Package `ecclesiastic.sty`*

Claudio Beccari Donald Goodman

v.0.3 2015/08/20

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. This package works only with pdfLaTeX; if an attempt is made to use it while typesetting with Xe|Lua LaTeX, its input is aborted and nothing is done. Most additions and modifications that can be achieved with this package under pdfLaTeX typesetting, are obtained thanks to clever use of the OpenType fonts when using Xe|Lua LaTeX.

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 guillemots 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

*This document corresponds to ecclesiastic.sty v.0.3, dated 2015/08/20.
Claudio Beccari (`claudio dot beccari at gmail dot com`) did the programming. Donald Goodman (`dgoodmaniii at gmail dot com`) asked for this extension, produced the requirements, and tested the results.

the need of leaving blanks 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 typos are likely less frequent.

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 diphthongs until the end of the word, even if the grammar allowed it; you can correct by hand this behavior if you add an ASCII straight double quote *after* the accented diphthong: (*a*) if the grammar allows a line break at that point, and (*b*) if justification of the right margin requires it.

2 Usage

Previous versions of this package were loaded after package babel with the `latin` language option; you can still do this way because for backwards compatibility this procedure is maintained, but we discourage it for the reasons that are explained hereafter.

With this new version the preferred procedure is to specify the `ecclesiastic` modifier to the `latin` language option. This very file contains le line:

```
\usepackage[latin.ecclesiastic,english]{babel}
```

Of course this new way of loading the package assures it to be loaded at the proper moment, but it requires a babel of at least version 3.9k; the old procedure is still usable with previous versions of the babel package.

Important notice: This package works only while using the `pdfLaTeX` typesetting program and babel; if you try to use it with `XeLaTeX` or `LuaLaTeX` its loading procedure issues an error message and the it aborts. if for any reason you have to use `XeLaTeX` or `LuaLaTeX` you can safely run them without loading thins package, but simply specifying the "key=value" option `variant=ecclesiastic` in a similar way as you specify the `ecclesiastic` modifier when using babel. With `XeLaTeX` you have the full functionality of the ecclesiastic Latin variant, while, at the moment, with `LuaLaTeX` the functionality misses the particular spacing around some punctuation marks and around guillemots. This functionality is under study, but it is hopefully available with the next package version.

After in one way or another you have assured the presence of the special macros that implement the ecclesiastic Latin style, all you have to do (with babel) is to input your source code the usual way, except that for guillemots 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 guillemots, 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; this space is gobbled and substituted with the proper space.

3 Documented code

Some checks in order to use this package together with the one it should extend. Firse we load the `iftex` package in order to have the right switch to test if we are typesetting with the `pdftex` engine; inf not, an error message is issued and loading aborted.

```
\RequirePackage{iftex}
\unless\ifPDFTeX
\PackageError{ecclesiastic}{\MessageBreak
*********************************************\MessageBreak
* This package works only with pdfLaTeX       \MessageBreak
* Please do not load it when typesetting with \MessageBreak
* XeLaTeX or LuaLaTeX.                        \MessageBreak
*********************************************\MessageBreak
}{%
*********************************************************\MessageBreak
* Carefully read the documentation of package            \MessageBreak
* ecclesiastic, and understand why most functionalities  \MessageBreak
* of this package are obtained with completely different \MessageBreak
* means, thanks the use of OpenType fonts.               \MessageBreak
*                                                        \MessageBreak
* Input of this package is aborted.                      \MessageBreak
*********************************************************\MessageBreak
}
\expandafter\endinput
\fi
```

Then we test if babel has already been loaded; if not an error message is issued and loading aborted. Loading is aborted also if babel has been loaded but the `latin` option has not been specified. Notice that with version 3.5 of the Latin description file, it is recommended to use the `ecclesiastic` modifier to the `latin` option; or it is possible to specify the `ecclesiastic` attribute after loading babel with the `latin` option, or even after specifying this language option to the `\documentclass` statement, where the modifiers can't be used. By so doing you

are sure that the next tests will never produce unexpected results. Nevertheless we recommend to use the first method, that is to specify the `ecclesiastic` modifier to the `latin` option.

```
\def\CheckLatin{\unless\ifcsname captionslatin\endcsname
                \PackageWarning{ecclesiastic}{\MessageBreak
                latin must be specified as a global option\MessageBreak
                or it must be passed as an option to babel\MessageBreak
                \MessageBreak
                Nothing done}\expandafter\endinput\fi}

\@ifpackageloaded{babel}{\CheckLatin}{%
           \PackageError{ecclesiastic}{\MessageBreak
           **********************************************\MessageBreak
           Package babel must be loaded before this package\MessageBreak
           **********************************************%
           }{Package loading is aborted}\endinput}
```

In practice you can chose among one of these methods to use this package.

