\@magscale2 % ex. rom
...
```

5. modify the definitions of the font changing commands for all point sizes; e.g. for the ten-point size, search for the definition `\def\xpt` and change:

   `\def\prm{\fam\z@\tenrm}%`

   to

   `\def\prm{\fam\z@\etenrm}%`

   doing the same for all the other font selections.

When you edit `emlfonts.tex`, near the end of the file, you should comment out the definition of `\$`, because with dc or em fonts the dollar sign always has the correct shape.

The procedure seems very complicated but really it amounts to just some time spent in careful repetitive editing: duplicating lines, replacing `cm` with `dc` and adding an `e` in the proper places.

Another point must be kept in mind: the program `tex` has a finite memory for holding font information. If you have made all the modifications explained above, you end up with a LaTeX format where 106 fonts have been preloaded. This might be too much for your "small TeX", but presents no problems to the "big TeX" implementations.

You also need to edit your newly created files `emplain.tex` and/or `emlplain.tex` and replace

```
\input hyphen
```

with

```
\input lhyphen
```

unless your `lplain.tex` file is dated 31 March 1992 or later, in which case it may already have been done.

4 Hyphenation patterns

The best way to have the proper hyphenation patterns for the languages one chooses to use is to retrieve them from the CTAN archives. If you have access to the Internet, just `ftp` to the nearest CTAN archive and explore the directory `/tex-archive/languages`. Select the directory corresponding to the chosen language, and hopefully the proper set of hyphenation patterns is already there.

If you do not have access to the Internet, you probably know somebody who does. If you don't, ask the TUG office for a diskette containing what you can't otherwise obtain, and pay what the TUG office will charge you. If the whole procedure seems too complicated, just consult the advertising pages of any issue of *TUGboat* and choose the vendor who can provide you with what you desire; the vendor prices are higher but generally you get a full set of diskettes with an `install` program that saves you all the burden of retrieving, initializing, etc.

But what happens if you decide to use a language for which no hyphenation patterns have been prepared? This is most unlikely. If you read the paper by Sojka and Ševeček [6], you will find a table where 38 hyphenation pattern files are examined; they deal with 32 different languages that include both flavors of English (US and UK), modern and classical Latin and Greek, Esperanto, and most modern European and North and South American languages. Besides Greek, patterns exist also for languages that do not use the Latin alphabet (with or without diacritics), such as Russian, and possibly for less known regional languages.

But what if you are unlucky — the patterns you are looking for do not exist, or they exist but are unreachable, or they are too large for the capabilities of your `tex` program ... In this case you yourself must create a file containing your patterns; you must carefully read Appendix H of *The TeXbook* and, of course, you must know the "strange" language you want to use, or at least have a perfect knowledge of its grammar and hyphenation rules. It's not too difficult, with or without the use of the program `patgen`, but this topic is sort of self-contained, so I leave it for another tutorial [1].

So, from now on, we assume that you have all the pattern files you need. You should at this point edit (or create) the file `lhyphen.tex` to read:

```
% File lhyphen.tex created on ...
% by ...
% It loads the hyphenation patterns
% for the following languages:
% 0) US english
% 1) italian
% 2) french
% 3) ...
%
\input chardefs
%
\language=0 \chardef\l@english 0
\input hyphen
%
\newlanguage\l@italian
\language\l@italian
\input ithyph
%
\newlanguage\l@french
\language\l@french
\input frhyph
%
% and so on
%
% Default values
\language\l@english
```

```
\lccode‘’= 0
\lefthyphenmin=2
\righthyphenmin=3
%
\endinput
```

If you are following the "poor man" procedure, the `chardefs.tex` file to be input might read simply like this:

```
\lccode‘’=‘’
\catcode’33=11 %\oe
\lccode’33=’33 %\oe
\let\oe=^^1b
...
```

and similar definitions and code assignments for all the other special ligatures or diacriticized characters present in the cm fonts that are necessary in the languages you are going to use.

The apostrophe should receive an `\lccode` different from 0 because it should not interrupt TEX words of the type *quest'anello*, *l'approbation*, *s'organizzen* in languages such as Italian, French, or Catalan (where vocalic elision is marked with an apostrophe) — remember the definition of a TEX word: "... characters..., with a non-zero `\lccode`,...". This is one of the most frequent requests for help I receive from Italian users, who forget to `\lccode` the apostrophe and complain about (LA)TEX not hyphenating after such a sign.

If you are going to use extended fonts (dc or em) you need a `chardefs.tex` file that defines a set of macros for changing such sequences as `\u{a}` (ă) into the numerical code of the diacriticized letter (in this case `’240` or `"A0` or `160`), while assigning a non-zero `\lccode` to the character in question. Such a file, in other words, should contain things similar to those that have been described for LATEX 2_ε.

Such macros are simple but numerous when you want to have a complete map to all 102 diacriticized letters of the Cork encoding and to the 13 special characters ß, œ, æ, ø, å, ł, ı, and ȷ, with their uppercase possible counterparts (plus the apostrophe):

