Package `ecclesiastic.sty`*

Claudio Beccari Donald Goodman

v.0.1 2007/09/28

Abstract

This package extends the typesetting facilities of the *latin* option to the babel package for typesetting Latin according to the tradition of ecclesiastic documents; these documents are mainly the devotional books used by the Roman Catholic clergy, but not limited to them, that are being published not only by the Vatican Typography, but also by many Printing Companies around the world.

1 Introduction

This small extension package extends the features of `latin.ldf` by adding a certain level of "frenchization" to the way of typesetting Ecclesiastical Latin; in particular all punctuation marks, except comma and full stop are preceded by a small space. The guillemets are also accompanied by small spaces to the right of the opening marks and to the left of the closing ones, with the provision of removing spurious previous spaces. Footnotes are not indented and their reference number is not an exponent, although footnote marks in the text keep being exponents.

The acute accent (actually the apostrophe sign) is made active so as to set an acute accent over the following vowel (notice that in Latin there is no elision, so there cannot be any conflict between the acute accent and the elision apostrophe). Ecclesiastical Latin uses the æ and œ ligatures. Goodman asked to declare 'a' and 'o' as active characters so that the spelling `ae` and `oe` would automatically produce the equivalent of `\ae` and `\oe` respectively.

In practice Beccari found serious programming problems with this solution and adopted an alternative one; specifically the adopted solution was to type in `"ae` and `"oe` respectively, and æ and œ would be inserted in the source text without the need of leaving blancks after the control sequences or the need of inserting extra braces; therefore one types in `c"aelum` and this is equivalent to `c\ae␣lum` or `c{\ae}lum` or `c\ae{}lum`; the saving in the input stream is evident and misstypings are likely less frequent.

*This document corresponds to ecclesiastic.sty v.0.1, dated 2007/09/28.
Claudio Beccari (`claudio dot beccari at alice dot it`) did the programming. Donald Goodman (`dgoodmaniii at gmail dot com`) asked for this extension, produced the requirements, and tested the results.

The active apostrophe for the acute accent behaves properly also with y and `'ae` and `'oe` produce the accented diphthongs.

Of course, when using the OT1 encoding all accents interfere with hyphenation and kerning. When using the T1 encoding this interference takes place only with the accented diphthongs æ and œ; no visible problems for the lack of kerning, but no hyphenation takes place after the accented diphthong until the end of the word, even if the grammar allowed it.

2 Usage

Besides loading this package with the usual

```
\usepackage{ecclesiastic}
```

after loading babel with the option *latin* (possibly among other ones) all you have to do is to input your source code the usual way, except that for guillemets and accents you are supposed to use the `"` and the `'` active characters. The input code

```
Ita enim fit, ut regn'are is "< in m'entibus h'ominum "> dic'atur non
tam ob mentis 'aciem scienti'aeque su"ae amplit'udinem, quam quod ipse est
V'eritas, et verit'atem ab eo mort'ales haur'ire atque obedi'enter acc'ipere
nec'esse est;  "< in volunt'atibus "> item "< h'ominum ">, quia \dots
```

will produce the following text:

> Ita enim fit, ut regnáre is « in méntibus hóminum » dicátur non tam ob mentis áciem scientiǽque suæ amplitúdinem, quam quod ipse est Véritas, et veritátem ab eo mortáles hauríre atque obediénter accípere necésse est ; « in voluntátibus » item « hóminum », quia . . .

Notice that the source text has spaces around the guillemets, but the typeset code has the right small and constant space, irrespective of justification. Notice the use of the 'acute' accent (actually the apostrophe) for accented vowels and diphthongs. Notice the space in the typeset text before the semicolon.

3 Documented code

Some checks in order to use this package together with the one it should extend.

```
1 ⟨*package⟩
2 \def\CheckLatin{\expandafter\ifx\csname captionslatin\endcsname\relax
3                   \PackageWarning{ecclesiastic}{\MessageBreak
4                   latin must be specified as a global option\MessageBreak
5                   or it must be passed as an option to babel\MessageBreak
6                   \MessageBreak
7                   Nothing done}\expandafter\endinput\fi}
8
9 \@ifpackageloaded{babel}{\CheckLatin}{%
10            \PackageWarning{ecclesiastic}{\MessageBreak
```

```
11          Package babel must be loaded before this package\MessageBreak
12          \MessageBreak
13          Nothing will be done}\endinput}
```

