1 The Italian language

The file `italian.dtx`[1] defines all the language-specific macros for the Italian language.

The features of this language definition file are the following:

1. The Italian hyphenation is invoked, provided that file `ithyph.tex` was loaded when the LaTeX 2_ε format was built; in case it was not, read the information coming with your distribution of the TeX software, and the babel documentation.

2. The language dependent fixed words to be inserted by such commands as `\chapter`, `\caption`, `\tableofcontents`, etc. are redefined in accordance with the Italian typographical practice.

3. Since Italian can be easily hyphenated and Italian practice allows to break a word before the last two letters, hyphenation parameters have been set accordingly, but a very high demerit value has been set in order to avoid word breaks in the penultimate line of a paragraph. Specifically the `\clubpenalty`, and the `\widowpenalty` are set to rather high values and `\finalhyphendemerits` is set to such a high value that hyphenation is prohibited between the last two lines of a paragraph. In orer to make it consistent, also `\@clubpenalty` is set to the same value; actualy the latter value is the reset value after every sectioning command, so that after the first section, `\clubpenalty` is reset to the low default value. Thanks to Enrico Gregorio for spotting this serious bug.

4. Some language specific shortcuts have been defined so as to allow etymological hyphenation, specifically `"` inserts a break point in any word boundary that the typesetter chooses, provided it is not followed by and accented letter (very unlikely in Italian, where compulsory accents fall only on the last and ending vowel of a word, but may take place with compound words that include foreign roots), and `"|` when the desired break point falls before an accented letter.

5. The shortcut `""` introduces the raised (English) opening double quotes; this shortcut proves its usefulness when one reminds that the Italian keyboard misses the backtick key, and the backtick on a Windows based platform may be obtained only by pressing the `Alt` key while inputting the numerical code 0096; very, very annoying!

6. The shortcuts `"<` and `">` insert the French guillemots, sometimes used in Italian typography; with the T1 font encoding the ligatures `<<` and `>>` should insert such signs directly, but not all the virtual fonts that claim to follow the T1 font encoding actually contain the guillemots; with the OT1 encoding

[1] The file described in this section has version number v1.2t and was last revised on 2008/03/14. The original author is Maurizio Codogno, (`mau@beatles.cselt.stet.it`). It has been largely revised by Johannes Braams and Claudio Beccari

the guillemots are not available and must be faked in some way. By using the `"<` and `">` shortcuts (even with the T1 encoding) the necessary tests are performed and in case the suitable glyphs are taken from other fonts normally available with any good, modern LaTeX distribution.

7. Three new specific commands `\unit`, `\ped`, and `\ap` are introduced so as to enable the correct composition of technical mathematics according to the ISO 31/XI recommendations. `\unit` does not get redefined if the babel package is loaded *after* the package `units.sty` whose homonymous command plays a different role and follows a different syntax.

For this language a limited number of shortcuts has been defined, table 1, some of which are used to overcome certain limitations of the Italian keyboard; in section 1.3 there are other comments and hints in order to overcome some other keyboard limitations.

`"`	inserts a compound word mark where hyphenation is legal; it allows etymological hyphenation which is recommended for technical terms, chemical names and the like; it does not work if the next character is represented with a control sequence or is an accented character.
`"|`	the same as the above without the limitation on characters represented with control sequences or accented ones.
`""`	inserts open quotes “.
`"<`	inserts open guillemots.
`">`	inserts closed guillemots.
`"/`	equivalent to `\slash`

Table 1: Shortcuts for the Italian language

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the `@` sign, etc.

```
1 ⟨*code⟩
2 \LdfInit{italian}{captionsitalian}%
```

When this file is read as an option, i.e. by the `\usepackage` command, `italian` will be an ‘unknown’ language in which case we have to make it known. So we check for the existence of `\l@italian` to see whether we have to do something here.

```
3 \ifx\l@italian\@undefined
4     \@nopatterns{Italian}%
5     \adddialect\l@italian0\fi
```

The next step consists of defining commands to switch to (and from) the Italian language.

