Babel, a multilingual package for use with LaTeX's standard document classes[*]

Johannes Braams
Kooienswater 62
2715 AJ Zoetermeer
The Netherlands
`JLBraams@cistron.nl`

Printed May 21, 1998

Abstract

The standard distribution of LaTeX contains a number of document classes that are meant to be used, but also serve as examples for other users to create their own document classes. These document classes have become very popular among LaTeX users. But it should be kept in mind that they were designed for American tastes and typography. At one time they contained a number of hard-wired texts. This report describes babel, a package that makes use of the new capabilities of TeX version 3 to provide an environment in which documents can be typeset in a non-american language or in more than one language.

Contents

[*]During the development ideas from Nico Poppelier, Piet van Oostrum and many others have been used. Bernd Raichle has provided many helpful suggestions.

1 The user interface

The user interface of this package is quite simple. It consists of a set of commands that switch from one language to another and a set of commands that deal with shorthands. It is also possible to find out out what the current language is.

`\selectlanguage` When a user wants to switch from one language to another he can do so using the macro `\selectlanguage`. This macro takes the language, defined previously by a language definition file, as its argument. It calls several macros that should be defined in the language definition files to activate the special definitions for the language chosen.

`otherlanguage` The environment `otherlanguage` does basically the same as `\selectlanguage`, except the language change is local to the environment. For mixing left-to-right typesetting with right-to-left typesetting the use of this environment is a prerequisite. The language to switch to is specified as an argument to `\begin{otherlanguage}`.

`\foreignlanguage` The command `\foreignlanguage` takes two arguments, the second argument is a phrase to be typeset according to the rules of the language named in its first argument. This command only switches the extra definitions and the hyphenation rules for the language, *not* the names and dates.

`otherlanguage*` In the environment `otherlanguage*` only the typesetting is done according to the rules of the other language, but the text-strings such as 'figure', 'table', etc. are left as they were set outside this environment.

`\languagename` The control sequence `\languagename` contains the name of the current language.

`\iflanguage` If more than one language is used it might be necessary to know which language is active at a specific time. This can be checked by a call to `\iflanguage`. This macro takes three arguments. The first argument is the name of a language, the second and third arguments are the actions to take if the result of the test is `true` or `false` respectively.

`\useshorthands` The command `\useshorthands` initiates the definition of user-defined shorthand sequences. It has one argument, the character which starts these personal shorthands.

`\defineshorthand` The command `\defineshorthand` takes two arguments, the first of which is a one or two character sequence, the second argument is the code the shorthand should expand to.

`\aliasshorthand` The command `\aliasshorthand` can be used to let another character perform the same functions as the default shorthand character. If one prefers for example to use the character `/` over `"` in typing polish texts this can be acheived by entering `\aliasshorthand{"}{/}`.

`\languageshorthands` The command `\languageshorthands` can be used to switch the shorthands on the language level. It takes one argument, the name of a language. Note that for this to work the language should have been specified as an option when loading the `babel` package.

1.1 Languages supported by Babel

In the following table all the languages supported by Babel are listed, together with the names of the options with which you can load babel for each language.

Afrikaans	afrikaans
Bahasa	bahasa
Breton	breton
Catalan	catalan
Croatian	croatian
Czech	czech
Danish	danish
Dutch	dutch
English	english, USenglish, american, UKenglish, british
Esperanto	esperanto
Estonian	estonian
Finnish	finnish
French	french, francais
Galician	galician
German	austrian, german, germanb
Greek	greek
Hebrew	hebrew
Hungarian	magyar, hungarian
Irish Gaelic	irish
Italian	italian
Lower Sorbian	lowersorbian
Norwegian	norsk, nynorsk
Polish	polish
Portuguese	portuges, portuguese, brazilian, brazil
Romanian	romanian
Russian	russian
Scottish Gaelic	scottish
Spanish	spanish
Slovakian	slovak
Slovenian	slovene
Swedish	swedish
Turkish	turkish
Upper Sorbian	uppersorbian
Welsh	welsh

For some languages babel supports the options activeacute and activegrave; for typestting russian texts babel knows about the options LWN and LCY to specify the fontencoding of the cyrillic font used. Currently only LWN is supported.

1.2 Workarounds

When you use the document class `book` *and* you use `\ref` inside the argument of `\chapter` you will experience the problem that LaTeX will keep complaining about an undefined label. The reason is that the argument of `\ref` is passed through `\uppercase` at some time during processing. To prevent such problems you could revert to using uppercase labels, or you can use `\lowercase{\ref{foo}}` inside the argument of `\chapter`.

2 Changes for LaTeX 2ε

With the advent of LaTeX 2ε the interface to babel in the preamble of the doument has changed. With LaTeX2.09 one used to call up the babel system with a line such as:

```
\documentstyle[dutch,english]{article}
```

which would tell LaTeX that the document would be written in two languages, dutch and english and that english would be the first language in use.

The LaTeX 2ε way of providing the same information is:

```
\documentclass{article}
\usepackage[dutch,english]{babel}
```

or, making dutch and english global options in order to let other packages detect and use them:

```
\documentclass[dutch,english]{article}
\usepackage{babel}
\usepackage{varioref}
```

In this last example the package `varioref` will also see the options and will be able to use them.

3 Changes in Babel version 3.6

In Babel version 3.6 a number of bugs that were found in version 3.5 are fixed. Also a number of changes and additions have occured:

- A new environment otherlanguage* is introduced. it only switches the 'specials', but leaves the 'captions' untouched.
- The shorthands are no longer fully expandable. Some problems could only be solved by peeking at the token following an active character. The advantage is that `'{}a` works as expected for languages that have the `'` active.
- Support for typesetting french texts is much enhanced; the file `francais.ldf` is now replaced by `frenchb.ldf` which is maintained by Daniel Flipo.
- Support for typesetting the russian language is again available. The languange definition file was originally developped by Olga Lapko from cyrtug. The fonts needed to typeset the russian language are now part of the babel distribution. The support is not yet up to the level which is needed according to Olga, but this is a start.
- Support for typesetting greek texts is now also available. What is offered in this release is a first attempt; it will be enhanced later on by Yannis Haralambous.

- in babel 3.6j some hooks have been added for the development of support for Hebrew typesetting.
- Support for typesetting texts in Afrikaans (a variant of Dutch, spoken in South Africa) has been added to `dutch.ldf`.
- Support for typesetting welsh texts is now available.
- A new command `\aliasshorthand` is introduced. It seems that in Poland various conventions are used to type the necessary polish letters. It is now possible to use the character `/` as a shorthand character instead of the character `"` by issuing the command `\aliasshorthand{"}{/}`.
- The shorthand mechanism now deals correctly with characters that are already active.
- Shorthand characters are made active at `\begin{document}`, not earlier. This is to prevent problems with other packages.
- A *preambleonly* command `\substitutefontfamily` has been added to create `.fd` files on the fly when the font families of the latin text differ from the families used for the cyrillic or greek parts of the text.
- Three new comands `\LdfInit`, `\ldf@quit` and `\ldf@finish` are introduced that perform a number of standard tasks.

4 Changes in Babel version 3.5

In Babel version 3.5 a lot of changes have been made when compared with the previous release. Here is a list of the most important ones:

- the selection of the language is delayed untill `\begin{document}`, this has the consequence that you need to add appropriate `\selectlanguage` commands if you include `\hyphenation` lists in the preamble of your document.
- babel now has a language environment and a new command `\foreignlanguage`;
- the way active characters are dealt with is completely changed. They are called 'shorthands'; one can have three levels of shorthands: on the user level, the language level and on 'system level'. A consequence of the new way of handling active characters is that they are now written to auxiliary files 'verbatim';
- A language change now also writes information in the `.aux` file as the change might also affect typesetting the table of contents. The consequence is that an .aux file generated by a LaTeX format with babel preloaded gives errors when read with a LaTeX format without babel, but I think this problaly doesn't occur;
- babel is now compatible with the `inputenc` and `fontenc` packages;
- the language definition files now have a new extension, `ldf`;
- the syntax of the file `language.dat` is extended to be compatible with the `french` package by Bernard Gaulle;

- each language definition file looks for a configuration file which has the same name, but the extension `.cfg`. It can cantain any valid LaTeX code.

5 The interface between the core of babel and the language definition files

In the core of the babel system two macros are defined that are to be used in language definition files. Their purpose is to make a new language known.

`\addlanguage` The macro `\addlanguage` is a non-outer version of the macro `\newlanguage`, defined in `plain.tex` version 3.x. For older versions of `plain.tex` and `lplain.tex` a substitute definition is used.

`\adddialect` The macro `\adddialect` can be used in the case where two languages can (or have to) use the same hyphenation patterns. This can alos be useful when a user wants to use a language for which no patterns are preloaded in the format. In such a case the default behaviour of the babel system is to define this language as a 'dialect' of the language for which the patterns were loaded as `\language0`.

The language definition files have to conform to a number of conventions. The reason for this is that these files have to fill in the gaps left by the common code in `babel.def`, i. e., the definitions of the macros that produce texts. Also the language-switching possibility which has been built into the babel system has its implications.

The following assumptions are made:

- Some of the language-specific definitions might be used by plain TeX users, so the files have to be coded such that they can be read by LaTeX as well as by plain TeX. The current format can be checked by looking at the value of the macro `\fmtname`.

- The common part of the babel system redefines a number of macros and environments (defined previously in the document style) to put in the names of macros that replace the previously hard-wired texts. These macros have to be defined in the language definition files.

- The language definition files define five macros, used to activate and deactivate the language-specific definitions. These macros are `\`⟨*lang*⟩`hyphenmins`, `\captions`⟨*lang*⟩, `\date`⟨*lang*⟩, `\extras`⟨*lang*⟩and `\noextras`⟨*lang*⟩; where ⟨*lang*⟩ is either the name of the language definition file or the name of the LaTeX option that is to be used. These macros and their functions are discussed below.

- When a language definition file is loaded, it can define `\l@`⟨*lang*⟩to be a dialect of `\language0` when `\l@`⟨*lang*⟩is undefined.

- The languagedefinition files can be read in the preamble of the document, but also in the middle of document processing. This means that they have to function independently of the current `\catcode` of the `@` sign.

`\langhyphenmins` The macro `\`⟨*lang*⟩`hyphenmins` is used to store the values of the `\lefthyphenmin` and `\righthyphenmin`.

`\captionslang` The macro `\captions`⟨*lang*⟩defines the macros that hold the texts to replace the original hard-wired texts.

`\datelang` The macro `\date`⟨*lang*⟩defines `\today` and

`\extraslang` The macro `\extras`⟨*lang*⟩contains all the extra definitions needed for a specific language.

`\noextraslang` Because we want to offer the user the possibility to switch between languages and we do not know in what state TEX might be after the execution of `\extras`⟨*lang*⟩, a macro that brings TEX into a predefined state is needed. It will be no surprise that the name of this macro is `\noextras`⟨*lang*⟩.

`\main@language` To postpone the activation of the definitions needed for a language untill the beginning of a document, all language definition files should use `\main@language` instead of `\selectlanguage`. This will just store the name of the language and the proper language will be activated at the start of the document.

`\LdfInit` The macro `\LdfInit` performs a couple of standard checks that have to be made at the beginning of a language definition file, such as checking the category code of the @-sign, preventing that the `.ldf` file is processed twice, etc.

`\ldf@quit` The macro `\ldf@quit` performs a couple of tasks that need to be taken care of when a `.ldf` file was processed earlier. These tasks include the resetting of the category code of the @-sign, preparing the language to be activated at `\begin{document}` time and ending the input stream.

`\ldf@finish` The macro `\ldf@finish` performs a couple of tasks that need to be taken care of at the end of each `.ldf` file. These tasks include the resetting of the category code of the @-sign, the loading of a local configuration file and preparing the language to be activated at `\begin{document}` time.

`\loadlocalcfg` At the end of the processing of a language definition file LATEX can be instructed to load a local configuration file. This file can for instance be used to add strings to `\captions`⟨*lang*⟩ in order to support local document classes. The user will be informed of the fact that this configuration file is loaded. This macro is called by `\finish@ldf`.

`\subsitutefontfamily` This command takes three arguments, a font encoding and two font family names. It creates a font description file for the first font in the given encoding. This `.fd` file will instruct LATEX to use a font from the second family when a font from the first family in the given encoding seems to needed.