The following code was borrowed from frenchle.sty by Bernard Gaule, but there are several modifications; in particular the Cyrillic guillemets are effectively chosen as possible candidates before resorting to the horrible patch made up with the 'much smaller' and 'much larger' scriptsize signs. The Latin Modern fonts are preferred if they are available. When the T1 encoding is in force the guillemets are taken from the current font. The first macro specifies a common interface for chosing where to get guillemets from.

```
14 \let\og\empty\let\fg\empty%
15   \def\FrenchGuillemetsFrom#1#2#3#4{%
16     \DeclareFontEncoding{#1}{}{}%
17     \DeclareFontSubstitution{#1}{#2}{m}{n}%
18     \DeclareTextCommand{\guillemotleft}{OT1}{%
19        {\fontencoding{#1}\fontfamily{#2}\selectfont\char#3}}%
20     \DeclareTextCommand{\guillemotright}{OT1}{%
21        {\fontencoding{#1}\fontfamily{#2}\selectfont\char#4}}}
```

Then come the macros for selecting various type of guillemets: the first macro `\CyrillicGuillemets` selects them from the Cyrillic fonts; the second macro `\PolishGuillemets` selects them from the Latin Modern fonts, who were assembled by the Polish TUG, from which the name; the `\LasyGuillemets` represent the poor man solution, which represents the last resort:

```
22   \def\CyrillicGuillemets{\FrenchGuillemetsFrom{OT2}{wncyr}{60}{62}}
23   \def\PolishGuillemets{\FrenchGuillemetsFrom{T1}{lmr}{19}{20}}
24   \def\LasyGuillemets{%
25     \DeclareTextCommand{\guillemotleft}{OT1}{\hbox{%
26        \fontencoding{U}\fontfamily{lasy}\selectfont(\kern-0.20em(}}%
27     \DeclareTextCommand{\guillemotright}{OT1}{\hbox{%
28        \fontencoding{U}\fontfamily{lasy}\selectfont)\kern-0.20em)}}}
```

Now the previous macros are executed depending on what is available on the particular computer the code is executed on. Thinking to end up with a PDF file, we prefer the T1 encodec latin Modern fonts; if these are not installed we resort to the Cyrillic one, since the OT2 encoded Cyrillic fonts from the University o Wisconsin are generally installed by default also in pfb format; should htese be missing, then the poor man solution of the LaTeX simbols is chosen. At the same time, since the latter are the last resort, they are chosen as the default solution, although its difficult to find an installation where either the Latin Modern or the Wisconsin University fonts are not installed.

```
29   \IfFileExists{t1lmr.fd}{\PolishGuillemets}{%
30       \IfFileExists{ot2wncyr.fd}{\CyrillicGuillemets}{\LasyGuillemets}}
31   \DeclareTextSymbolDefault{\guillemotleft}{OT1}
32   \DeclareTextSymbolDefault{\guillemotright}{OT1}
```

Having defined the symbols, we now think to the spacing; we chose a smaller space than in French typography, but, essencially this glue is without stretch and shrink

components, so that this space remains constant and does not stretch or shrink for helping in line justification.

```
\def\guill@spacing{\penalty\@M\hskip.3\fontdimen2\font
                        \@plus\z@\@minus\z@}
```

Now we are in the position to define the opening and the closing guillemet commands.

The spacings on the interior of the guillements and the spacings before the "high" pucntuation marks are smaller than with the `frenchle.sty` settings for the French typography. This has been made following Robert Bringhurst reccomendations of tight spacings, in particular before the punctuation marks and within the French quotes.

Since Beccari is not used to such spacings, forbidden in Italian typography, he finds the traditional French spacings very large, too large for his taste. Bringhurst reccomendations come in very handy to justify the chosen spacings. May be who is used to wider spacings finds them too tight. We think we found a compromise.

```
\DeclareRobustCommand*{\begin@guill}{\leavevmode
                              \guillemotleft\penalty\@M\guill@spacing
                              \ignorespaces}
\DeclareRobustCommand*{\end@guill}{\ifdim\lastskip>\z@\unskip\fi
                     \penalty\@M\guill@spacing\guillemotright{}}
```

We add the definition of `\og` (ouvrir guillemets) and `\fg` (fermer guillements) to the `\extraslatin` list, as well as we add their 'emptiness' to the `\noextrnolatin` one.

```
\addto\extraslatin{%
      \renewcommand{\og}{\begin@guill}\renewcommand{\fg}{\end@guill}%
}
\addto\noextraslatin{\let\og\empty\let\fg\empty}
```

Therefore open guillemets may be input with the `\og` macro and the closed ones with the `\fg` macro. This might be inconvenient, so that the `"<` and `">` shortcuts should be preferred; these shortcuts assure that the spaces after these shortcuts are really spaces and are not used to terminate the macro name. B. Gaulle uses the `\xspace` macro from the xspace package, but if this package is not loaded or is not available, the `\xspace` macro behaves as `\relax` and does not produce what is intended to do. See below the extended defintion of the `"` shortcut active character.