\captionsitalian The macro \captionsitalian defines all strings used in the four standard document classes provided with LaTeX.

```
\addto\captionsitalian{%
  \def\prefacename{Prefazione}%
  \def\refname{Riferimenti bibliografici}%
  \def\abstractname{Sommario}%
  \def\bibname{Bibliografia}%
  \def\chaptername{Capitolo}%
  \def\appendixname{Appendice}%
  \def\contentsname{Indice}%
  \def\listfigurename{Elenco delle figure}%
  \def\listtablename{Elenco delle tabelle}%
  \def\indexname{Indice analitico}%
  \def\figurename{Figura}%
  \def\tablename{Tabella}%
  \def\partname{Parte}%
  \def\enclname{Allegati}%
  \def\ccname{e~p.~c.}%
  \def\headtoname{Per}%
  \def\pagename{Pag.}%    % in Italian the abbreviation is preferred
  \def\seename{vedi}%
  \def\alsoname{vedi anche}%
  \def\proofname{Dimostrazione}%
  \def\glossaryname{Glossario}%
  }%
```

\dateitalian The macro \dateitalian redefines the command \today to produce Italian dates.

```
\def\dateitalian{%
  \def\today{\number\day~\ifcase\month\or
    gennaio\or febbraio\or marzo\or aprile\or maggio\or giugno\or
    luglio\or agosto\or settembre\or ottobre\or novembre\or
    dicembre\fi\space \number\year}}%
```

\italianhyphenmins The italian hyphenation patterns can be used with both \lefthyphenmin and \righthyphenmin set to 2.

```
\providehyphenmins{\CurrentOption}{\tw@\tw@}
```

\extrasitalian Lower the chance that clubs or widows occur.
\noextrasitalian

```
\addto\extrasitalian{%
  \babel@savevariable\clubpenalty
  \babel@savevariable\widowpenalty
  \babel@savevariable\@clubpenalty
  \clubpenalty3000\widowpenalty3000\@clubpenalty\clubpenalty}%
```

Never ever break a word between the last two lines of a paragraph in italian texts.

```
\addto\extrasitalian{%
  \babel@savevariable\finalhyphendemerits
  \finalhyphendemerits50000000}%
```

In order to enable the hyphenation of words such as “nell’altezza” we give the ’ a non-zero lower case code. When we do that TEX finds the following hyphenation points `nel-l’al-tez-za` instead of none.

```
\addto\extrasitalian{%
  \lccode‘’=‘’}%
\addto\noextrasitalian{%
  \lccode‘’=0}%
```

1.1 Support for etymological hyphenation

In his article on Italian hyphenation [1] Beccari pointed out that the Italian language gets hyphenated on a phonetic basis, although etymological hyphenation is allowed; this is in contrast with what happens in Latin, for example, where etymological hyphenation is always used. Since the patterns for both languages would become too complicated in order to cope with etymological hyphenation, in his paper Beccari proposed the definition of an active character ‘_’ such that it could insert a “soft” discretionary hyphen at the compound word boundary. For several reasons that idea and the specific active character proved to be unpractical and was abandoned.

This problem is so important with the majority of the European languages, that babel from the very beginning developed the tradition of making the " character active so as to perform several actions that turned useful with every language. One of these actions consisted in defining the shortcut `"|` that was extensively used in German and in many other languages in order to insert a discretionary hyphen such that hyphenation would not be precluded in the rest of the word as it happens with the standard TEX command `\-`.

Meanwhile the `ec` fonts with the double Cork encoding (thus formerly called the `dc` fonts) have become more or less standard and are widely used by virtually all Europeans that write languages with many special national characters; by so doing they avoid the use of the `\accent` primitive which would be required with the standard `cm` fonts; with the latter fonts the primitive command `\accent` is such that hyphenation becomes almost impossible, in any case strongly impeached.

The `ec` fonts contain a special character, named “compound word mark”, that occupies position 23 in the font scheme and may be input with the sequence `^^W`. Up to now, apparently, this special character has never been used in a practical way for the typesetting of languages rich of compound words; also it has never been inserted in the hyphenation pattern files of any language. Beccari modified his pattern file `ithyph.tex v4.8b` for Italian so as to contain five new patterns that involve `^^W`, and he tried to give the babel active character " a new shortcut definition, so as to allow the insertion of the “compound word mark” in the proper place within any word where two semantic fragments join up. With such facility for marking the compound word boundaries, etymological hyphenation becomes possible even if the patterns know nothing about etymology (but the typesetter hopefully does!). In Italian such etymological hyphenation is desirable with technical terms, chemical names, and the like.