5.1 Support for active characters

In quite a number of language definition files, active characters are introduced. To facilitate this, some support macros are provided.

`\initiate@active@char` The internal macro `\initiate@active@char` is used in language definition files to instruct LATEX to give a character the category code 'active'. When a character has been made active it will remain that way untill the end of then document. Its definition may vary.

`\bbl@activate`
`\bbl@deactivate` The command `\bbl@activate` is used to change the way an active character expands. `\bbl@activate` 'switches on' the active behaviour of the character. `\bbl@deactive` lets the active character expand to its former (mostly) non-active self.

`\declare@shorthand` The macro `\declare@shorthand` is used to define the various shorthands. It takes three arguments, the name for the collection of shorthands this definition belongs to; the character (sequence) that makes up the shorthand i.i. `~` or `"a` and the code to be executed when the shorthand is encountered.

`\bbl@add@special`
`\bbl@remove@special` The TEXbook states: "Plain TEX includes a macro called `\dospecials` that

is essentially a set macro, representing the set of all characters that have a special category code." [1, p. 380] It is used to set text 'verbatim'. To make this work if more characters get a special category code, you have to add this character to the macro `\dospecial`. LaTeX adds another macro called `\@sanitize` representing the same character set, but without the curly braces. The macros `\bbl@add@special`⟨*char*⟩ and `\bbl@remove@special`⟨*char*⟩ add and remove the character ⟨*char*⟩ to these two sets.

5.2 Support for saving macro definitions

Language definition files may want to *re*define macros that already exist. Therefore a mechanism for saving (and restoring) the original definition of those macros is provided. We provide two macros for this[1].

`\babel@save` To save the current meaning of any control sequence the macro `\babel@save` is provided. It takes one argument, ⟨*csname*⟩, the control sequence for which the meaning has to be saved.

`\babel@savevariable` A second macro is provided to save the current value of a variable. In this context anything that is allowed after the `\the` primitive is considered to be a variable. The macro takes one argument, the ⟨*variable*⟩.

The effect of the aforementioned macros is that a piece of code is appended to the current definition of `\originalTeX`. When `\originalTeX` is expanded this code restores the previous definition of the control sequence or the previous value of the variable.

5.3 Support for extending macros

`\addto` The macro `\addto{`⟨*control sequence*⟩`}{`⟨*TeX code*⟩`}` can be used to extend the definition of a macro. The macro need not be defined. This macro can, for instance, be used in adding instructions to a macro like `\extrasenglish`.

5.4 Macros common to a number of languages

`\allowhyphens` In a couple of european languages compound words are used. This means that when TeX has to hyphenate such a compound word it only does that at the '`-`' that is used in such words. To allow hyphenation in the rest of such a compound word the macro `\allowhyphens` can be used.

`\set@low@box` For some languages quotes need to be lowered to the baseline. For this purpose the macro `\set@low@box` is available. It takes one argument and puts that argument in an `\hbox`, at the baseline. The result is available in `\box0` for further processing.

`\save@sf@q` Sometimes it is necessary to preserve the `\spacefactor`. For this purpose the macro `\save@sf@q` is available. It takes one argument, saves the current spacefactor, executes the argument and restores the spacefactor.

`\bbl@frenchspacing` `\bbl@nonfrenchspacing` The commands `\bbl@frenchspacing` and `\bbl@nonfrenchspacing` can be used to properly switch french spacing on and off.

[1]This mechanism was introduced by Bernd Raichle.

6 Compatibility with german.sty

The file german.sty has been one of the sources of inspiration for the babel system. Because of this I wanted to include german.sty in the babel system. To be able to do that I had to allow for one incompatibility: in the definition of the macro \selectlanguage in german.sty the argument is used as the ⟨*number*⟩ for an \ifcase. So in this case a call to \selectlanguage might look like \selectlanguage{\german}.

In the definition of the macro \selectlanguage in babel.def the argument is used as a part of other macronames, so a call to \selectlanguage now looks like \selectlanguage{german}. Notice the absence of the escape character. As of version 3.1a of babel both syntaxes are allowed.

All other features of the original german.sty have been copied into a new file, called germanb.sty[2].

Although the babel system was developed to be used with LaTeX, some of the features implemented in the language definition files might be needed by plain TeX users. Care has been taken that all files in the system can be processed by plain TeX.

7 Compatibility with the french package

It has been reported to me that the package french by Bernard Gaulle (gaulle@idris.fr) works together with babel.

Therefore, babel will first search for the file french.ldf when you give it the option french; then it will try to load frenchb.ldf. When you give babel the option francais it will only look for frenchb.ldf.

8 Identification

The file babel.sty[3] is meant for LaTeX 2_ε, therefore we make sure that the format file used is the right one.

The identification code for each file is something that was introduced in LaTeX 2_ε. When the command \ProvidesFile does not exist, v3.6 dummy definition is provided.

```
⟨*!package⟩
\ifx\ProvidesFile\@undefined
  \def\ProvidesFile#1[#2 #3 #4]{%
    \wlog{#4 #3 <#2>}%
⟨kernel & patterns⟩      \toks8{Babel <#3> and hyphenation patterns for }%
    }
⟨*kernel & patterns⟩
\else
```

In this case we save the orginal definition of \ProvidesFile in \bbl@tempa and restore it after we have stored the version of the file in \toks8.

```
  \let\bbl@tempa\ProvidesFile
  \def\ProvidesFile#1[#2 #3 #4]{%
```

[2]The 'b' is added to the name to distinguish the file from Partls' file.

[3]The file described in this section is called babel.dtx, has version number v3.6j and was last revised on 1998/03/24.

```
    \toks8{Babel <#3> and hyphenation patterns for }%
    \bbl@tempa{#1}[#2 #3 #4]%
    \let\ProvidesFile\bbl@tempa}
⟨/kernel & patterns⟩
\fi
⟨/!package⟩
```

Identify each file that is produced from this source file.

```
⟨+package⟩\ProvidesPackage{babel}
⟨+core⟩\ProvidesFile{babel.def}
⟨+kernel & patterns⟩\ProvidesFile{hyphen.cfg}
⟨+kernel&!patterns⟩\ProvidesFile{switch.def}
⟨+driver&!user⟩\ProvidesFile{babel.drv}
⟨+driver & user⟩\ProvidesFile{user.drv}
                    [1998/03/24 v3.6j
⟨+package⟩      The Babel package]
⟨+core⟩          Babel common definitions]
⟨+kernel⟩      Babel language switching mechanism]
⟨+driver⟩]
```

9 The Package File

In order to make use of the new features of LaTeX 2$_\varepsilon$, a new file is introdued to the babel system, `babel.sty`. This file is loaded by the `\usepackage` command and defines all the language options known in the babel system.

For all the languages supported we need to declare an option.

```
⟨*package⟩
\ifx\LdfInit\undefined\input{babel.def}\fi
\DeclareOption{afrikaans}{\input{dutch.ldf}}
\DeclareOption{american}{\input{english.ldf}}
```

Austrian is really a dialect of German.

```
\DeclareOption{austrian}{\input{germanb.ldf}}
\DeclareOption{bahasa}{\input{bahasa.ldf}}
\DeclareOption{brazil}{\input{portuges.ldf}}

\DeclareOption{brazilian}{\input{portuges.ldf}}
\DeclareOption{breton}{\input{breton.ldf}}

\DeclareOption{british}{\input{english.ldf}}
\DeclareOption{catalan}{\input{catalan.ldf}}
\DeclareOption{croatian}{\input{croatian.ldf}}
\DeclareOption{czech}{\input{czech.ldf}}
\DeclareOption{danish}{\input{danish.ldf}}
\DeclareOption{dutch}{\input{dutch.ldf}}

\DeclareOption{english}{\input{english.ldf}}
\DeclareOption{esperanto}{\input{esperant.ldf}}
\DeclareOption{estonian}{\input{estonian.ldf}}
\DeclareOption{finnish}{\input{finnish.ldf}}
```

The babel support or French used to be stored in `francais.ldf`; therefore the LaTeX2.09 option used to be francais. The hyphenation patterns may be loaded as either ‘french’ or as ‘francais’.

```
\DeclareOption{francais}{\input{frenchb.ldf}}
\DeclareOption{frenchb}{\input{frenchb.ldf}}
```

With $\LaTeX\,2_\varepsilon$ we can now also use the option french and still call the file `francais.ldf`.

```
\IfFileExists{french.ldf}{%
  \DeclareOption{french}{\input{french.ldf}}%
  }{%
  \DeclareOption{french}{\input{frenchb.ldf}}%
  }
\DeclareOption{galician}{\input{galician.ldf}}
\DeclareOption{german}{\input{germanb.ldf}}
\DeclareOption{germanb}{\input{germanb.ldf}}

\DeclareOption{greek}{\input{greek.ldf}}
\DeclareOption{hebrew}{\input{rlbabel.def}\input{hebrew.ldf}}
```

hungarian is just a synonym for magyar

```
\DeclareOption{hungarian}{\input{magyar.ldf}}
\DeclareOption{irish}{\input{irish.ldf}}
\DeclareOption{italian}{\input{italian.ldf}}
\DeclareOption{lowersorbian}{\input{lsorbian.ldf}}
\DeclareOption{magyar}{\input{magyar.ldf}}
\DeclareOption{norsk}{\input{norsk.ldf}}
```

For Norwegian two spelling variants are provided.

```
\DeclareOption{nynorsk}{\input{norsk.ldf}}
\DeclareOption{polish}{\input{polish.ldf}}
\DeclareOption{portuges}{\input{portuges.ldf}}
\DeclareOption{portuguese}{\input{portuges.ldf}}
\DeclareOption{romanian}{\input{romanian.ldf}}
\DeclareOption{russian}{\input{russianb.ldf}}
\DeclareOption{scottish}{\input{scottish.ldf}}
\DeclareOption{slovak}{\input{slovak.ldf}}
\DeclareOption{slovene}{\input{slovene.ldf}}
\DeclareOption{spanish}{\input{spanish.ldf}}
\DeclareOption{swedish}{\input{swedish.ldf}}
\DeclareOption{turkish}{\input{turkish.ldf}}
\DeclareOption{uppersorbian}{\input{usorbian.ldf}}
\DeclareOption{welsh}{\input{welsh.ldf}}

\DeclareOption{UKenglish}{\input{english.ldf}}
\DeclareOption{USenglish}{\input{english.ldf}}
```

Apart from all the language options we also have a few options that influence the behaviour of language definition files.

The following options don't do anything themselves, they are just defined in order to make it possible for language definition files to check if one of them was specified by the user.

```
\DeclareOption{activeacute}{}
\DeclareOption{activegrave}{}
```

The next option tells babel to leave shorthand characters active at the end of processing the package. This is *not* the default as it can cause problems with other packages, but for those who want to use the shorthand characters in the preamble of their documents this can help.

```
\DeclareOption{KeepShorthandsActive}{%
  \def\KeepShorthandsActive{}}
```

The options have to be processed in the order in which the user specified them:

```
\ProcessOptions*
```

\substitutefontfamily The command `\substitutefontfamily` creates an `.fd` file on the fly. The first argument is an encoding mnemonic, the second and third arguments are font family names.

```
\def\substitutefontfamily#1#2#3{%
  \immediate\openout15=#1#2.fd\relax
  \immediate\write15{%
    \string\ProvidesFile{#1#2.fd}%
    [\the\year/\two@digits{\the\month}/\two@digits{\the\day}
     \space generated font description file]^^J
    \string\DeclareFontFamily{#1}{#2}{}^^J
    \string\DeclareFontShape{#1}{#2}{m}{n}{<->ssub * #3/m/n}{}^^J
    \string\DeclareFontShape{#1}{#2}{m}{it}{<->ssub * #3/m/it}{}^^J
    \string\DeclareFontShape{#1}{#2}{m}{sl}{<->ssub * #3/m/sl}{}^^J
    \string\DeclareFontShape{#1}{#2}{m}{sc}{<->ssub * #3/m/sc}{}^^J
    \string\DeclareFontShape{#1}{#2}{b}{n}{<->ssub * #3/bx/n}{}^^J
    \string\DeclareFontShape{#1}{#2}{b}{it}{<->ssub * #3/bx/it}{}^^J
    \string\DeclareFontShape{#1}{#2}{b}{sl}{<->ssub * #3/bx/sl}{}^^J
    \string\DeclareFontShape{#1}{#2}{b}{sc}{<->ssub * #3/bx/sc}{}^^J
    }%
  \closeout15
  }
```

This command should only be used in the preamble of a document.

```
\@onlypreamble\substitutefontfamily
```

```
⟨/package⟩
```

10 The Kernel of Babel

The kernel of the babel system is stored in either `hyphen.cfg` or `switch.def` and `babel.def`. The file `hyphen.cfg` is a file that can be loaded into the format, which is necessary when you want to be able to switch hyphenation patterns. The file `babel.def` contains some TEX code that can be read in at run time. When `babel.def` is loaded it checks if `hyphen.cfg` is in the format; if not the file `switch.def` is loaded.