Here we make the apostrophe an active char and define the shortcuts for Latin that introduce the acute accent over the specified vowels, lower and upper case. Probably upper case is useless, but it does not harm.

```
\initiate@active@char{'}%
\addto\extraslatin{\bbl@activate{'}}%
\addto\noextraslatin{\bbl@deactivate{'}}%

\declare@shorthand{latin}{'a}{\@ifnextchar e{\'\ae\@gobble}{\'a}}
\declare@shorthand{latin}{'e}{\'e}
\declare@shorthand{latin}{'i}{\'i}
\declare@shorthand{latin}{'o}{\@ifnextchar e{\'\oe\@gobble}{\'o}}
```

```
\declare@shorthand{latin}{'u}{\'u}
\declare@shorthand{latin}{'y}{\'y}
\declare@shorthand{latin}{'A}{\@ifnextchar E{\'\AE\@gobble}{\'A}}
\declare@shorthand{latin}{'E}{\'E}
\declare@shorthand{latin}{'I}{\'I}
\declare@shorthand{latin}{'O}{\@ifnextchar E{\'\OE\@gobble}{\'O}}
\declare@shorthand{latin}{'U}{\'U}
\declare@shorthand{latin}{'Y}{\'Y}
```

Here we redeclare the definition of the " shortcut active character; it is borrowed from italian.ldf, but a new \LT@cwm macro is added to the existing \lt@@cwm one so as to cope also with "ae and "oe, besides the guillemet commands.

The following declaration is probably a repetition of what is already in latin.ldf

```
\declare@shorthand{latin}{"}{%
\ifmmode
    \def\lt@next{''}%
\else
    \def\lt@next{\futurelet\lt@temp\lt@cwm}%
\fi
\lt@next
}%
```

This also should already be in latin.ldf; it is the command that inserts a discretionary break, but does not inhibit hyphenation in the rest of the word.

```
\def\lt@@cwm{\nobreak\discretionary{-}{}{}\nobreak\hskip\z@skip}%
```

This, for what concerns Latin, is new as an interface with the definitions of the guillemets

```
\def\lt@@ocap#1{\begin@guill}\def\lt@@ccap#1{\end@guill}%
```

This is completely new; il deals with \ae and \oe; since \ae is much more frequent than \oe, we start with testing for an 'a' followed by an 'e', otherwise we test about the presence of an 'o':

```
\DeclareRobustCommand\LT@cwm[2]{%
    \ifx#1a\bbl@afterelse
       \maybeae#1#2%
    \else\bbl@afterfi
        \testoe#1#2%
    \fi}
```

If a sequence ae was detected, then \ae is inserted in the input stream in place of that sequence, otherwise the two tokens are inserted in the input stream preceded by the discretionary break implied by the presence of the " sign that thriggered the whole process.

```
\def\maybeae#1#2{%
    \ifx#2e\bbl@afterelse
       \ae%
    \else\bbl@afterfi
        \lt@@cwm#1#2%
    \fi
}
```

The same procedure is valid for the sequence `oe`

```
\def\maybeoe#1#2{%
    \ifx#2e\bbl@afterelse
       \oe%
    \else\bbl@afterfi
       \lt@@cwm#1#2%
    \fi
}
```

But the presence of an 'o' must be checked before activating the previous macro:

```
\def\testoe#1#2{%
   \ifx#1o\bbl@afterelse
       \maybeoe#1#2%
   \else\bbl@afterfi
       \lt@@cwm#1#2%
   \fi}
```