Even this solution proved to be inconvenient on certain UN*X platforms, so

Beccari resorted to another approach that uses the babel active character " and relies on the category code of the character that follows ".

```
\initiate@active@char{"}%
\addto\extrasitalian{\bbl@activate{"}\languageshorthands{italian}}%
```

\it@cwm The active character " is now defined for language italian so as to perform different actions in math mode compared to text mode; specifically in math mode a double quote is inserted so as to produce a double prime sign, while in text mode the temporary macro \it@next is defined so as to defer any further action until the next token category code has been tested.

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

\it@cwm The \it@next service control sequence is such that upon its execution a temporary variable \it@temp is made equivalent to the next token in the input list without actually removing it. Such temporary token is then tested by the macro \it@cwm and if it is found to be a letter token, then it introduces a compound word separator control sequence \it@allowhyphens whose expansion introduces a discretionary hyphen and an unbreakable space; in case the token is not a letter, then it is tested against $|_{12}$: if so a compound word separator is inserted and the | token is removed, otherwise another test is performed so as to see if another double quote sign follows: in this case a double open quote mark is inserted, otherwise two other tests are performed so as to see if guillemets have to be inserted, otherwise nothing is done. The double quote shortcut for inserting a double open quote sign is useful for people who are inputting Italian text by means of an Italian keyboard that unfortunately misses the grave or backtick key. By this shortcut "" becomes equivalent to `` for inserting raised open high double quotes.

```
\def\it@@cwm{\nobreak\discretionary{-}{}{}\nobreak\hskip\z@skip}%
\def\it@@ocap#1{\it@ocap}\def\it@@ccap#1{\it@ccap}%
\DeclareRobustCommand*{\it@cwm}{\let\it@@next\relax
\ifcat\noexpand\it@temp a%
    \def\it@@next{\it@@cwm}%
\else
    \if\noexpand\it@temp \string|%
        \def\it@@next{\it@@cwm\@gobble}%
    \else
        \if\noexpand\it@temp \string<%
            \def\it@@next{\it@@ocap}%
        \else
            \if\noexpand\it@temp \string>%
                \def\it@@next{\it@@ccap}%
            \else
```

```
                \if\noexpand\it@temp\string/%
                    \def\it@@next{\slash\@gobble}%
                \else
                    \ifx\it@temp"%
                        \def\it@@next{``\@gobble}%
                    \fi
                \fi
            \fi
        \fi
    \fi
\fi
\it@@next}%
```

By this definition of " if one types `macro"istruzione` the possible break points become ma-cro-istru-zio-ne, while without the " mark they would be ma-croi-stru-zio-ne, according to the phonetic rules such that the `macro` prefix is not taken as a unit. A chemical name such as `des"clor"fenir"amina"cloridrato` is breakable as des-clor-fe-nir-ami-na-clo-ri-dra-to instead of de-sclor-fe-ni-ra-mi-na-...

In other language description files a shortcut is defined so as to allow a break point without actually inserting any hyphen sign; examples are given such as entrada/salida; actually if one wants to allow a breakpoint after the slash, it is much clearer to type `\slash` instead of `/` and LaTeX does everything by itself; here the shortcut `"/` was introduced to stand for `\slash` so that one can type `input"/output` and allow a line break after the slash. This shortcut works only for the slash, since in Italian such constructs are extremely rare.

Attention: the expansion of " takes place before the actual expansion of OT1 or T1 accented sequences such as `\‘{a}`; therefore this etymological hyphenation facility works as it should only when the semantic word fragments *do not start* with an accented letter; this in Italian is always avoidable, because compulsory accents fall only on the last vowel, but it may be necessary to mark a compound word where one or more components come from a foreign language and contain diacritical marks according to the spelling rules of that language. In this case the special shorthand `"|` may be used that performs exactly as " normally does, except that the | sign is removed from the token input list: `kilo"|{\"o}rsted` gets hyphenated as `ki-lo-ör-sted`.

1.2 Facilities required by the ISO 31/XI regulations

The ISO 31/XI regulations require that units of measure are typeset in upright font in any circumstance, math or text, and that in text mode they are separated from the numerical indication of the measure with an unbreakable (thin) space. The command `\unit` that was defined for achieving this goal happened to conflict with the homonymous command defined by the package `units.sty`; we therefore need to test if that package has already been loaded so as to avoid conflicts; we assume that if the user loads that package, s/he wants to use that package facilities and command syntax.