Because plain TEX users might want to use some of the features of the babel system too, care has to be taken that plain TEX can process the files. For this reason the current format will have to be checked in a number of places. Some of the code below is common to plain TEX and LATEX, some of it is for the LATEX case only.

When the command `\AtBeginDocument` doesn't exist we assume that we are dealing with a plain-based format. In that case the file `plain.def` is needed.

```
⟨*kernel | core⟩
\ifx\AtBeginDocument\@undefined
  \input plain.def\relax
\fi
⟨/kernel | core⟩
```

Check the presence of the command `\iflanguage`, if it is undefined read the file `switch.def`.

```
⟨*core⟩
\ifx\iflanguage\@undefined
  \input switch.def\relax
\fi
⟨/core⟩
```

10.1 Multiple languages

With TeX version 3.0 it has become possible to load hyphenation patterns for more than one language. This means that some extra administration has to be taken care of. The user has to know for which languages patterns have been loaded, and what values of `\language` have been used.

Some discussion has been going on in the TeX world about how to use `\language`. Some have suggested to set a fixed standard, i. e., patterns for each language should *always* be loaded in the same location. It has also been suggested to use the ISO list for this purpose. Others have pointed out that the ISO list contains more than 256 languages, which have *not* been numbered consecutively.

I think the best way to use `\language`, is to use it dynamically. This code implements an algorithm to do so. It uses an external file in which the person who maintains a TeX environment has to record for which languages he has hyphenation patterns *and* in which files these are stored[4]. When hyphenation exceptions are stored in a separate file this can be indicated by naming that file *after* the file with the hyphenation patterns.

This "configuration file" can contain empty lines and comments, as well as lines which start with an equals (`=`) sign. Such a line will instruct LaTeX that the hyphenation patterns just processed have to be known under an alternative name. Here is an example:

```
% File    : language.dat
% Purpose : tell iniTeX what files with patterns to load.
english    english.hyphenations
=british

dutch      hyphen.dutch exceptions.dutch % Nederlands
german hyphen.ger
```

As the file `switch.def` needs to be read only once, we check whether it was read before. If it was, the command `\iflanguage` is already defined, so we can stop processing.

```
⟨*kernel⟩
⟨*!patterns⟩
\expandafter\ifx\csname iflanguage\endcsname\relax \else
\expandafter\endinput
\fi
⟨/!patterns⟩
```

[4]This is because different operating systems sometimes use *very* different filenaming conventions.

`\language` Plain TeX version 3.0 provides the primitive `\language` that is used to store the current language. When used with a pre-3.0 version this function has to be implemented by allocating a counter.

```
10.17 \ifx\language\@undefined
10.18   \csname newcount\endcsname\language
10.19 \fi
```

`\last@language` Another counter is used to store the last language defined. For pre-3.0 formats an extra counter has to be allocated,

```
10.20 \ifx\newlanguage\@undefined
10.21   \csname newcount\endcsname\last@language
```

plain TeX version 3.0 uses `\count 19` for this purpose.

```
10.22 \else
10.23   \countdef\last@language=19
10.24 \fi
```

`\addlanguage` To add languages to TeX's memory plain TeX version 3.0 supplies `\newlanguage`, in a pre-3.0 environment a similar macro has to be provided. For both cases a new macro is defined here, because the original `\newlanguage` was defined to be `\outer`.

For a format based on plain version 2.x, the definition of `\newlanguage` can not be copied because `\count 19` is used for other purposes in these formats. Therefore `\addlanguage` is defined using a definition based on the macros used to define `\newlanguage` in plain TeX version 3.0.

```
10.25 \ifx\newlanguage\@undefined
10.26   \def\addlanguage#1{%
10.27     \global\advance\last@language \@ne
10.28     \ifnum\last@language<\@cclvi
10.29     \else
10.30         \errmessage{No room for a new \string\language!}%
10.31     \fi
10.32     \global\chardef#1\last@language
10.33     \wlog{\string#1 = \string\language\the\last@language}}
```

For formats based on plain version 3.0 the definition of `\newlanguage` can be simply copied, removing `\outer`.

```
10.34 \else
10.35   \def\addlanguage{\alloc@9\language\chardef\@cclvi}
10.36 \fi
```

`\adddialect` The macro `\adddialect` can be used to add the name of a dialect or variant language, for which an already defined hyphenation table can be used.

```
10.37 \def\adddialect#1#2{%
10.38     \global\chardef#1#2\relax
10.39     \wlog{\string#1 = a dialect from \string\language#2}}
```

`\iflanguage` Users might want to test (in a private package for instance) which language is currently active. For this we provide a test macro, `\iflanguage`, that has three arguments. It checks whether the first argument is a known language. If so, it compares the first argument with the value of `\language`. Then, depending on the result of the comparison, it executes either the second or the third argument.

```
\def\iflanguage#1#2#3{%
  \expandafter\ifx\csname l@#1\endcsname\relax
    \@nolanerr{#1}%
  \else
    \ifnum\csname l@#1\endcsname=\language #2%
    \else#3\fi
  \fi}
```

`\selectlanguage`

The macro `\selectlanguage` checks whether the language is already defined before it performs its actual task, which is to update `\language` and activate language-specific definitions.

To allow the call of `\selectlanguage` either with a control sequence name or with a simple string as argument, we have to use a trick to delete the optional escape character.

To convert a control sequence to a string, we use the `\string` primitive. Next we have to look at the first character of this string and compare it with the escape character. Because this escape character can be changed by setting the internal integer `\escapechar` to a character number, we have to compare this number with the character of the string. To do this we have to use TEX's backquote notation to specify the character as a number.

If the first character of the `\string`'ed argument is the current escape character, the comparison has stripped this character and the rest in the 'then' part consists of the rest of the control sequence name. Otherwise we know that either the argument is not a control sequence or `\escapechar` is set to a value outside of the character range 0–255.

If the user gives an empty argument, we provide a default argument for `\string`. This argument should expand to nothing.

```
\edef\selectlanguage{%
  \noexpand\protect
  \expandafter\noexpand\csname selectlanguage \endcsname
  }
```

Because the command `\selectlanguage` could be used in a moving argument it expands to `\protect\selectlanguage␣`. Therefore, we have to make sure that a macro `\protect` exists. If it doesn't it is `\let` to `\relax`.

```
\ifx\@undefined\protect\let\protect\relax\fi
```

As LATEX 2.09 writes to files *expanded* whereas LATEX 2$_\varepsilon$ takes care *not* to expand the arguments of `\write` statements we need to be a bit clever about the way we add information to `.aux` files. Therefore we introduce the macro `\xstring` which should expand to the right amount of `\string`'s.

```
\ifx\documentclass\@undefined
  \def\xstring{\string\string\string}
\else
  \let\xstring\string
\fi
```

```
\expandafter\def\csname selectlanguage \endcsname#1{%
  \edef\languagename{%
    \ifnum\escapechar=\expandafter`\string#1\@empty
    \else \string#1\@empty\fi}%
  \select@language{\languagename}%
```

We also write a command to change the current language in the auxiliary files.

```
  \if@filesw
    \protected@write\@auxout{}{\string\select@language{\languagename}}%
    \addtocontents{toc}{\xstring\select@language{\languagename}}%
    \addtocontents{lof}{\xstring\select@language{\languagename}}%
    \addtocontents{lot}{\xstring\select@language{\languagename}}%
  \fi}
```

First, check if the user asks for a known language. If so, update the value of `\language` and call `\originalTeX` to bring TEX in a certain pre-defined state.

```
\def\select@language#1{%
  \expandafter\ifx\csname date#1\endcsname\relax
    \@nolanerr{#1}%
  \else
    \language=\csname l@#1\endcsname\relax
    \originalTeX
```

The name of the language is stored in the control sequence `\languagename`. The contents of this control sequence could be tested in the following way:

```
\edef\tmp{\string english}
\ifx\languagename\tmp
    ...
\else
    ...
\fi
```

The construction with `\string` is necessary because `\languagename` returns the name with characters of category code `12` (other). Then we have to *re*define `\originalTeX` to compensate for the things that have been activated. To save memory space for the macro definition of `\originalTeX`, we construct the control sequence name for the `\noextras`⟨*lang*⟩command at definition time by expanding the `\csname` primitive.

```
    \expandafter\def\expandafter\originalTeX
        \expandafter{\csname noextras#1\endcsname
                      \let\originalTeX\@empty}%

    \languageshorthands{none}%
    \babel@beginsave
```

Now activate the language-specific definitions. This is done by constructing the names of three macros by concatenating three words with the argument of `\selectlanguage`, and calling these macros.

```
    \csname captions#1\endcsname
    \csname date#1\endcsname
    \csname extras#1\endcsname\relax
```

The switching of the values of `\lefthyphenmin` and `\righthyphenmin` is somewhart different. First we save their current values, then we check if `\`⟨*lang*⟩`hyphenmins` is defined. If it is not we set default values (2 and 3), otherwise the values in `\`⟨*lang*⟩`hyphenmins` will be used.

```
    \babel@savevariable\lefthyphenmin
    \babel@savevariable\righthyphenmin
    \expandafter\ifx\csname #1hyphenmins\endcsname\relax
```

```
      \lefthyphenmin\tw@\righthyphenmin\thr@@\relax
    \else
      \expandafter\expandafter\expandafter\set@hyphenmins
        \csname #1hyphenmins\endcsname\relax
    \fi
  \fi}
```

otherlanguage The otherlanguage environment can be used as an alternative to using the `\selectlanguage` declarative command. When you are typesetting a document which mixes left-to-right and right-to-left typesetting you have to use this environment in order to let things work as you expect them to.

The first thing this environment does is store the name of the language in `\languagename`; it then calls `\selectlanguage␣` to switch on everything that is needed for this language The `\ignorespaces` command is necessary to hide the environment when it is entered in horizontal mode.

```
\long\def\otherlanguage#1{%
  \def\languagename{#1}%
  \csname selectlanguage \endcsname{#1}%
  \ignorespaces
  }
```

The `\endotherlanguage` part of the environment calls `\originalTeX` to restore (most of) the settings and tries to hide itself when it is called in horizontal mode.

```
\long\def\endotherlanguage{%
  \originalTeX
  \global\@ignoretrue\ignorespaces
  }
```

otherlanguage* The otherlanguage environment is meant to be used when a large part of text from a different language needs to be typeset, but without changing the translation of words such as 'figure'.

This environment makes use of `\foreign@language`.

```
\expandafter\def\csname otherlanguage*\endcsname#1{%
  \foreign@language{#1}%
  }
```

At the end of the environment we need to switch off the extra definitions. The grouping mechanism of the environment will take care of resetting the correct hyphenation rules.

```
\expandafter\def\csname endotherlanguage*\endcsname{%
  \csname noextras\languagename\endcsname
  }
```

\foreignlanguage The `\foreignlanguage` command is another substitute for the `\selectlanguage` command. This command takes two arguments, the first argument is the name of the language to use for typesetting the text specified in the second argument.