```
% Save current @ catcode and ...
\chardef\atcatcode=\the\catcode‘\@
% ... make it a letter.
\catcode‘\@=11
\def\ss{^^ff}\lccode"FF="FF\uccode"FF="DF
\def\SS{^^df}\lccode"DF="FF\uccode"DF="DF
\def\ae{^^e6}\lccode"E6="E6\uccode"E6="C6
\def\AE{^^c6}\lccode"C6="E6\uccode"C6="C6
\def\oe{^^f7}\lccode"F7="F7\uccode"F7="D7
\def\OE{^^d7}\lccode"D7="F7\uccode"D7="D7
\def\o{^^f8} \lccode"F8="F8\uccode"F8="D8
\def\O{^^d8} \lccode"D8="F8\uccode"D8="D8
\def\i{^^19} \lccode"19="19\uccode"19="49
\def\j{^^1a} \lccode"1A="1A\uccode"1A="4A
\def\aa{^^e5}\lccode"E5="E5\uccode"E5="C5
\def\AA{^^c5}\lccode"C5="E5\uccode"C5="C5
\def\l{^^aa} \lccode"AA="AA\uccode"AA="8A
\def\L{^^8a} \lccode"8A="AA\uccode"8A="8A
\lccode‘’=‘’
```

The hexadecimal codes appearing in the above definitions can be found in Table 1, where the octal and hexadecimal codes for all the other diacriticized characters can also be found. The uppercase codes are specified so that the case of such special letters can be properly changed.[3]

Besides the above macros `chardefs.tex` must also contain the character mappings necessary for the cases when accent macros are used; in the CTAN archives you can find a file, `compatible.tex`, that contains intermediate macros for these mappings. They look like this:

```
\def\’#1{{\expandafter
  \ifx\csname @ac@#1\endcsname\relax
    {\accent19 #1}%
  \else
    \csname @ac@#1\endcsname
  \fi}}
\def\‘#1{{\expandafter
  \ifx\csname @gr@#1\endcsname\relax
    {\accent18 #1}%
  \else
    \csname @gr@#1\endcsname
  \fi}}
...
\def\~#1{{\expandafter
  \ifx\csname @til@#1\endcsname\relax
    {\accent’176 #1}%
  \else
    \csname @til@#1\endcsname
  \fi}}
\def\"#1{{\expandafter
  \ifx\csname @um@#1\endcsname\relax
    {\accent’177 #1}%
  \else
    \csname @um@#1\endcsname
  \fi}}
```

The trick is this: any accent macro (for example, the one for the acute accent), operating, say,

[3] This is not necessary with LATEX 2_ε because definitions are local to the section where patterns are handled. On the other hand, remember to avoid groups while specifying these codes with LATEX2.09, otherwise you lose the possibility of treating words with special characters in the proper way.

on the letter 'a', maps to the protected[4] internal control word `\@ac@a`, if this word is defined; otherwise, it operates as a regular accent macro with cm fonts. If you are using the extended fonts and the set-up I am describing, you might as well simplify such macros to:

```
\def\'#1{\csname @ac@#1\endcsname}
\def\`#1{\csname @gr@#1\endcsname}
...
\def\~#1{\csname @til@#1\endcsname}
\def\"#1{\csname @um@#1\endcsname}
```

The whole set of prefixes of such control words is summarized in Table 2. Actually, there are no LaTeX macros for the ring accent and the ogonek diacritical mark;[5] if you need to use them, you can define `\r` and `\g` to map to the proper control words by means of the prefixes shown in Table 2.

Next you must assign every such control word to an extended character and then assign that character the proper `\lccode` and `\uccode`, an operation that is lengthy because of the number of characters, but at least is repetitive; all such assignments are of this type:

```
\catcode`\^^a0=11 % letter  \u{a}
\lccode"A0="A0    % lc code
\uccode"A0="80    % uc code \u{A}
\def\@u@a{^^a0}
%
\catcode`\^^80=11 % letter  \u{A}
\lccode"80="A0    % lc code \u{a}
\uccode"80="80    % uc code
\def\@u@A{^^80}
%
...
```

The CTAN archives contain a file called `extdef.tex` that has definitions similar to the ones above, but resorts to two macros:

`\csubinverse` and `\charsubdef`

that map em/dc-font extended characters to corresponding cm-font accent macros + characters, and vice versa. These macros should not be necessary if you consistently use only extended fonts.

On the other hand, if you have a keyboard with national characters (and you pay some attention by editing your files before you send them to colleagues who might not have the same keyboard) you could reduce your keying if you map the input codes directly to the corresponding extended TeX codes; for this you might add to your `chardefs.tex` file some definitions of the form:

```
\catcode`\à=13 \def à{\@gr@a}
\catcode`\á=13 \def á{\@ac@a}
...
\catcode`\ñ=13 \def ñ{\@til@n}
```

Remember to end the file by restoring the right `@` catcode:

```
\catcode`\@=\atcatcode
```

If you spend the time necessary to create this `chardefs.tex` file from scratch, to explore the CTAN archives, and to put together the various parts, you understand why vendors charge you a reasonable price for selling TeXware that everybody could otherwise get at no cost!

grave	`@gr@`	caron	`@v@`
acute	`@ac@`	breve	`@u@`
circumflex	`@hat@`	macron	`@eq@`
tilde	`@til@`	dotaccent	`@dot@`
dieresis	`@um@`	cedilla	`@c@`
hungarumlaut	`@H@`	ogonek	`@og@`
ring	`@r@`		

Table 2: Control-word prefixes for accent-macro mapping

5 Initialization

Now you have all the necessary pieces of information to create the formats `latex.fmt` with LaTeX 2_ε, `emplain.fmt` with extended font plain TeX, and `emlplain.fmt` for extended font LaTeX2.09.

It's time to run `initex`, the TeX initialization program, the only one that can chew and digest the hyphenation patterns and store them into memory in a special way so that during a regular TeX run the hyphenation process proceeds efficiently.

Depending on your software implementation, this initialization program may be a different `.exe` (i.e. there are both `tex.exe` and `initex.exe`), or the initialization may be an option to the regular procedure, or the executable module may behave differently depending on the name used to invoke it. Again, depending on the operating system, you might also include the arguments to the command within double quotes. So this simple initialization task may prove to be more difficult than it should.

On my DOS personal computer, with the various implementations I have used, I had to follow three different approaches; however, command files come in handy for doing the whole job without bothering too much about the details. You have to:

[4] 'Protected' in the sense that it contains the character `@`, which is not a letter during normal (LA)TeX operation, so that it is impossible to inadvertently redefine it.

[5] Actually the ring accent is used only on the letters 'a' and 'u', so that the standard LaTeX macros `\aa` and `\AA` are sufficient for the former, but you still need something for using the latter.

1. set up the environment variables according to the documentation for your particular implementation of TEX;
2. set up the search paths, if necessary;
3. invoke the initialization program with the proper syntax and with the proper arguments;
4. `\dump` the format;
5. move the format to the directory where TEX expects to find format files;
6. possibly preload the format so as to create a specific executable program.

Steps 1 and 2 are very much system- and software-dependent, so please check your documentation very carefully. Below I show how I would do the job for LATEX2.09, by means of a C-source derived `tex` program named `ctex` when it operates as a regular `tex` program, and `cinitex` when it operates as an initializer program.[6]

1. I set up the following environment variables (but remember that the names of my directories do not necessarily match yours):

   ```
   SET TEXFORMATS=C:\TEX\ctexfmts
   SET TEXPOOL=C:\TEX\ctexfmts
   SET TEXFONTS=C:\TEX\fonts
   SET TEXINPUTS=.;C:\TEX\inputs
   ```

   With other implementations it might be possible to set environment variables that control the amount of memory that the program assigns to the various operating parts.
2. I decide if the executable `cinitex.exe` is already in the proper directory; if not, I rename the executable `ctex.exe` to that name