The same regulations require also that super and subscripts (apices and

pedices) are in upright font, *not in math italics*, when they represent "adjectives" or appositions to mathematical or physical variables that do not represent countable or measurable entities such as, for example, $V_{\max}$ or V_{rms} for a maximum or a root mean square voltage, compared to V_i or V_T as the i-th voltage in a set, or a voltage that depends on the thermodynamic temperature T. See [2] for a complete description of the ISO regulations in connection with typesetting.

More rarely it happens to use superscripts that are not mathematical variables, such as the notation $\mathbf{A}^{\mathrm{T}}$ to denote the transpose of matrix $\mathbf{A}$; text superscripts are useful also as ordinals or in old fashioned abbreviations in text mode; for example the feminine ordinal 1ª or the old fashioned obsolete abbreviation F^lli for Fratelli in company names (compare with "Bros." for brothers in American English); text subscripts are mostly used in logos.

\unit
\ap
\ped

First we define the new (internal) commands `\bbl@unit`, `\bbl@ap`, and `\bbl@ped` as robust ones.

```
\@ifpackageloaded{units}{}{%
  \DeclareRobustCommand*{\bbl@unit}[1]{%
    \textormath{\,\mbox{#1}}{\,\mathrm{#1}}}%
  }%
\DeclareRobustCommand*{\bbl@ap}[1]{%
  \textormath{\textsuperscript{#1}}{^{\mathrm{#1}}}}%
\DeclareRobustCommand*{\bbl@ped}[1]{%
  \textormath{$_{\mbox{\fontsize\sf@size\z@
        \selectfont#1}}$}{_\mathrm{#1}}}%
```

Then we can use `\let` to define the user level commands, but in case the macros already have a different meaning before entering in Italian mode typesetting, we first memorize their meaning so as to restore them on exit.

```
\@ifpackageloaded{units}{}{%
  \addto\extrasitalian{%
    \babel@save\unit\let\unit\bbl@unit}%
  }%
\addto\extrasitalian{%
  \babel@save\ap\let\ap\bbl@ap
  \babel@save\ped\let\ped\bbl@ped
  }%
```

1.3 Accents

Most of the other language description files introduce a number of shortcuts for inserting accents and other language specific diacritical marks in a more comfortable way compared with the lengthy standard TEX conventions. When an Italian keyboard is being used on a Windows based platform, it exhibits such limitations that up to now no convenient shortcuts have been developed; the reason lies in the fact that the Italian keyboard lacks the grave accent (also known as "backtick"), which is compulsory on all accented vowels except the 'e', but, on the opposite, it carries the keys with all the accented lowercase vowels; the keyboard lacks also the tie ~ (tilde) key, while the curly braces require pressing three keys simultaneously.

The best solution Italians have found so far is to use a smart editor that accepts shortcut definitions such that, for example, by striking `"(` one gets directly `{` on the screen and the same sign is saved into the `.tex` file; the same smart editor should be capable of translating the accented characters into the standard TEX sequences when writing a file to disk (for the sake of file portability), and to transform the standard TEX sequences into the corresponding signs when loading a `.tex` file from disk to memory. Such smart editors do exist and can be downloaded from the CTAN archives.

For what concerns the missing backtick key, which is used also for inputting the open quotes, it must be noticed that the shortcut `""` described above completely solves the problem for *double* raised open quotes; according to the traditions of particular publishing houses, since there are no compulsory regulations on the matter, the French guillemets may be used; in this case the T1 font encoding solves the problem by means of its built in ligatures `<<` and `>>`. But...

1.4 *Caporali* or French double quotes

Although the T1 font encoding ligatures solve the problem, there are some circumstances where even the T1 font encoding cannot be used, either because the author/typesetter wants to use the OT1 encoding, or because s/he uses a font set that does not comply completely with the T1 font encoding; some virtual fonts, for example, are supposed to implement the double Cork font encoding but actually miss some glyphs; one such virtual font set is given by the `ae` virtual fonts, because they are supposed to implement such double font encoding simply using the `cm` fonts, of which the type 1 PostScript version exists and is freely available. Since guillemets (in Italian *caporali*) do not exist in any `cm` latin font, their glyphs must be substituted with something else that approaches them.