Unlike `\selectlanguage` this command doesn't switch *everything*, it only switches the hyphenation rules and the extra definitions for the language specified. It does this within a group and assumes the `\extras`⟨*lang*⟩ command doesn't make any `\global` changes. The coding is very similar to part of `\selectlanguage`.

```
\def\foreignlanguage{\protect\csname foreignlanguage \endcsname}
\expandafter\def\csname foreignlanguage \endcsname#1#2{%
  \begingroup
```

```
    \foreign@language{#1}%
    #2%
    \csname noextras#1\endcsname
  \endgroup
  }
```

\foreign@language This macro does the work for `\foreignlanguage` and the otherlanguage* environment.

```
\def\foreign@language#1{%
%    First we need to store the name of the language and check that it
%    is a known language.
%    \begin{macrocode}
  \def\languagename{#1}%
  \expandafter\ifx\csname l@#1\endcsname\relax
    \@nolanerr{#1}%
  \else
```

If it is we can select the proper hyphenation table and switch on the extra definitions for this language.

```
    \language=\csname l@#1\endcsname\relax
    \languageshorthands{none}%
```

Then we set the left- and right hyphenmin variables.

```
    \csname extras#1\endcsname
    \expandafter\ifx\csname #1hyphenmins\endcsname\relax
      \lefthyphenmin\tw@\righthyphenmin\thr@@\relax
    \else
      \expandafter\expandafter\expandafter\set@hyphenmins
        \csname #1hyphenmins\endcsname\relax
    \fi
  \fi
  }
```

\set@hyphenmins This macro sets the values of `\lefthyphenmin` and `\righthyphenmin`. It expects two values as its argument.

```
\def\set@hyphenmins#1#2{\lefthyphenmin#1\righthyphenmin#2}
```

\LdfInit This macro is defined in two versions. The first version is to be part of the 'kernel' of babel, ie. the part that is loaded in the format; the second version is defined in `babel.def`. The version in the format just checks the category code of the ampersand and then loads `babel.def`.

```
\def\LdfInit{%
  \chardef\atcatcode=\catcode`\@
  \catcode`\@=11\relax
  \input babel.def\relax
```

The categary code of the ampersand is restored and the macro calls itself again with the new definition from `babel.def`

```
  \catcode`\@=\atcatcode \let\atcatcode\relax
  \LdfInit}
⟨/kernel⟩
```

The second version of this macro takes two arguments. The first argument is the name of the language that will be defined in the language definition file; the second argument is either a control sequence or a string from which a control sequence should be constructed. The existence of the control sequence indicates that the file has been processed before.

At the start of processing a language definition file we always check the category code of the ampersand. We make sure that it is a ‘letter’ during the processing of the file.

```
⟨*core⟩
\def\LdfInit#1#2{%
  \chardef\atcatcode=\catcode`\@
  \catcode`\@=11\relax
```

Now we check whether we should perhaps stop the processing of this file. To do this we first need to check whether the second argument that is passed to `\LdfInit` is a control sequence. We do that by looking at the first token after passing `#2` through `string`. When it is equal to `\@backslashchar` we are dealing with a control sequence which we can compare with `\@undefined`.

```
  \let\bbl@tempa\relax
  \expandafter\if\expandafter\@backslashchar
                 \expandafter\@car\string#2\@nil
    \ifx#2\@undefined
    \else
```

If so, we call `\ldf@quit` (but after the end of this `\if` construction) to set the main language, restore the category code of the @-sign and call `\endinput`.

```
      \def\bbl@tempa{\ldf@quit{#1}}
    \fi
  \else
```

When `#2` was *not* a control sequence we construct one and compare it with `\relax`.

```
    \expandafter\ifx\csname#2\endcsname\relax
    \else
      \def\bbl@tempa{\ldf@quit{#1}}
    \fi
  \fi
  \bbl@tempa
```

Finally we check `\orginalTeX`.

```
  \ifx\originalTeX\@undefined
    \let\originalTeX\@empty
  \else
    \originalTeX
  \fi}
```

`\ldf@quit` This macro interrupts the processing of a language definition file.

```
\def\ldf@quit#1{%
  \expandafter\main@language\expandafter{#1}%
  \catcode`\@=\atcatcode \let\atcatcode\relax
  \endinput
}
```

`\ldf@finish` This macro takes one argument. It is the name of the language that was defined in the language definition file.

We load the local configuration file if one is present, we set the main language (taking into account that the argument might be a control sequence that needs to be expanded) and reset the category code of the @-sign.

```
\def\ldf@finish#1{%
  \loadlocalcfg{#1}
  \expandafter\main@language\expandafter{#1}%
  \catcode`\@=\atcatcode \let\atcatcode\relax
  }
```

After the preamble of the document the commands `\LdfInit`, `\ldf@quit` and `\ldf@finish` are no longer needed. Therefore they are turned into warning messages in LaTeX.

```
\@onlypreamble\LdfInit
\@onlypreamble\ldf@quit
\@onlypreamble\ldf@finish
```

`\main@language` `\bbl@main@language` This command should be used in the various language definition files. It stores its argument in `\bbl@main@language`; to be used to switch to the correct language at the beginning of the document.

```
\def\main@language#1{%
  \def\bbl@main@language{#1}%
  \let\languagename\bbl@main@language
  \language=\csname l@\languagename\endcsname\relax
  }
```

The default is to use English as the main language.

```
\ifx\l@english\undefined
  \let\l@english\z@
\fi
\main@language{english}
```

We also have to make sure that some code gets executed at the beginning of the document.

```
\AtBeginDocument{%
  \expandafter\selectlanguage\expandafter{\bbl@main@language}}
⟨/core⟩
```

`\originalTeX` The macro`\originalTeX` should be known to TeX at this moment. As it has to be expandable we `\let` it to `\@empty` instead of `\relax`.

```
⟨*kernel⟩
\ifx\originalTeX\@undefined\let\originalTeX\@empty\fi
```

Because this part of the code can be included in a format, we make sure that the macro which initialises the save mechanism, `\babel@beginsave`, is not considered to be undefined.

```
\ifx\babel@beginsave\@undefined\let\babel@beginsave\relax\fi
```

`\@nolanerr` `\@nopatterns` The babel package will signal an error when a documents tries to select a language that hasn't been defined earlier. When a user selects a language for which no hyphenation patterns were loaded into the format he will be given a warning about that fact. We revert to the patterns for `\language`=0 in that case. In most formats that will be (US)english, but it might also be empty.

When the format knows about `\PackageError` it must be LaTeX 2_ε, so we can safely use its error handling interface. Otherwise we'll have to 'keep it simple'.

```
\ifx\PackageError\@undefined
  \def\@nolanerr#1{%
```

```
    \errhelp{Your command will be ignored, type <return> to proceed}%
    \errmessage{You haven't defined the language #1\space yet}}
  \def\@nopatterns#1{%
    \message{No hyphenation patterns were loaded for}
    \message{the language '#1'}
    \message{I will use the patterns loaded for \string\language=0
          instead}}
  \def\@activated#1{%
    \wlog{Package babel Info: Making #1 an active character}}
\else
  \newcommand*{\@nolanerr}[1]{%
    \PackageError{babel}%
                 {You haven't defined the language #1\space yet}%
        {Your command will be ignored, type <return> to proceed}}
  \newcommand*{\@nopatterns}[1]{%
    \PackageWarningNoLine{babel}%
        {No hyphenation patterns were loaded for\MessageBreak
          the language '#1'\MessageBreak
          I will use the patterns loaded for \string\language=0
          instead}}
  \newcommand*{\@activated}[1]{%
    \PackageInfo{babel}{%
      Making #1 an active character}}
\fi
```

The following code is meant to be read by iniTEX because it should instruct TEX to read hyphenation patterns. To this end the `docstrip` option `patterns` can be used to include this code in the file `hyphen.cfg`.

```
⟨*patterns⟩
```

`\process@line` Each line in the file `language.dat` is processed by `\process@line` after it is read. The first thing this macro does is to check wether the line starts with `=`. When the first token of a line is an `=`, the macro `\process@synonym` is called; otherwise the macro `\process@language` will continue.

```
\def\process@line#1#2/{%
  \ifx=#1
    \process@synonym#2/
  \else
    \process@language#1#2/%
  \fi
  }
```

`\process@synonym` This macro takes care of the lines which start with an `=`.

```
\def\process@synonym#1 /{%
  \ifnum\last@language=\m@ne
```

When no languages have been loaded yet the name following the `=` will be a synonym for hyphenation register 0.

```
    \expandafter\global
    \expandafter\chardef\csname l@#1\endcsname0\relax
    \wlog{\string\l@#1=\string\language0}
  \else
```

Otherwise the name will be a synonym for the language loaded last.

```
    \expandafter\global
    \expandafter\chardef\csname l@#1\endcsname\last@language
    \wlog{\string\l@#1=\string\language\the\last@language}
  \fi
  }
```

\process@language The macro `\process@language` is used to process a non-empty line from the 'configuration file'. It has three arguments, each delimited by white space. The third argument is optional, therfore a `/` character is expected to delimit the last argument. The first argument is the 'name' of a language, the second is the name of the file that contains the patterns. The optional third argument is the name of a file containing hyphenation exceptions.

The first thing to do is call `\addlanguage` to allocate a pattern register and to make that register 'active'.

```
\def\process@language#1 #2 #3/{%
    \expandafter\addlanguage\csname l@#1\endcsname
    \expandafter\language\csname l@#1\endcsname
```

Then the 'name' of the language that will be loaded now is added to the token register `\toks8`. and finally the pattern file is read.

```
    \global\toks8\expandafter{\the\toks8#1, }%
```

Some pattern files contain assignments to `\lefthyphenmin` and `\righthyphenmin`. TEX does not keep track of these assignments. Therefore we try to detect such assignments and store them in the \⟨*lang*⟩`hyphenmins` macro. When no assignments were made we provide a default setting.

```
    \lefthyphenmin\m@ne
    \input #2\relax
    \ifnum\lefthyphenmin=\m@ne
      \lefthyphenmin\tw@
      \righthyphenmin\thr@@
    \fi
```

When the hyphenation patterns have been processed we need to see if a file with hyphenation exceptions needs to be read. This is the case when the third argument is not empty and when it does not contain a space token.

```
    \def\bbl@tempa{#3}
    \ifx\bbl@tempa\@empty
    \else
      \ifx\bbl@tempa\space
      \else
        \input #3\relax
      \fi
    \fi
```

Finally we store the settings of `\lefthyphenmin` and `\righthyphenmin`.

```
    \expandafter\edef\csname #1hyphenmins\endcsname{%
      \the\lefthyphenmin\the\righthyphenmin}}
```

\readconfigfile The configuration file can now be opened for reading.

```
\openin1 = language.dat
```

See if the file exists, if not, use the default hyphenation file `hyphen.tex`. The user will be informed about this.

```
\ifeof1
  \message{I couldn't find the file language.dat,\space
           I will try the file hyphen.tex}
  \input hyphen.tex\relax
\else
```

Pattern registers are allocated using count register `\last@language`. Its initial value is 0. The definition of the macro `\newlanguage` is such that it first increments the count register and then defines the language. In order to have the first patterns loaded in pattern register number 0 we initialize `\last@language` with the value −1.

```
  \last@language\m@ne
```

We now read lines from the file until the end is found

```
  \loop
```

While reading from the input it is useful to switch off recognition of the end-of-line character. This saves us stripping off spaces from the contents of the controlsequence.

```
    \endlinechar\m@ne
    \read1 to \bbl@line
    \endlinechar`\^^M
```

Empty lines are skipped.

```
    \ifx\bbl@line\@empty
    \else
```

Now we add a space and a `/` character to the end of `\bbl@line`. This is needed to be able to recognize the third, optional, argument of `\process@language` later on.

```
      \edef\bbl@line{\bbl@line\space/}
      \expandafter\process@line\bbl@line
    \fi
```

Check for the end of the file. To avoid a new `if` control sequence we create the necessary `\iftrue` or `\iffalse` with the help of `\csname`. But there is one complication with this approach: when skipping the `loop...repeat` TEX has to read `\if`/`\fi` pairs. So we have to insert a 'dummy' `\iftrue`.

```
    \iftrue \csname fi\endcsname
    \csname if\ifeof1 false\else true\fi\endcsname
  \repeat
```

Reactivate the default patterns,

```
  \language=0
\fi
```

and close the configuration file.

```
\closein1
```

Also remove some macros from memory

```
\let\process@language\@undefined
\let\process@synonym\@undefined
\let\process@line\@undefined
```

```
\let\bbl@tempa\@undefined
\let\bbl@tempb\@undefined
\let\bbl@eq@\@undefined
\let\bbl@line\@undefined
```

We add a message about the fact that babel is loaded in the format and with which language patterns to the `\everyjob` register.

```
\ifx\addto@hook\@undefined
\else
  \expandafter\addto@hook\expandafter\everyjob\expandafter{%
    \expandafter\typeout\expandafter{\the\toks8 loaded.}}
\fi
```

Here the code for iniTEX ends.

```
⟨/patterns⟩
⟨/kernel⟩
```

10.2 Support for active characters

`\bbl@add@special` The macro `\bbl@add@special` is used to add a new character (or single character control sequence) to the macro `\dospecials` (and `\@sanitize` if LATEX is used).