```
if exist C:\TEX\cinitex.exe goto runinitex
rename C:\TEX\ctex.exe cinitex.exe
```

3. Then I produce the format and `\dump` it in one step:

   ```
   :runinitex
   C:\TEX\cinitex %1 \dump
   ```

   In the above command `%1` is the name of the format I want to produce; in our examples it might be `emplain` or `emlplain`.
4. Next I move the format file to the proper directory

   ```
   move %1.fmt C:\TEX\ctexfmts
   ```
5. Finally I reset the program name to `ctex.exe`

   ```
   rename C:\TEX\cinitex.exe ctex.exe
   ```
6. At this point, with some implementations of `tex`, it might be possible to run a preloading facility that creates an executable image of the program with the format preloaded; it speeds up the preliminary operations of a regular TEX run a little, but it is not a real necessity; many implementations do not have this facility.

That's all. For running (LA)TEX, another command file sets up the same environment variables (in case they were not set or had been reset), and invokes the executable with:[7]

```
ctex &emlplain %1
```

where, as usual, `%1` is replaced by the `.tex` file name you want to process.

During initialization you might experience a number of error messages; try to understand which is the cause and exit the initializer with `x` at the program prompt. Some possible causes include:

1. `initex` did not find some `.tex` file(s): check that all the necessary files are on a TEX search path conforming to the `TEXINPUTS` environment variable.
2. `initex` did not find some font files: check for spelling errors in the parts you have edited; check that all `.tfm` files are in the directory(ies) identified by the `TEXFONTS` environment variable.
3. `initex` complains about the `tex.pool` file:[8] the file is missing or the file in the directory pointed at by `TEXPOOL` does not belong to the particular TEX implementation you are using; retrieve the correct `tex.pool` file and be sure to put it in the correct directory.
4. `initex` complains about some undefined control sequence: check the name of the control sequence and correct your spelling in the files you have edited.
5. `initex` complains about duplicate patterns: press `<Enter>` and keep going, but if the problem shows up again, you'd better get out with `x`. In any case, check your hyphenation files and your `lhyphen.tex` file; you may have forgotten to issue the command `\newlanguage` or most probably you made a spelling mistake. If you made your own patterns the possibility of

[6] Other implementations may have `initex` as the initializer, `virtex` as a "virgin" version of TEX (that is, a program without any format preloaded), `tex` as a version with `plain.fmt` preloaded, and `latex` as a version with `lplain.fmt` preloaded. In other systems, particularly UNIX, all these names are aliases that call the same executable which behaves differently according to the name by which it has been called.

[7] Change `emlplain` to `emplain` if you want to use plain TEX instead of LATEX.

[8] On DOS platforms the name is `tex.poo`.

having inadvertently duplicated some patterns is normal, but must be corrected.

6. `initex` complains about memory limitations: if you are using a "big TEX" implementation based on a C-source, this should not happen. If it does, either you want to go into the Guinness book of records for the largest number of languages treated at the same time, or you have bad patterns, or your C-source program did not work properly. If your program comes from a Pascal source and you have the source code, after having properly checked that you are not going into the Guinness book, and that your patterns are OK, you could carefully modify the memory declarations in the source code and recompile and link the program. I hope you know what you are doing. If you do not have the source code, but you have an implementation (like `sb39tex`) that allows you to exercise some control over the memory usage, check the documentation and proceed accordingly.

With respect to memory limitations, specifically hyphenation memory limitation, TEX has two memory areas, the *trie* and the *ops* ones, the former being dedicated to holding the patterns (all the patterns of all the languages that have been loaded) and the latter to holding the "branch" information of the pattern structured lists. The greater the number of patterns, the greater the trie memory occupied; the more complex the pattern structure, the greater the ops memory occupied.

When you run `initex`, the `.log` (or `.lis`) file documents all the relevant information about the run; in particular, towards the end you will find a listing that looks like this:

```
14 hyphenation exceptions
Hyphenation trie of length 8172
         has 407 ops out of 750
  23 for language 6
  15 for language 5
  15 for language 4
  25 for language 3
  110 for language 2
  38 for language 1
  181 for language 0
```

from which you can obtain important information about the memory occupation dedicated to multilanguage issues:

1. the 14 hyphenation exceptions are those that come with the US-English `hyphen.tex` file, originated by Liang and Knuth and described in *The TEXbook*, Appendix H. The other languages I loaded do not have hyphenation exceptions because this is my policy when I prepare my patterns.

   There is nothing wrong with hyphenation exceptions; simply they are better suited for a non-flexive or moderately flexive language, such as English, than for flexive languages such as all the Romance ones. In English, for a noun you have two forms, for an adjective one form, for a verb four or five forms. In Italian nouns require two forms, adjectives up to four, verbs approximately 60 and this does not include all the possible agglutinations of enclitic pronouns. For French, Spanish, Portuguese, Catalan, Romanian, and Latin the situation is similar, so that if a hyphenation exception involves the stem of a verb, you may have to input some 60 entries (at least) in the `\hyphenation` argument!
2. US-English hyphenation patterns occupy 6075 trie memory words and 181 ops; the standard size of the trie memory (in a regular-sized TEX implementation) is 8000 words; French comes second in complexity, requiring from 1122 to 1433 trie memory words[9] and 110 ops; the other Romance languages require on the average 600 trie memory words and 25 ops. The ops numbers are simply additive while the trie memory words are not, because the cited numbers include some undocumented overhead, so that the final trie memory occupied is smaller than the sum of the single language trie memory occupied. But during the `initex` run you need the availability of at least the sum of the single language occupations, so that the program can massage the patterns and compress them in the proper way.

   When you have to control the memory occupation, you must keep these numbers in mind in order to understand possible complaints by `initex`.
3. The ops numbers give you a fairly good idea of the complexity of the hyphenation rules for a given language as they are translated into TEX code. In my experience, German patterns appear to be the most complicated, requiring 9980 trie memory words and 281 ops; this is the set of patterns indicated as DE_{max} in Sojka and Ševeček [6],[10] and if you have to use German

[9] Depending on the presence of extended characters; the numbers refer to my poor-man or extended-character patterns, respectively.

[10] However, they report 255 ops while my `cinitex` executed 281; perhaps the files we examined are not exactly the same.

you'd better have a "big TEX" implementation with a large trie memory size.

6 Language-dependent macros

The best thing to do is to retrieve the babel package; this is the finest set of multilanguage macros I know of and I recommend it to everybody. My polyglot macros are also an acceptable "poor man" alternative for those who have stuck to cm fonts and relied on intelligent accent macros. The macros should be divided into three categories:

1. language selection;
2. label selection;
3. style selection.

If you want to make your own macros, because you are going to use your multilanguage implementation in a restricted or in a special way, you can do the following.

6.1 Language selection

In the lhyphen.tex file all languages were identified by a literal name mapped to an internal number of the form \l@*language*; this allows you to set up simple macros such as these:[11]

```
\def\Lang#1{\expandafter
   \language\csname l@#1\endcsname
   \csname #1settings\endcsname}
%
\def\englishsetting{\lccode‘’=0
\righthyphenmin=3}
%
\def\italiansettings{\lccode‘’=‘’
\righthyphenmin=2}
%
\def\frenchsettings{\lccode‘’=‘’
\righthyphenmin=3}
%
% and so on
%
```

You change hyphenation rules simply by issuing commands of the type

```
{\Lang{french} ... % french text
}                  % end french
```

without forgetting the group braces so that the action is confined to the enclosed group alone.

In LATEX you can use the above definitions also as environments:

```
\begin{Lang}{french}
...                % french text
\end{Lang}
```

or you might prefer to define a new environment:

```
\newenvironment{French}{\Lang{french}}{}
```

so that you can type:

```
\begin{French}
...                % french text
\end{French}
```

and thereby following standard LATEX markup style.

6.2 Label selection

What has been described in the previous subsection is suitable for setting short passages of one language within a text composed in another language; for example, French extracts (properly hyphenated in French) in an English-language essay (with citations in an appropriate English style).

If you want to write a whole document in another language you should be able to change "Chapter" into "Chapitre", "Table" into "Tableau", and so on, with a single command. Fortunately, as of 31 March 1992, these words are no longer hardwired into the LATEX styles; they are contained in control sequences with self-explanatory names. So, together with the language settings, you need a \setcaptions macro of the following type:

```
\def\setcaptions#1{%
\csname #1captions\endcsname}
%
\def\frenchcaptions{%
\def\refname{R\’ef\’erences}
\def\abstractname{R\’esum\’e}
\def\bibname{Bibliographie}
\def\chaptername{Chapitre}
\def\appendixname{Annexe}
\def\contentsname{Table des mati\‘eres}
\def\listfigurename{Liste des figures}
\def\listtablename{Liste des tableaux}
\def\indexname{Index}
\def\figurename{Figure}
\def\tablename{Tableau}
\def\partname{Partie}
\def\enclname{P.~J.}
\def\ccname{Copie \‘a}
\def\headtoname{A}
\def\pagename{Page}
%
\def\today{\ifnum\day=1\relax
1\/$^{\rm er}$\else\number\day\fi
\space\ifcase\month\or
janvier\or f\’evrier\or mars\or
avril\or mai\or juin\or juillet\or
ao\^ut\or septembre\or octobre\or
```

[11] \lefthyphenmin is supposed to maintain the default value of 2; of course some settings might change this value, but if you choose to do so, you should insert an appropriate setting for every language.