1. First and recommended choice:

   ```
   \documentclass[...]{...}
   ...
   \usepackage[...,latin.ecclesiastic,...]{babel}
   ```

2. Second choice:

   ```
   \documentclass[...]{...}
   ...
   \usepackage[...,latin,...]{babel}
   ...
   \usepackage{ecclesiastic}
   ```

3. Third choice:

   ```
   \documentclass[...,latin,...]{...]
   ...
   \usepackage[...]{babel}
   \languageattribute{latin}{ecclesiastic}
   ```

4. Fourth choice:

   ```
   \documentclass[...]{...}
   ...
   \usepackage[...,latin,...]{babe}
   \languageattribute{latin}{ecclesiastic}
   ```

The following code was borrowed from frenchle.sty by Bernard Gaule, but there are several modifications; in particular we avoid the horrible patch made up with

the shorter and wider script size math signs. The Latin Modern fonts are preferred if they are available. When the T1 encoding is in force the guillemots are taken from the current font; if you want to typeset ecclesiastic Latin, you'd better use the T1 font encoding; if you don't, you may get proper accented glyphs, but no hyphenation after any accented glyph until the end of the word. No attempt is made to create fake guillemots or to pick them from fonts that do not match the current text font. In any case we provide the command `\FrenchGuillemetsFrom` (and its alias `\FrenchGuillemotsFrom`) that allows the user to chose what font s/he likes best.

The first macro specifies a common interface for choosing where to get guillemots from.

```
\let\og\empty\let\fg\empty%

  \def\FrenchGuillemetsFrom#1#2#3#4{%
    \DeclareFontEncoding{#1}{}{}%
    \DeclareFontSubstitution{#1}{#2}{m}{n}%
    \DeclareTextCommand{\guillemotleft}{T1}{%
       {\fontencoding{#1}\fontfamily{#2}\selectfont\char#3}}%
    \DeclareTextCommand{\guillemotright}{T1}{%
       {\fontencoding{#1}\fontfamily{#2}\selectfont\char#4}}}

  \let\FrenchGuillemotsFrom\FrenchGuillemetsFrom

```

Then come the macros for selecting various type of guillemots: the `\ToneGuillemets` macro selects them from the T1 encoded fonts; according to the user input encoding, in particular the `utf8` one, when this package is loaded the macros `\guillemotleft` and `\guillemotright` should be defined; if either one is not, then the default guillemots are taken from the T1 encoded Latin Modern fonts:

```
\AtBeginDocument{%
\unless\ifcsname guillemotleft\endcsname
  \def\ToneGuillemets{\FrenchGuillemetsFrom{T1}{lmr}{19}{20}}\fi}
```

Having defined the symbols, we now provide to the spacing; we chose a smaller space than in French typography, but, essentially 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@}
```

Notice that `\fontdimen2\font` gets the normal inter word space of the current font; therefore the defined spacing of the guillemots is about 1 pt, with a 10 pt sized font.

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

The spacings on the interior of the guillemots and the spacings before the "high" punctuation marks are smaller than with the `frenchle.sty` settings for the French typography. This has been made according to Robert Bringhurst's

recommendation to maintain tight spacing, 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 recommendations 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 guillemets*) to the `\extraslatin` list, as well as we add their 'emptiness' to the `\noextraslatin` one.

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

Therefore open guillemots 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 definition 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 redefine 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 guillemot commands.

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

```
\declare@shorthand{latin}{"}{%
\textormath{\def\lt@next{\futurelet\lt@temp\lt@cwm}}%
           {\def\lt@next{''}}\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 guillemots

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

This is completely new; it 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 triggered 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@cwm` is activated by `\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 do the same. So, first the temporary token category code is compared to that of a letter; if it's a letter then `\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 guillemots; otherwise it checks for a 'greater than' sign and possibly inserts double closed guillemots; otherwise it checks for the slash and possibly it inserts a breakable slash `\slash`; otherwise it checks for another double straight quote 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 punctuation 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 punctuation 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.3\fontdimen2\font
                   \@plus\z@\@minus\z@}
```

Then we give a definition to these active characters; in each definition we start by eliminating any 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 typed flush to the left margin and that it is typed in normal size; this requires 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.

```
166 \addto\extraslatin{\let\@makefntext\lt@makefntext}
167 \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 `\noextraslatin`. If Latin is the default language, or when Latin is selected, the `\extraslatin` macro is automatically executed and the original category codes reassigned to the active characters.

```
168 \noextraslatin
```