To keep all changes local, we begin a new group. Then we redefine the macros `\do` and `\@makeother` to add themselves and the given character without expansion.

```
⟨*core | shorthands⟩
\def\bbl@add@special#1{\begingroup
    \def\do{\noexpand\do\noexpand}%
    \def\@makeother{\noexpand\@makeother\noexpand}%
```

To add the character to the macros, we expand the original macros with the additional character inside the redefinition of the macros. Because `\@sanitize` can be undefined, we put the definition inside a conditional.

```
    \edef\x{\endgroup
      \def\noexpand\dospecials{\dospecials\do#1}%
      \expandafter\ifx\csname @sanitize\endcsname\relax \else
        \def\noexpand\@sanitize{\@sanitize\@makeother#1}%
      \fi}%
```

The macro `\x` contains at this moment the following:
`\endgroup\def\dospecials{`*old contents* `\do`⟨*char*⟩`}`.
If `\@sanitize` is defined, it contains an additional definition of this macro. The last thing we have to do, is the expansion of `\x`. Then `\endgroup` is executed, which restores the old meaning of `\x`, `\do` and `\@makeother`. After the group is closed, the new definition of `\dospecials` (and `\@sanitize`) is assigned.

```
  \x}
```

`\bbl@remove@special` The companion of the former macro is `\bbl@remove@special`. It is used to remove a character from the set macros `\dospecials` and `\@sanitize`.

To keep all changes local, we begin a new group. Then we define a help macro `\x`, which expands to empty if the characters match, otherwise it expands to its nonexpandable input. Because TEX inserts a `\relax`, if the corresponding `\else` or `\fi` is scanned before the comparison is evaluated, we provide a 'stop sign' which should expand to nothing.

```
\def\bbl@remove@special#1{\begingroup
    \def\x##1##2{\ifnum`#1=`##2\noexpand\@empty
                 \else\noexpand##1\noexpand##2\fi}%
```

With the help of this macro we define `\do` and `\make@other`.

```
    \def\do{\x\do}%
    \def\@makeother{\x\@makeother}%
```

The rest of the work is similar to `\bbl@add@special`.

```
    \edef\x{\endgroup
      \def\noexpand\dospecials{\dospecials}%
      \expandafter\ifx\csname @sanitize\endcsname\relax \else
        \def\noexpand\@sanitize{\@sanitize}%
      \fi}%
  \x}
```

10.3 Shorthands

`\initiate@active@char` A language definition file can call this macro to make a character active. This macro takes one argument, the character that is to be made active. When the character was already active this macro does nothing. Otherwise, this macro defines the control sequence `\normal@char`⟨*char*⟩ to expand to the character in its 'normal state' and it defines the active character to expand to `\normal@char`⟨*char*⟩ by default (⟨*char*⟩ being the character to be made active). Later its definition can be be changed to expand to `\active@char`⟨*char*⟩ by calling `\bbl@activate{`⟨*char*⟩`}`.

For example, to make the double quote character active one could have the following line in a language definition file:

```
\initiate@active@char{"}
```

`\bbl@afterelse`
`\bbl@afterfi` Because the code that is used in the handling of active characters may need to look ahead, we take extra care to 'throw' it over the `\else` and `\fi` parts of an `\if`-statement[5].

```
\def\bbl@afterelse#1\else#2\fi{\fi#1}
\def\bbl@afterfi#1\fi{\fi#1}
```

`\peek@token` In order to prevent error messages when a shorthand, which normally takes an argument sees a `\par`, or `}`, or similar tokens we need to be able to 'peek' at what is coming up next in the input stream. Depending on the category code of the token that is seen we need to either continue the code for the active character, or insert the non-active version of that character in the output. The macro `\peek@token` therefore takes two arguments, with which it constructs the control sequence to expand next. It `\let`'s `\bbl@nexta` and `\bbl@nextb` to the two possible macro's. This is necessary for `\bbl@test@token` to take the right decision.

```
\def\peek@token#1#2{%
  \expandafter\let\expandafter\bbl@nexta\csname #1\string#2\endcsname
  \expandafter\let\expandafter\bbl@nextb
    \csname system@active\string#2\endcsname
  \futurelet\bbl@token\bbl@test@token}
```

[5]This code is based on code presented in TUGboat vol. 12, no2, June 1991 in "An expansion Power Lemma" by Sonja Maus.

`bbl@test@token` When the result of peeking at the next token has yeilded a token with category 'letter', 'other' or 'active' it is safe to proceed with evaluating the code for the shorthand. When a token is found with any other category code proceeding is unsafe and therefore the orginal shorthand character is inserted in the output. The macro that calls `\bbl@test@token` needs to setup `\bbl@nexta` and `\bbl@nextb` in order to achieve this.

```
\def\bbl@test@token{%
  \let\bbl@next\bbl@nexta
  \ifcat\noexpand\bbl@token a%
  \else
    \ifcat\noexpand\bbl@token=%
    \else
      \ifcat\noexpand\bbl@token\noexpand\bbl@next
      \else
        \let\bbl@next\bbl@nextb
      \fi
    \fi
  \fi
  \bbl@next}
```

Note that the definition of `\initiate@active@char` needs an active character, for this the ~ is used. Some of the changes we need, do not have to become available later on, so we do it inside a group.

```
\begingroup
  \catcode`\~\active
  \def\x{\endgroup
    \def\initiate@active@char##1{%
```

If the character is already active we provide the default expansion under this shorthand mechanism.

```
      \ifcat\noexpand##1\noexpand~\relax
        \expandafter\edef\csname normal@char\string##1\endcsname{##1}%
        \expandafter\gdef
          \expandafter##1%
          \expandafter{%
          \expandafter\active@prefix\expandafter##1%
          \csname normal@char\string##1\endcsname}
      \else
```

Otherwise we write a message in the transcript file,

```
        \@activated{##1}%
```

and define `\normal@char`⟨*char*⟩ to expand to the character in its default state.

```
        \@namedef{normal@char\string##1}{##1}%
```

If we are making the right quote active we need to change `\pr@m@s` as well.

```
        \ifx##1'%
          \let\pr@m@s\bbl@pr@m@s
        \fi
```

To prevent problems with the loading of other packages after babel we reset the catcode of the character at the end of the package.

```
        \ifx\KeepShorthandsActive\@undefined
          \edef\bbl@tempa{\catcode`\noexpand##1\the\catcode`##1}
          \expandafter\AtEndOfPackage\expandafter{\bbl@tempa}%
        \fi
```

Now we set the lowercase code of the ~ equal to that of the character to be made active and execute the rest of the code inside a `\lowercase` 'environment'.

```
        \@tempcnta=\lccode`\~
        \lccode`~=`##1%
        \lowercase{%
```

Make the character active and add it to `\dospecials` and `\@sanitize`.

```
          \catcode`~\active
          \expandafter\bbl@add@special
            \csname \string##1\endcsname
```

Also re-activate it again at `\begin{document}`.

```
          \AtBeginDocument{\catcode`##1\active}%
```

Define the character to expand to

$$\texttt{\textbackslash active@prefix}\ \langle \mathit{char}\rangle\ \texttt{\textbackslash normal@char}\langle \mathit{char}\rangle$$

(where `\active@char`⟨*char*⟩ is *one* control sequence!).

```
          \expandafter\gdef
            \expandafter~%
            \expandafter{%
            \expandafter\active@prefix\expandafter##1%
            \csname normal@char\string##1\endcsname}}%
        \lccode`\~\@tempcnta
      \fi
```

We define the first level expansion of `\active@char`⟨*char*⟩ to check the status of the `@safe@actives` flag. If it is set to true we expand to the 'normal' version of this character, otherwise we call `\@active@char`⟨*char*⟩.

```
      \@namedef{active@char\string##1}{%
        \if@safe@actives
          \bbl@afterelse\csname normal@char\string##1\endcsname
        \else
          \bbl@afterfi\csname user@active\string##1\endcsname
        \fi}%
```

The next level of the code checks whether a user has defined a shorthand for himself with this character. First we check for a single character shorthand. If that doesn't exist we check for a shorthand with an argument.

```
      \@namedef{user@active\string##1}{%
        \expandafter\ifx
        \csname \user@group @sh@\string##1@\endcsname
        \relax
          \bbl@afterelse\csname @sh@\string##1@sel\endcsname
        {user@active@arg\string##1}{language@active\string##1}%
        \else
          \bbl@afterfi\csname \user@group @sh@\string##1@\endcsname
        \fi}%
```

When there is also no user-level shorthand with an argument we will check whether there is a language defined shorthand for this active character. Before the next token is absorbed as argument we need to make sure that this is safe. Therefore `\peek@token` is called to decide that.

```
      \@namedef{user@active@arg\string##1}{%
        \peek@token{@user@active@arg}{##1}}
      \long\@namedef{@user@active@arg\string##1}####1{%
        \expandafter\ifx
        \csname \user@group @sh@\string##1\string####1@\endcsname
        \relax
          \bbl@afterelse
          \csname language@active\string##1\endcsname####1%
        \else
          \bbl@afterfi
          \csname \user@group @sh@\string##1\string####1@%
          \endcsname
        \fi}%
```

Like the shorthands that can be defined by the user, a language definition file can also define shorthands with and without an argument, so we need two more macros to check if they exist.

```
      \@namedef{language@active\string##1}{%
        \expandafter\ifx
        \csname \language@group @sh@\string##1@\endcsname
        \relax
          \bbl@afterelse\csname @sh@\string##1@sel\endcsname
          {language@active@arg\string##1}{system@active\string##1}%
        \else
          \bbl@afterfi
          \csname \language@group @sh@\string##1@\endcsname
        \fi}%
```

```
      \@namedef{language@active@arg\string##1}{%
        \peek@token{@language@active@arg}{##1}}
      \long\@namedef{@language@active@arg\string##1}####1{%
        \expandafter\ifx
        \csname \language@group @sh@\string##1\string####1@\endcsname
        \relax
          \bbl@afterelse
          \csname system@active\string##1\endcsname####1%
        \else
          \bbl@afterfi
          \csname \language@group @sh@\string##1\string####1@%
          \endcsname
        \fi}%
```

And the same goes for the system level.

```
      \@namedef{system@active\string##1}{%
        \expandafter\ifx
        \csname \system@group @sh@\string##1@\endcsname
        \relax
          \bbl@afterelse\csname @sh@\string##1@sel\endcsname
          {system@active@arg\string##1}{normal@char\string##1}%
        \else
          \bbl@afterfi\csname \system@group @sh@\string##1@\endcsname
        \fi}%
```

When no shorthands were found the 'normal' version of the active character is inserted.

```
      \@namedef{system@active@arg\string##1}{%
        \peek@token{@system@active@arg}{##1}}
      \long\@namedef{@system@active@arg\string##1}####1{%
        \expandafter\ifx
        \csname \system@group @sh@\string##1\string####1@\endcsname
        \relax
          \bbl@afterelse\csname normal@char\string##1\endcsname####1%
        \else
          \bbl@afterfi
          \csname \system@group @sh@\string##1\string####1@\endcsname
        \fi}%
      }%
    }\x
```

`\active@prefix` The command `\active@prefix` which is used in the expansion of active characters has a function similar to `\OT1-cmd` in that it `\protect`s the active character whenever `\protect` is *not* `\@typeset@protect`.

```
\def\active@prefix#1{%
  \ifx\protect\@typeset@protect
  \else
    \bbl@afterfi\protect#1\@gobble
  \fi}
```

`\if@safe@actives` In some circumstances it is necessary to be able to change the expansion of an active character on the fly. For this purpose the switch `@safe@actives` is available. This setting of this switch should be checked in the first level expansion of `\active@char`⟨*char*⟩.

```
\newif\if@safe@actives
\@safe@activesfalse
```

`\bbl@activate` This macro takes one argument, like `\initiate@active@char`. The macro is used to change the definition of an active character to expand to `\active@char`⟨*char*⟩ instead of `\normal@char`⟨*char*⟩.

```
\def\bbl@activate#1{%
  \expandafter\def
  \expandafter#1\expandafter{%
    \expandafter\active@prefix
    \expandafter#1\csname active@char\string#1\endcsname}%
}
```