Since in French typesetting guillemets are compulsory, the French language definition file resorts to a clever font substitution; such file exploits the LaTeX 2_ε font selection machinery so as to get the guillemets from the Cyrillic fonts, because it suffices to locally change the default encoding. There are several sets of Cyrillic fonts, but the ones that obey the OT2 font encoding are generally distributed with all recent implementations of the TEX software; they are part of the American Mathematical Society fonts and come both as METAFONT source files and Type 1 PostScript `.pfb` files. The availability of such fonts should be guaranteed by the presence of the `OT2cmr.fd` font description file. Actually the presence of this file does not guarantee the completeness of your TEX implementation; should LaTeX complain about a missing Cyrillic `.tfm` file (that kind of file that contains the font metric parameters) and/or about missing Cyrillic (.mf) files, then your TEX system is *incomplete* and you should download such fonts from the CTAN archives. Temporarily you may issue the command `\LtxSymbCaporali` so as to approximate the missing glyphs with the LaTeX symbol fonts. In some case warning messages are issued so as to inform the typesetter about the necessity of resorting to some *poor man* solution.

In spite of these alternate fonts, we must avoid invoking unusual fonts if the available encoding allows to use built in caporali. As far as I know (CB) the only

T1-encoded font families that miss the guillemets are the AE ones; we therefore first test if the default encoding id the T1 one and in this case if the AE families are the default ones; in order for this to work properly it is necessary to load these optional packages *before* babel. If the T1 encoding is not the default one when the Italian language is specified, then some substitutions must be done.

\LtxSymbCaporali
\it@ocap
\it@ccap

We define some macros for substituting the default guillemets; first the emulation by means of the LaTeX symbols; each one of these macro sets actually redefines the control sequences \it@ocap and \it@ccap that are the ones effectively activated by the shortcuts "< and ">.

```
\def\LtxSymbCaporali{%
    \DeclareRobustCommand*{\it@ocap}{\mbox{%
       \fontencoding{U}\fontfamily{lasy}\selectfont(\kern-0.20em(}%
       \ignorespaces}%
    \DeclareRobustCommand*{\it@ccap}{\ifdim\lastskip>\z@\unskip\fi
    \mbox{%
       \fontencoding{U}\fontfamily{lasy}\selectfont)\kern-0.20em)}}%
}%
```

Then the substitution with any specific font that contains such glyphs; it might be the CBgreek fonts, the Cyrillic one, the super-cm ones, the lm ones, or any other the user might prefer (the code is adapted from the one that appears in the frenchb.ld file; thanks to Daniel Flipo). By default if the user did not select the T1 encoding, the existence of the CBgreek fonts is tested; if they exist the guillemets are taken from this font, and since its families are a superset of the default CM ones and they apply also to typeset slides with the standard class slides. If the CBgreek fonts are not found, then the existence of the Cyrillic ones is tested, although this choice is not suited for typesetting slides; otherwise the poor man solution of the LaTeX special symbols is used. In any case the user can force the use of the Cyrillic guillemets substitution by issuing the declaration \CyrillicCaporali just before the \begin{document} statement; in alternative the user can specify with

\CaporaliFrom{⟨*encoding*⟩}{⟨*family*⟩}{⟨*opening number*⟩}{⟨*closing number*⟩}

the encoding and family of the font s/he prefers, and the slot numbers of the opening and closing guillemets respectively. For example if the T1-encoded Latin Modern fonts are desired the specific command should be

```
\CaporaliFrom{T1}{lmr}{19}{20}
```

These user choices might be necessary for assuring the correct typesetting with fonts that contain the required glyphs and are available also in PostScript form so as to use them directly with pdflatex, for example.

```
\def\CaporaliFrom#1#2#3#4{%
  \DeclareFontEncoding{#1}{}{}%
  \DeclareTextCommand{\it@ocap}{T1}{%
    {\fontencoding{#1}\fontfamily{#2}\selectfont\char#3\ignorespaces}}%
  \DeclareTextCommand{\it@ccap}{T1}{\ifdim\lastskip>\z@\unskip\fi%
```

```
    {\fontencoding{#1}\fontfamily{#2}\selectfont\char#4}}%
  \DeclareTextCommand{\it@ocap}{OT1}{%
    {\fontencoding{#1}\fontfamily{#2}\selectfont\char#3\ignorespaces}}%
  \DeclareTextCommand{\it@ccap}{OT1}{\ifdim\lastskip>\z@\unskip\fi%
    {\fontencoding{#1}\fontfamily{#2}\selectfont\char#4}}}
```