```
novembre\or d\'ecembre\fi
\space\number\year}
}
%
% and so on for the other languages
%
```

With a LaTeX2.09 document you can start this way:

```
\documentstyle{book}
\Lang{french}\setcaptions{french}
\begin{document}
...
\end{document}
```

The language selection macros and the caption definition commands could also be incorporated into `hyphen.cfg` for LaTeX 2_ε, and into `lhyphen.tex` for LaTeX2.09, so that they remain available with any document type and you need not specify a particular option or package every time you start a new document.

6.3 Style selection

If you want your composition style to be correct, all the way down to respecting the typographic traditions of another country, then you should rely completely on the babel macros.

The most apparent differences consist in these points:

1. In France (and perhaps in some other countries) French spacing is normally used; French spacing leaves the same amount of white space after all punctuation marks, but leaves some thin space before the "tall" punctuation marks such as the colon and semicolon, the question and exclamation marks, the parentheses,[12] the quotation marks; such spaces cannot split at the end of the line. All these punctuation marks must be made active in text mode so that they introduce the right amount of white space, or, conversely gobble extra white space.
2. The quotation marks are the most variable in different countries; in the US high quotation marks with the shape of normal or reversed commas are used; in France and many other countries guillemets are used; in Italy we use both types. In some countries open quotation marks are identical to closed quotation marks but are lowered; in other countries they are reversed, that is left quotation marks are used for closing a quotation instead of opening it. And so it goes.

   All these marks are present in the extended fonts, although in the em fonts the guillemets are far from optimal; dc fonts and PostScript fonts have perfect shapes. But in any case suitable macros must be set up correctly in order to insert the proper quotation marks.
3. Ordinal numbers are another point of difference; in some countries a digit is immediately followed by the desinence or the final letter of the corresponding spelled-out ordinal; in other countries this literal ending is separated by a period; in yet other countries this ending is set as an exponent (sometimes underlined). Flexive languages may have different endings for masculine and feminine, singular and plural.

   Elsewhere, ordinals are expressed by roman numerals; the lowercase roman letters are customary in English, and this habit has gained wide acceptance in many places, while good style in many countries requires the use of uppercase roman letters only, possibly from a small caps font.

For these sorts of style problems, don't do it the do-it-yourself way; it's too difficult (unless you are a book connoisseur, of course). It's better to rely on the babel package, which, although it was created and is maintained by one man, J. Braams, benefits from continuing suggestions and constructive criticism by all members of the TeX community, with the result that the package is constantly being upgraded and always becoming better.

7 Conclusion

This long tutorial on multilanguage typesetting has tried to focus on some of the problems concerning setting texts in different languages by means of a single program, TeX or LaTeX. Configuring the program to handle several languages is not that difficult, but it does require patience, good knowledge of the software, a long week-end... But when you've set up the program the proper way, you will enjoy it much more than before.

References

[1] Beccari C., Oprea R., Tulei E., "How to make a foreign language pattern file: Romanian", *TUGboat*, 16(1):31–42, March 1995.

[2] Braams J., "babel, a multilingual style-option system for use with LaTeX standard document styles", *TUGboat*, 12(2):291–301, June 1991. Update: *TUGboat* 14(1):60–62, April 1993.

[3] Goossens M., Mittelbach F., Samarin A., *The LaTeX Companion*, Addison-Wesley, Reading, Mass., 1994.

[4] Knuth D.E., *The TeXbook*, Addison-Wesley, Reading, Mass., 1990.

[12] The thin white space goes *after* the open parenthesis.

[5] Lamport L., *LaTeX A Document Preparation System*, Addison-Wesley, Reading, Mass., 1986. Second edition, dealing with LaTeX 2$_\varepsilon$, 1994.

[6] Sojka P., Ševeček P., "Hyphenation in TeX — Quo Vadis?", *Proceedings of the Eighth European TeX Conference* (Sept. 26–30, 1994, Gdańsk, Poland), 59–68.

◇ Claudio Beccari
Dipartimento di Elettronica
Politecnico di Torino
Turin, Italy
Email: beccari@polito.it