`\bbl@deactivate` This macro takes one argument, like `\bbl@ctivate`. The macro doesn't really make a character non-active; it changes its definition to expand to `\normal@char`⟨*char*⟩.

```
\def\bbl@deactivate#1{%
  \expandafter\def
  \expandafter#1\expandafter{%
    \expandafter\active@prefix
    \expandafter#1\csname normal@char\string#1\endcsname}%
}
```

`\bbl@firstcs` `\bbl@scndcs` These macros have two arguments. They use one of their arguments to build a control sequence from.

```
\def\bbl@firstcs#1#2{\csname#1\endcsname}
\def\bbl@scndcs#1#2{\csname#2\endcsname}
```

`\declare@shorthand` The command `\declare@shorthand` is used to declare a shorthand on a certain level. It takes three arguments:

1. a name for the collection of shorthands, i.e. 'system', or 'dutch';
2. the character (sequence) that makes up the shorthand, i.e. `~` or `"a`;
3. the code to be executed when the shorthand is encountered.

```
\def\declare@shorthand#1#2{\@decl@short{#1}#2\@nil}
\def\@decl@short#1#2#3\@nil#4{%
  \def\bbl@tempa{#3}%
  \ifx\bbl@tempa\@empty
    \expandafter\let\csname @sh@\string#2@sel\endcsname\bbl@scndcs
  \else
    \expandafter\let\csname @sh@\string#2@sel\endcsname\bbl@firstcs
  \fi
  \@namedef{#1@sh@\string#2\string#3@}{#4}}
```

`\textormath` Some of the shorthands that will be declared by the language definition files have to be useable in both text and mathmode. To achieve this the helper macro `\textormath` is provided.

```
\def\textormath#1#2{%
  \ifmmode
    \bbl@afterelse#2%
  \else
    \bbl@afterfi#1%
  \fi}
```

`\user@group` `\language@group` `\system@group` The current concept of 'shorthands' supports three levels or groups of shorthands. For each level the name of the level or group is stored in a macro. The default is to have no user group; use language group 'english' and have a system group called 'system'.

```
\def\user@group{}
\def\language@group{english}
\def\system@group{system}
```

`\useshorthands` This is the user level command to tell LaTeX that user level shorthands will be used in the document. It takes one argument, the character that starts a shorthand.

```
\def\useshorthands#1{%
  \def\user@group{user}%
  \initiate@active@char{#1}%
  \bbl@activate{#1}}
```

`\defineshorthand` Currently we only support one group of user level shorthands, called 'user'.

```
\def\defineshorthand{\declare@shorthand{user}}
```

`\languageshorthands` A user level command to change the language from which shorthands are used.

```
\def\languageshorthands#1{\def\language@group{#1}}
```

`\aliasshorthand` Because we deal with active characters here we need to use the `\lccode` trick. Therefore we save the current `\lccode` of the `~`-character and restore it later. Then we make the new character active and `\let` it be equal to the original.

```
\def\aliasshorthand#1#2{%
  \@tempcnta\lccode`\~
  \lccode`~=`#2%
  \lowercase{\catcode`~\active\let~#1\catcode`#112\relax}%
  \lccode`\~\@tempcnta}
```

To prevent problems with constructs such as `\char"01A` when the double quote is made active, we define a shorthand on system level.

```
\declare@shorthand{system}{"}{\csname normal@char\string"\endcsname}
```

When the right quote is made active we need to take care of handling it correctly in mathmode. Therefore we define a shorthand at system level to make it expand to a non-active right quote in textmode, but expand to its original definition in mathmode. (Note that the right quote is 'active' in mathmode because of its mathcode.)

```
\declare@shorthand{system}{'}{%
  \textormath{\csname normal@char\string'\endcsname}%
             {\sp\bgroup\prim@s}}
```

When the left quote is made active we need to take care of handling it correctly when it is followed by for instance an open brace token. Therefore we define a shorthand at system level to make it expand to a non-active left quote.

```
\declare@shorthand{system}{`}{\csname normal@char\string`\endcsname}
```

`\bbl@pr@m@s` One of the internal macros that are involved in substituting `\prime` for each right quote in mathmode is `\pr@m@s`. This checks if the next character is a right quote. When the right quote is active, the definition of this macro needs to be adapted to look for an active right quote.

```
\begingroup
  \catcode`\'\active\let'\relax
  \def\x{\endgroup
    \def\bbl@pr@m@s{%
      \ifx'\@let@token
        \expandafter\pr@@@s
      \else
        \ifx^\@let@token
          \expandafter\expandafter\expandafter\pr@@@t
        \else
          \egroup
        \fi
      \fi}%
    }
\x
⟨/core | shorthands⟩
```

Normally the `~` is active and expands to `\penalty\@M\␣`. When it is written to the `.aux` file it is written expanded. To prevent that and to be able to use the character `~` as a start character for a shorthand, it is redefined here as a one character shorthand on system level.

```
% \changes{babel~3.5f}{1996/04/02}{No need to reset the category code
%    ofg the tilde as \cs{initiate@active@char now cooreclty deals
%    with active characters}
⟨*core⟩
```

```
\initiate@active@char{~}
\declare@shorthand{system}{~}{\penalty\@M\ }
\bbl@activate{~}
```

`\OT1dqpos` `\T1dqpos` The position of the double quote character is different for the OT1 and T1 encodings. It will later be selected using the `\f@encoding` macro. Therefore we define two macros here to store the position of the character in these encodings.

```
\expandafter\def\csname OT1dqpos\endcsname{127}
\expandafter\def\csname T1dqpos\endcsname{4}
```

When the macor `\f@encoding` is undefined (as it is in plain TEX) we define it here to expand to `OT1`

```
\ifx\f@encoding\@undefined
  \def\f@encoding{OT1}
\fi
```

10.4 Support for saving macro definitions

To save the meaning of control sequences using `\babel@save`, we use temporary control sequences. To save hash table entries for these control sequences, we don't use the name of the control sequence to be saved to construct the temporary name. Instead we simply use the value of a counter, which is reset to zero each time we begin to save new values. This works well because we release the saved meanings before we begin to save a new set of control sequence meanings (see `\selectlanguage` and `\originalTeX`).

`\babel@savecnt` `\babel@beginsave` The initialization of a new save cycle: reset the counter to zero.

```
\def\babel@beginsave{\babel@savecnt\z@}
```

Before it's forgotten, allocate the counter and initialize all.

```
\newcount\babel@savecnt
\babel@beginsave
```

`\babel@save` The macro `\babel@save`⟨*csname*⟩ saves the current meaning of the control sequence ⟨*csname*⟩ to `\originalTeX`[6]. To do this, we let the current meaning to a temporary control sequence, the restore commands are appended to `\originalTeX` and the counter is incremented.

```
\def\babel@save#1{%
  \expandafter\let\csname babel@\number\babel@savecnt\endcsname #1\relax
  \begingroup
    \toks@\expandafter{\originalTeX \let#1=}%
    \edef\x{\endgroup
      \def\noexpand\originalTeX{\the\toks@ \expandafter\noexpand
         \csname babel@\number\babel@savecnt\endcsname\relax}}%
  \x
  \advance\babel@savecnt\@ne}
```

`\babel@savevariable` The macro `\babel@savevariable`⟨*variable*⟩ saves the value of the variable. ⟨*variable*⟩ can be anything allowed after the `\the` primitive.

```
\def\babel@savevariable#1{\begingroup
    \toks@\expandafter{\originalTeX #1=}%
```

[6] `\originalTeX` has to be expandable, i. e. you shouldn't let it to `\relax`.

```
    \edef\x{\endgroup
      \def\noexpand\originalTeX{\the\toks@ \the#1\relax}}%
  \x}
```

`\bbl@frenchspacing` `\bbl@nonfrenchspacing` Some languages need to have `\frenchspacing` in effect. Others don't want that. The command `\bbl@frenchspacing` switches it on when it isn't already in effect and `\bbl@nonfrenchspacing` switches it off if necessary.

```
\def\bbl@frenchspacing{%
  \ifnum\the\sfcode`\.=\@m
    \let\bbl@nonfrenchspacing\relax
  \else
    \frenchspacing
    \let\bbl@nonfrenchspacing\nonfrenchspacing
  \fi}
\let\bbl@nonfrenchspacing\nonfrenchspacing
```

10.5 Support for extending macros

`\addto` For each language four control sequences have to be defined that control the language-specific definitions. To be able to add something to these macro once they have been defined the macro `\addto` is introduced. It takes two arguments, a ⟨*control sequence*⟩ and TEX-code to be added to the ⟨*control sequence*⟩.

If the ⟨*control sequence*⟩ has not been defined before it is defined now.

```
\def\addto#1#2{%
  \ifx#1\@undefined
    \def#1{#2}
  \else
```

The control sequence could also expand to `\relax`, in which case a circular definition results. The net result is a stack overflow.

```
    \ifx#1\relax
      \def#1{#2}
    \else
```

Otherwise the replacement text for the ⟨*control sequence*⟩ is expanded and stored in a token register, together with the TEX-code to be added. Finally the ⟨*control sequence*⟩ is *re*defined, using the contents of the token register.

```
      {\toks@\expandafter{#1#2}%
        \xdef#1{\the\toks@}}%
    \fi
  \fi
}
```

10.6 Macros common to a number of languages

`\allowhyphens` This macro makes hyphenation possible. Basically its definition is nothing more than `\nobreak \hskip 0pt plus 0pt`[7].

```
\def\allowhyphens{\penalty\@M \hskip\z@skip}
```

[7]TEX begins and ends a word for hyphenation at a glue node. The penalty prevents a linebreak at this glue node.

`\set@low@box` The following macro is used to lower quotes to the same level as the comma. It prepares its argument in box register 0.

```
\def\set@low@box#1{\setbox\tw@\hbox{,}\setbox\z@\hbox{#1}%
    \dimen\z@\ht\z@ \advance\dimen\z@ -\ht\tw@%
    \setbox\z@\hbox{\lower\dimen\z@ \box\z@}\ht\z@\ht\tw@ \dp\z@\dp\tw@}
```

`\save@sf@q` The macro `\save@sf@q` is used to save and reset the current space factor.

```
\def\save@sf@q#1{{\ifhmode
    \edef\@SF{\spacefactor\the\spacefactor}\else
    \let\@SF\@empty \fi \leavevmode #1\@SF}}
```

`\bbl@disc` For some languages the macro `\bbl@disc` is used to ease the insertion of discretionaries for letters that behave ‘abnormally’ at a breakpoint.

```
\def\bbl@disc#1#2{%
  \penalty\@M\discretionary{#2-}{}{#1}\allowhyphens}
```

10.7 Making glyphs available

The file `babel.dtx`[8] makes a number of glyphs available that either do not exist in the `OT1` encoding and have to be ‘faked’, or that are not accessible through `T1enc.def`.

10.8 Quotation marks

`\quotedblbase` In the `T1` encoding the opening double quote at the baseline is available as a separate character, accessible via `\quotedblbase`. In the `OT1` encoding it is not available, therefore we make it available by lowering the normal open quote character to the baseline.

```
\ProvideTextCommand{\quotedblbase}{OT1}{%
  \save@sf@q{\set@low@box{\textquotedblright\/}%
    \box\z@\kern-.04em\allowhyphens}}
```

Make sure that when an encoding other then `OT1` ot `T1` is used this glyph can still be typeset.

```
\ProvideTextCommandDefault{\quotedblbase}{%
  \UseTextSymbol{OT1}{\quotedblbase}}
```

`\quotesinglbase` We also need the single quote character at the baseline.

```
\ProvideTextCommand{\quotesinglbase}{OT1}{%
  \save@sf@q{\set@low@box{\textquoteright\/}%
    \box\z@\kern-.04em\allowhyphens}}
```

Make sure that when an encoding other then `OT1` ot `T1` is used this glyph can still be typeset.

```
\ProvideTextCommandDefault{\quotesinglbase}{%
  \UseTextSymbol{OT1}{\quotesinglbase}}
```

[8]The file described in this section has version number v3.6j, and was last revised on 1998/03/24.