Then we set a boolean variable and test the default family; if such family has a name that starts with the letters "ae" then we have no built in guillemets; of course if the AE font family is chosen after the babel package is loaded, the test does not perform as required.

```
\def\get@ae#1#2#3!{\def\bbl@ae{#1#2}}%
\def\@ifT@one@noCap{\expandafter\get@ae\f@family!%
\def\bbl@temp{ae}\ifx\bbl@ae\bbl@temp\expandafter\@firstoftwo\else
    \expandafter\@secondoftwo\fi}%
```

We set another couple of boolean variables for testing the existence of the CBgreek or the Cyrillic fonts

```
\newif\if@CBgreekEncKnown
\IfFileExists{lgrcmr.fd}%
      {\@CBgreekEncKnowntrue}{\@CBgreekEncKnownfalse}
\newif\if@CyrEncKnown
\IfFileExists{ot2cmr.fd}%
    {\@CyrEncKnowntrue}{\@CyrEncKnownfalse}%
```

\CBgreekCaporali
\CyrillicCaporali
\T@unoCaporali

Next we define the macros \CBgreekCaporali, \T@unoCaporali, and \CyrillicCaporali; with the latter one we test the loaded class, and if it's slides nothing gets done. In any case each one of these declarations, if used, must be specified in the preamble.

```
\def\CBgreekCaporali{\@ifclassloaded{slides}{%
      \IfFileExists{lgrlcmss.fd}{\DeclareFontEncoding{LGR}{}{}%
            \DeclareRobustCommand*{\it@ccap}%
                  {\ifdim\lastskip>\z@\unskip\fi
                        {\fontencoding{LGR}\selectfont))}}%
            \DeclareRobustCommand*{\it@ocap}%
                  {{\fontencoding{LGR}\selectfont((}\ignorespaces}}%
            {\LtxSymbCaporali}}%
      {\DeclareFontEncoding{LGR}{}{}%
      \DeclareRobustCommand*{\it@ccap}%
            {\ifdim\lastskip>\z@\unskip
            \fi{\fontencoding{LGR}\selectfont))}}%
      \DeclareRobustCommand*{\it@ocap}%
            {{\fontencoding{LGR}\selectfont((}\ignorespaces}}%
      }%
\def\CyrillicCaporali{\@ifclassloaded{slides}{\relax}%
      {\DeclareFontEncoding{OT2}{}{}%
      \DeclareRobustCommand*{\it@ccap}%
            {\ifdim\lastskip>\z@\unskip\fi
            {\fontencoding{OT2}\selectfont\char62\relax}}%
      \DeclareRobustCommand*{\it@ocap}%
            {{\fontencoding{OT2}\selectfont\char60\relax}\ignorespaces}}}%
\@onlypreamble{\CBgreekCaporali}\@onlypreamble{\CyrillicCaporali}%
```

```
\def\T@unoCaporali{\DeclareRobustCommand*{\it@ocap}{<<\ignorespaces}%
     \DeclareRobustCommand*{\it@ccap}{\ifdim\lastskip>\z@\unskip\fi>>}}%
```

Now we can do some real setting; first we start testing the encoding; if the encoding is T1 we test if the font family is the AE one; if so, we further test for other possibilities

```
\ifx\cf@encoding\bbl@t@one
  \@ifT@one@noCap{%
     \if@CBgreekEncKnown
        \CBgreekCaporali
     \else
        \if@CyrEncKnown
           \CyrilicCaporali
        \else
           \LtxSymbCaporali
        \fi
     \fi}%
     {\T@unoCaporali}%
```

But if the default encoding is not the T1 one, then the substitutions must be performed.

```
\else
     \if@CBgreekEncKnown
        \CBgreekCaporali
     \else
        \if@CyrEncKnown
           \CyrilicCaporali
        \else
           \LtxSymbCaporali
        \fi
     \fi
\fi
```

1.5 Finishing commands

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of `@` to its original value.

```
\ldf@finish{italian}%
⟨/code⟩
```

References

[1] Beccari C., "Computer Aided Hyphenation for Italian and Modern Latin", TUGboat vol. 13, n. 1, pp. 23-33 (1992).

[2] Beccari C., "Typesetting mathematics for science and technology according to ISO 31/XI", TUGboat vol. 18, n. 1, pp. 39-48 (1997).