This is the real execution of the `"` shortcut; remember that `\lt@csw` is an alias for `\lt@next`, the action associated with `"` when outside the math mode; furthermore `\lt@temp` contains the token following the `"` sign. Notice that the category code of the `\lt@temp` is compared to that of a generic letter; the choice of 'e' is absolutely irrelevant, because it is a generic letter; any other letter would have done the same. So, first the temporary token is compared to a letter; if it's a letter the `\LT@cwm` is executed; the latter on turn looks for an 'a' or an 'o' and possibly inserts a diphthong or a discretionary break; otherwise the temporary token is compared to `|`, so that the shortcut `"|` is possibly executed by inserting a discretionary break and by gobbling the bar; otherwise it checks for a 'less than' sign and possibly inserts double open guillemets; otherwise it checks for a 'greater than' sign and possibly inserts double closed guillemets; otherwise it checks for the slash and possibly it inserts a breakable slash `\slash`; otherwise it checks for another double straight quotes sign and possibly it inserts double open high quotes (this is useful for those keyboards that do not have the 'back tick' sign `).

```
\DeclareRobustCommand*{\lt@cwm}{\let\lt@@next\relax
\ifcat\noexpand\lt@temp e%
    \def\lt@@next{\LT@cwm}%
\else
    \if\noexpand\lt@temp \string|%
        \def\lt@@next{\lt@@cwm\@gobble}%
    \else
        \if\noexpand\lt@temp \string<%
            \def\lt@@next{\lt@@ocap}%
        \else
            \if\noexpand\lt@temp \string>%
                \def\lt@@next{\lt@@ccap}%
            \else
                \if\noexpand\lt@temp\string/%
                    \def\lt@@next{\slash\@gobble}%
                \else
```

```
                    \ifx\lt@temp"%
                        \def\lt@@next{``\@gobble}%
                    \fi
                \fi
            \fi
        \fi
    \fi
\fi
\lt@@next}%
```

This done let's take care of the punctuation. First we create the aliases of the puntuation marks with their original category codes

```
\edef\puntoevirgola{\string;}\edef\cc@pv{\the\catcode`;}%
\edef\duepunti{\string:}\edef\cc@dp{\the\catcode`:}%
\edef\puntoesclamativo{\string!}\edef\cc@pe{\the\catcode`!}%
\edef\puntointerrogativo{\string?}\edef\cc@pi{\the\catcode`?}%
```

Then we make those puctuation marks active and add their activeness to `\extraslatin`, and also their "deactiveness" to the `\noextraslatin` list. In this way we are sure that there is no interference with other languages.

```
\initiate@active@char{;}
\initiate@active@char{:}
\initiate@active@char{!}
\initiate@active@char{?}
\addto\extraslatin{\bbl@activate{;}}
\addto\extraslatin{\bbl@activate{:}}
\addto\extraslatin{\bbl@activate{!}}
\addto\extraslatin{\bbl@activate{?}}
\addto\noextraslatin{\bbl@deactivate{;}}
\addto\noextraslatin{\bbl@deactivate{:}}
\addto\noextraslatin{\bbl@deactivate{!}}
\addto\noextraslatin{\bbl@deactivate{?}}
```

Here we define the space before punctuation; again the glue that is inserted in the French typography is too large according to our taste; the glue we want to put in front of the high punctuation marks should be smaller and we chose a smaller compromise value, but again we fix the stretch and shrink components to zero.

```
\def\punct@spacing{\penalty\@M\hskip.4\fontdimen2\font
                   \@plus\z@\@minus\z@}
```

When then we give a definition to these actie characters; in each definition we start by eliminating aby previous spacing inserted by the typist, then we insert our space and finally the punctuation mark.

```
\declare@shorthand{latin}{;}{\ifdim\lastskip>\z@\unskip\fi
      \punct@spacing\puntoevirgola}
\declare@shorthand{latin}{:}{\ifdim\lastskip>\z@\unskip\fi
      \punct@spacing\duepunti}
\declare@shorthand{latin}{!}{\ifdim\lastskip>\z@\unskip\fi
      \punct@spacing\puntoesclamativo}
\declare@shorthand{latin}{?}{\ifdim\lastskip>\z@\unskip\fi
      \punct@spacing\puntointerrogativo}
```

For footnotes we require that the footnote mark be tiped flush to the left margin and that it is typed normalsize; this rquires the redefinition of the `\@makefntext` macro that must call a different version of `\@makefnmark`.

```
%
\let\lt@ori@makefntext\@makefntext
\newcommand\lt@makefntext[1]{%
    \parindent 1em%
    \noindent
    \lt@Makefnmark\enspace #1}
\newcommand\lt@Makefnmark{\hbox{\normalfont\@thefnmark.}}
```

We add these commands to the `\extraslatin` and `\noextraslatin` lists.

```
\addto\extraslatin{\let\@makefntext\lt@makefntext}
\addto\noextraslatin{\let\@makefntext\lt@ori@makefntext}
```

Is this correct? May be not! In a mixed language text footnotes get labelled in a different way depending on which language was in force when the `\footnote` command was issued. Any solution?

In order to leave the category codes clean we re-establish the default codes reassigning the active cars their initial meaning; we do this by executing . If Latin is the default language, or when Latin is selected, the macro is automatically executed and active catcodes reassigned to the active characters.

```
\noextraslatin
⟨/package⟩
```