\guillemotleft
\guillemotright

The guillemot characters are not available in OT1 encoding. They are faked.

```
\ProvideTextCommand{\guillemotleft}{OT1}{%
  \ifmmode
    \ll
  \else
    \save@sf@q{\penalty\@M
      \raise.2ex\hbox{$\scriptscriptstyle\ll$}\allowhyphens}%
  \fi}
\ProvideTextCommand{\guillemotright}{OT1}{%
  \ifmmode
    \gg
  \else
    \save@sf@q{\penalty\@M
      \raise.2ex\hbox{$\scriptscriptstyle\gg$}\allowhyphens}%
  \fi}
```

Make sure that when an encoding other then OT1 ot T1 is used these glyphs can still be typeset.

```
\ProvideTextCommandDefault{\guillemotleft}{%
  \UseTextSymbol{OT1}{\guillemotleft}}
\ProvideTextCommandDefault{\guillemotright}{%
  \UseTextSymbol{OT1}{\guillemotright}}
```

\guilsinglleft
\guilsinglright

The single guillemots are not available in OT1 encoding. They are faked.

```
\ProvideTextCommand{\guilsinglleft}{OT1}{%
  \ifmmode
    <%
  \else
    \save@sf@q{\penalty\@M
      \raise.2ex\hbox{$\scriptscriptstyle<$}\allowhyphens}%
  \fi}
\ProvideTextCommand{\guilsinglright}{OT1}{%
  \ifmmode
    >%
  \else
    \save@sf@q{\penalty\@M
      \raise.2ex\hbox{$\scriptscriptstyle>$}\allowhyphens}%
  \fi}
```

Make sure that when an encoding other then OT1 ot T1 is used these glyphs can still be typeset.

```
\ProvideTextCommandDefault{\guilsinglleft}{%
  \UseTextSymbol{OT1}{\guilsinglleft}}
\ProvideTextCommandDefault{\guilsinglright}{%
  \UseTextSymbol{OT1}{\guilsinglright}}
```

10.9 Letters

\ij
\IJ

The dutch language uses the letter 'ij'. It is available in T1 encoded fonts, but not in the OT1 encoded fonts. Therefore we fake it for the OT1 encoding.

```
\DeclareTextCommand{\ij}{OT1}{%
  \allowhyphens i\kern-0.02em j\allowhyphens}
\DeclareTextCommand{\IJ}{OT1}{%
  \allowhyphens I\kern-0.02em J\allowhyphens}
```

```
\DeclareTextCommand{\ij}{T1}{\char188}
\DeclareTextCommand{\IJ}{T1}{\char156}
```

Make sure that when an encoding other then OT1 or T1 is used these glyphs can still be typeset.

```
\ProvideTextCommandDefault{\ij}{%
  \UseTextSymbol{OT1}{\ij}}
\ProvideTextCommandDefault{\IJ}{%
  \UseTextSymbol{OT1}{\IJ}}
```

`\dj` `\DJ` The croatian language needs the letters `\dj` and `\DJ`; they are available in the T1 encoding, but not in the OT1 encoding by default.

Some code to construct these glyphs for the OT1 encoding was made available to me by Stipcevic Mario, (stipcevic@olimp.irb.hr).

```
\def\crrtic@{\hrule height0.1ex width0.3em}
\def\crttic@{\hrule height0.1ex width0.33em}
%
\def\ddj@{%
  \setbox0\hbox{d}\dimen@=\ht0
  \advance\dimen@1ex
  \dimen@.45\dimen@
  \dimen@ii\expandafter\rem@pt\the\fontdimen\@ne\font\dimen@
  \advance\dimen@ii.5ex
  \leavevmode\rlap{\raise\dimen@\hbox{\kern\dimen@ii\vbox{\crrtic@}}}}
\def\DDJ@{%
  \setbox0\hbox{D}\dimen@=.55\ht0
  \dimen@ii\expandafter\rem@pt\the\fontdimen\@ne\font\dimen@
  \advance\dimen@ii.15ex %          correction for the dash position
  \advance\dimen@ii-.15\fontdimen7\font %     correction for cmtt font
  \dimen\thr@@\expandafter\rem@pt\the\fontdimen7\font\dimen@
  \leavevmode\rlap{\raise\dimen@\hbox{\kern\dimen@ii\vbox{\crttic@}}}}
%
\DeclareTextCommand{\dj}{OT1}{\ddj@ d}
\DeclareTextCommand{\DJ}{OT1}{\DDJ@ D}
```

Make sure that when an encoding other then OT1 or T1 is used these glyphs can still be typeset.

```
\ProvideTextCommandDefault{\dj}{%
  \UseTextSymbol{OT1}{\dj}}
\ProvideTextCommandDefault{\DJ}{%
  \UseTextSymbol{OT1}{\DJ}}
```

10.10 Shorthands for quotation marks

Shorthands are provided for a number of different quotation marks, which make them useable both outside and inside mathmode.

`\glq` `\grq` The 'german' single quotes.

```
\DeclareRobustCommand{\glq}{%
  \textormath{\quotesinglbase}{\mbox{\quotesinglbase}}}
\DeclareRobustCommand{\grq}{%
  \textormath{\kern-.0125em\textquoteleft\kern.07em}%
             {\mbox{\textquoteleft}}}
```

\glqq \grqq The 'german' double quotes.

```
\DeclareRobustCommand{\glqq}{%
  \textormath{\quotedblbase}{\mbox{\quotedblbase}}}
\DeclareRobustCommand{\grqq}{%
  \textormath{\save@sf@q{\kern-.07em\textquotedblleft\kern.07em}}%
             {\mbox{\textquotedblleft}}%
  }
```

\flq \frq The 'french' single guillemets.

```
\DeclareRobustCommand{\flq}{%
  \textormath{\guilsinglleft}{\mbox{\guilsinglleft}}}
\DeclareRobustCommand{\frq}{%
  \textormath{\guilsinglright}{\mbox{\guilsinglright}}}
```

\flqq \frqq The 'french' double quillemets.

```
\DeclareRobustCommand{\flqq}{%
  \textormath{\guillemotleft}{\mbox{\guillemotleft}}}
\DeclareRobustCommand{\frqq}{%
  \textormath{\guillemotright}{\mbox{\guillemotright}}}
```

10.11 Umlauts and trema's

The command `\"` needs to have a different effect for different languages. For German for instance, the 'umlaut' should be positioned lower than the default position for placing it over the letters a, o, u, A, O and U. When placed over an e, i, E or I it can retain its normal position. For Dutch the same glyph is always placed in the lower position.

\umlauthigh \umlautlow To be able to provide both positions of `\"` we provide two commands to switch the positioning, the default will be `\umlauthigh` (the normal positioning).

```
\def\umlauthigh{%
  \def\bbl@umlauta##1{{%
      \expandafter\accent\csname\f@encoding dqpos\endcsname
      ##1\allowhyphens}}%
  \let\bbl@umlaute\bbl@umlauta}
\def\umlautlow{%
  \def\bbl@umlauta{\protect\lower@umlaut}}
\def\umlautelow{%
  \def\bbl@umlaute{\protect\lower@umlaut}}
\umlauthigh
```

\lower@umlaut The command `\lower@umlaut` is used to position the `\"` closer the the letter.

We want the umlaut character lowered, nearer to the letter. To do this we need an extra ⟨*dimen*⟩ register.

```
\expandafter\ifx\csname U@D\endcsname\relax
  \csname newdimen\endcsname\U@D
\fi
```

The following code fools TeX's `make_accent` procedure about the current x-height of the font to force another placement of the umlaut character.

```
\def\lower@umlaut#1{%
```

First we have to save the current x-height of the font, because we'll change this font dimension and this is always done globally.

```
  {\U@D 1ex%
```

Then we compute the new x-height in such a way that the umlaut character is lowered to the base character. The value of `.45ex` depends on the METAFONT parameters with which the fonts were built. (Just try out, which value will look best.)

```
  {\setbox\z@\hbox{%
      \expandafter\char\csname\f@encoding dqpos\endcsname}%
    \dimen@ -.45ex\advance\dimen@\ht\z@
```

If the new x-height is too low, it is not changed.

```
  \ifdim 1ex<\dimen@ \fontdimen5\font\dimen@ \fi}%
```

Finally we call the `\accent` primitive, reset the old x-height and insert the base character in the argument.

```
  \expandafter\accent\csname\f@encoding dqpos\endcsname
  \fontdimen5\font\U@D #1}}
```

For all vowels we declare `\"` to be a composite command which uses `\bbl@umlauta` or `\bbl@umlaute` to position the umlaut character. We need to be sure that these definitions override the ones that are provided when the package fontenc with option OT1 is used. Therefore these declarations are postponed until the beginning of the document.

```
\AtBeginDocument{%
  \DeclareTextCompositeCommand{\"}{OT1}{a}{\bbl@umlauta{a}}%
  \DeclareTextCompositeCommand{\"}{OT1}{e}{\bbl@umlaute{e}}%
  \DeclareTextCompositeCommand{\"}{OT1}{i}{\bbl@umlaute{\i}}%
  \DeclareTextCompositeCommand{\"}{OT1}{\i}{\bbl@umlaute{\i}}%
  \DeclareTextCompositeCommand{\"}{OT1}{o}{\bbl@umlauta{o}}%
  \DeclareTextCompositeCommand{\"}{OT1}{u}{\bbl@umlauta{u}}%
  \DeclareTextCompositeCommand{\"}{OT1}{A}{\bbl@umlauta{A}}%
  \DeclareTextCompositeCommand{\"}{OT1}{E}{\bbl@umlaute{E}}%
  \DeclareTextCompositeCommand{\"}{OT1}{I}{\bbl@umlaute{I}}%
  \DeclareTextCompositeCommand{\"}{OT1}{O}{\bbl@umlauta{O}}%
  \DeclareTextCompositeCommand{\"}{OT1}{U}{\bbl@umlauta{U}}%
}
```

10.12 The redefinition of the style commands

The rest of the code in this file can only be processed by LaTeX, so we check the current format. If it is plain TeX, processing should stop here. But, because of the need to limit the scope of the definition of `\format`, a macro that is used locally in the following `\if` statement, this comparison is done inside a group. To prevent TeX from complaining about an unclosed group, the processing of the command `\endinput` is deferred until after the group is closed. This is accomplished by the command `\aftergroup`.

```
{\def\format{lplain}
\ifx\fmtname\format
\else
  \def\format{LaTeX2e}
  \ifx\fmtname\format
  \else
    \aftergroup\endinput
  \fi
\fi}
```

Now that we're sure that the code is seen by LaTeX only, we have to find out what the main (primary) document style is because we want to redefine some macros. This is only necessary for releases of LaTeX dated before december 1991. Therefore this part of the code can optionally be included in `babel.def` by specifying the `docstrip` option `names`.

```
⟨*names⟩
```

The standard styles can be distinguished by checking whether some macros are defined. In table 1 an overview is given of the macros that can be used for this purpose.

article	:	both the `\chapter` and `\opening` macros are undefined
report and book	:	the `\chapter` macro is defined and the `\opening` is undefined
letter	:	the `\chapter` macro is undefined and the `\opening` is defined

Table 1: How to determine the main document style

The macros that have to be redefined for the `report` and `book` document styles happen to be the same, so there is no need to distinguish between those two styles.

`\doc@style` First a parameter `\doc@style` is defined to identify the current document style. This parameter might have been defined by a document style that already uses macros instead of hard-wired texts, such as `artikel1.sty` [6], so the existence of `\doc@style` is checked. If this macro is undefined, i. e., if the document style is unknown and could therefore contain hard-wired texts, `\doc@style` is defined to the default value '0'.

```
\ifx\@undefined\doc@style
  \def\doc@style{0}%
```

This parameter is defined in the following `if` construction (see table 1):

```
  \ifx\@undefined\opening
    \ifx\@undefined\chapter
      \def\doc@style{1}%
    \else
      \def\doc@style{2}%
    \fi
  \else
    \def\doc@style{3}%
  \fi%
\fi%
```

10.12.1 Redefinition of macros

Now here comes the real work: we start to redefine things and replace hard-wired texts by macros. These redefinitions should be carried out conditionally, in case it has already been done.

For the `figure` and `table` environments we have in all styles:

```
\@ifundefined{figurename}{\def\fnum@figure{\figurename{} \thefigure}}{}
\@ifundefined{tablename}{\def\fnum@table{\tablename{} \thetable}}{}
```

The rest of the macros have to be treated differently for each style. When `\doc@style` still has its default value nothing needs to be done.

```
\ifcase \doc@style\relax
\or
```

This means that `babel.def` is read after the `article` style, where no `\chapter` and `\opening` commands are defined[9].

First we have the `\tableofcontents`, `\listoffigures` and `\listoftables`:

```
\@ifundefined{contentsname}%
    {\def\tableofcontents{\section*{\contentsname\@mkboth
          {\uppercase{\contentsname}}{\uppercase{\contentsname}}}%
      \@starttoc{toc}}}{}

\@ifundefined{listfigurename}%
    {\def\listoffigures{\section*{\listfigurename\@mkboth
          {\uppercase{\listfigurename}}{\uppercase{\listfigurename}}}
     \@starttoc{lof}}}{}

\@ifundefined{listtablename}%
    {\def\listoftables{\section*{\listtablename\@mkboth
          {\uppercase{\listtablename}}{\uppercase{\listtablename}}}
      \@starttoc{lot}}}{}
```

Then the `\thebibliography` and `\theindex` environments.

```
\@ifundefined{refname}%
    {\def\thebibliography#1{\section*{\refname
      \@mkboth{\uppercase{\refname}}{\uppercase{\refname}}}%
      \list{[\arabic{enumi}]}{\settowidth\labelwidth{[#1]}%
        \leftmargin\labelwidth
        \advance\leftmargin\labelsep
        \usecounter{enumi}}%
        \def\newblock{\hskip.11em plus.33em minus.07em}%
        \sloppy\clubpenalty4000\widowpenalty\clubpenalty
        \sfcode`\.=1000\relax}}{}

\@ifundefined{indexname}%
    {\def\theindex{\@restonecoltrue\if@twocolumn\@restonecolfalse\fi
     \columnseprule \z@
     \columnsep 35pt\twocolumn[\section*{\indexname}]%
       \@mkboth{\uppercase{\indexname}}{\uppercase{\indexname}}%
       \thispagestyle{plain}%
       \parskip\z@ plus.3pt\parindent\z@\let\item\@idxitem}}{}
```

[9] A fact that was pointed out to me by Nico Poppelier and was already used in Piet van Oostrum's document style option `nl`.

The `abstract` environment:

```
\@ifundefined{abstractname}%
    {\def\abstract{\if@twocolumn
    \section*{\abstractname}%
    \else \small
    \begin{center}%
    {\bf \abstractname\vspace{-.5em}\vspace{\z@}}%
    \end{center}%
    \quotation
    \fi}}{}
```

And last but not least, the macro `\part`:

```
\@ifundefined{partname}%
{\def\@part[#1]#2{\ifnum \c@secnumdepth >\m@ne
        \refstepcounter{part}%
        \addcontentsline{toc}{part}{\thepart
        \hspace{1em}#1}\else
      \addcontentsline{toc}{part}{#1}\fi
   {\parindent\z@ \raggedright
    \ifnum \c@secnumdepth >\m@ne
      \Large \bf \partname{} \thepart
      \par \nobreak
    \fi
    \huge \bf
    #2\markboth{}{}\par}%
    \nobreak
    \vskip 3ex\@afterheading}%
}{}
```

This is all that needs to be done for the `article` style.

```
\or
```

The next case is formed by the two styles `book` and `report`. Basically we have to do the same as for the `article` style, except now we must also change the `\chapter` command.

The tables of contents, figures and tables:

```
\@ifundefined{contentsname}%
    {\def\tableofcontents{\@restonecolfalse
      \if@twocolumn\@restonecoltrue\onecolumn
      \fi\chapter*{\contentsname\@mkboth
          {\uppercase{\contentsname}}{\uppercase{\contentsname}}}%
      \@starttoc{toc}%
      \csname if@restonecol\endcsname\twocolumn
      \csname fi\endcsname}}{}

\@ifundefined{listfigurename}
    {\def\listoffigures{\@restonecolfalse
      \if@twocolumn\@restonecoltrue\onecolumn
      \fi\chapter*{\listfigurename\@mkboth
          {\uppercase{\listfigurename}}{\uppercase{\listfigurename}}}%
      \@starttoc{lof}%
      \csname if@restonecol\endcsname\twocolumn
      \csname fi\endcsname}}{}

```

```
\@ifundefined{listtablename}
    {\def\listoftables{\@restonecolfalse
      \if@twocolumn\@restonecoltrue\onecolumn
      \fi\chapter*{\listtablename\@mkboth
          {\uppercase{\listtablename}}{\uppercase{\listtablename}}}%
      \@starttoc{lot}%
      \csname if@restonecol\endcsname\twocolumn
      \csname fi\endcsname}}{}
```

Again, the `bibliography` and `index` environments; notice that in this case we use `\bibname` instead of `\refname` as in the definitions for the `article` style. The reason for this is that in the `article` document style the term 'References' is used in the definition of `\thebibliography`. In the `report` and `book` document styles the term 'Bibliography' is used.

```
\@ifundefined{bibname}
    {\def\thebibliography#1{\chapter*{\bibname
     \@mkboth{\uppercase{\bibname}}{\uppercase{\bibname}}}%
     \list{[\arabic{enumi}]}{\settowidth\labelwidth{[#1]}%
     \leftmargin\labelwidth \advance\leftmargin\labelsep
     \usecounter{enumi}}%
     \def\newblock{\hskip.11em plus.33em minus.07em}%
     \sloppy\clubpenalty4000\widowpenalty\clubpenalty
     \sfcode`\.=1000\relax}}{}

\@ifundefined{indexname}
    {\def\theindex{\@restonecoltrue\if@twocolumn\@restonecolfalse\fi
    \columnseprule \z@
    \columnsep 35pt\twocolumn[\@makeschapterhead{\indexname}]%
      \@mkboth{\uppercase{\indexname}}{\uppercase{\indexname}}%
    \thispagestyle{plain}%
    \parskip\z@ plus.3pt\parindent\z@ \let\item\@idxitem}}{}
```

Here is the `abstract` environment:

```
\@ifundefined{abstractname}
    {\def\abstract{\titlepage
    \null\vfil
    \begin{center}%
    {\bf \abstractname}%
    \end{center}}}{}
```

And last but not least the `\chapter`, `\appendix` and `\part` macros.

```
\@ifundefined{chaptername}{\def\@chapapp{\chaptername}}{}
%
\@ifundefined{appendixname}
    {\def\appendix{\par
      \setcounter{chapter}{0}%
      \setcounter{section}{0}%
      \def\@chapapp{\appendixname}%
      \def\thechapter{\Alph{chapter}}}}{}
%
\@ifundefined{partname}
    {\def\@part[#1]#2{\ifnum \c@secnumdepth >-2\relax
            \refstepcounter{part}%
            \addcontentsline{toc}{part}{\thepart
            \hspace{1em}#1}\else
            \addcontentsline{toc}{part}{#1}\fi
       \markboth{}{}%
       {\centering
        \ifnum \c@secnumdepth >-2\relax
          \huge\bf \partname{} \thepart
        \par
        \vskip 20pt \fi
        \Huge \bf
        #1\par}\@endpart}}{}%
\or
```

Now we address the case where `babel.def` is read after the `letter` style. The `letter` document style defines the macro `\opening` and some other macros that are specific to `letter`. This means that we have to redefine other macros, compared to the previous two cases.

First two macros for the material at the end of a letter, the `\cc` and `\encl` macros.

```
\@ifundefined{ccname}%
    {\def\cc#1{\par\noindent
     \parbox[t]{\textwidth}%
     {\@hangfrom{\rm \ccname : }\ignorespaces #1\strut}\par}}{}

\@ifundefined{enclname}%
    {\def\encl#1{\par\noindent
     \parbox[t]{\textwidth}%
     {\@hangfrom{\rm \enclname : }\ignorespaces #1\strut}\par}}{}
```

The last thing we have to do here is to redefine the `headings` pagestyle:

```
\@ifundefined{headtoname}
    {\def\ps@headings{%
        \def\@oddhead{\sl \headtoname{} \ignorespaces\toname \hfil
                      \@date \hfil \pagename{} \thepage}%
        \def\@oddfoot{}}}{}
```

This was the last of the four standard document styles, so if `\doc@style` has another value we do nothing and just close the `if` construction.

```
\fi
```

Here ends the code that can be optionally included when a version of LaTeX is in use that is dated *before* december 1991.

⟨/names⟩
⟨/core⟩

10.13 Cross referencing macros

The LaTeX book states:

> The *key* argument is any sequence of letters, digits, and punctuation symbols; upper- and lowercase letters are regarded as different.

When the above quote should still be true when a document is typeset in a language that has active characters, special care has to be taken of the category codes of these characters when they appear in an argument of the cross referencing macros.

When a cross referencing command processes its argument, all tokens in this argument should be character tokens with category 'letter' or 'other'.

The only way to accomplish this in most cases is to use the trick described in the TeXbook [1] (Appendix D, page 382). The primitive `\meaning` applied to a token expands to the current meaning of this token. For example, '`\meaning\A`' with `\A` defined as '`\def\A#1{\B}`' expands to the characters '`macro:#1->\B`' with all category codes set to 'other' or 'space'.

`\bbl@redefine` To redefine a command, we save the old meaning of the macro. Then we redefine it to call the original macro with the 'sanitized' argument. The reason why we do it this way is that we don't want to redefine the LaTeX macros completely in case their definitions change (they have changed in the past).

Bacsuse we need to redefine a number of commands we define the command `\bbl@redefine` which takes care of this. It creates a new control sequence, `\org@...`

```
⟨*core | shorthands⟩
\def\bbl@redefine#1{%
  \edef\bbl@tempa{\expandafter\@gobble\string#1}%
  \expandafter\let\csname org@\bbl@tempa\endcsname#1
  \expandafter\def\csname\bbl@tempa\endcsname}
```

This command should only be used in the preamble of the document.

```
\@onlypreamble\bbl@redefine
```

`\bbl@redefine@long` This version of `\babel@redefine` van be used to redefine `\long` commands such as `\ifthenelse`.

```
\def\bbl@redefine@long#1{%
  \edef\bbl@tempa{\expandafter\@gobble\string#1}%
  \expandafter\let\csname org@\bbl@tempa\endcsname#1
  \expandafter\long\expandafter\def\csname\bbl@tempa\endcsname}
\@onlypreamble\bbl@redefine@long
```

`\bbl@redefinerobust` For commands that are redefined, but which *might* be robust we need a slightly more intelligent macro. A robust command `foo` is defined to expand to `\protect\foo␣`. So it is necessary to check whether `\foo␣` exists.

```
\def\bbl@redefinerobust#1{%
  \edef\bbl@tempa{\expandafter\@gobble\string#1}%
  \expandafter\ifx\csname \bbl@tempa\space\endcsname\relax
    \expandafter\let\csname org@\bbl@tempa\endcsname#1
    \expandafter\edef\csname\bbl@tempa\endcsname{\noexpand\protect
      \expandafter\noexpand\csname\bbl@tempa\space\endcsname}%
  \else
    \expandafter\let\csname org@\bbl@tempa\expandafter\endcsname
                    \csname\bbl@tempa\space\endcsname
  \fi
```

The result of the code above is that the command that is being redefined is always robust afterwards. Therefore all we nee to do now is define `\foo␣`.

```
  \expandafter\def\csname\bbl@tempa\space\endcsname}
```

This command should only be used in the preamble of the document.

```
\@onlypreamble\bbl@redefinerobust
```

\newlabel The macro \label writes a line with a \newlabel command into the .aux file to define labels.

```
\bbl@redefine\newlabel#1#2{%
  \@safe@activestrue\org@newlabel{#1}{#2}\@safe@activesfalse}
```

\@testdef An internal LaTeX macro used to test if the labels that have been written on the .aux file have changed. It is called by the \enddocument macro. This macro needs to be completely rewritten, using \meaning. The reason for this is that in some cases the expansion of \#1@#2 contains the same characters as the #3; but the character codes differ. Therefore LaTeX keeps reporting that the labels may have changed.

```
\def\@testdef #1#2#3{%
  \expandafter\let\expandafter\bbl@tempa\csname #1@#2\endcsname
  \def\bbl@tempb{#3}%
  \ifx\bbl@tempa\relax\else
  \edef\bbl@tempa{\expandafter\strip@prefix\meaning\bbl@tempa}\fi
  \edef\bbl@tempb{\expandafter\strip@prefix\meaning\bbl@tempb}%
  \ifx \bbl@tempa \bbl@tempb
  \else \@tempswatrue \fi}
```

\ref
\pageref The same holds for the macro \ref that references a label and \pageref to reference a page. So we redefine \ref and \pageref. While we change these macros, we make them robust as well (if they weren't already) to prevent problems if they should become expanded at the wrong moment.

```
\bbl@redefinerobust\ref#1{%
  \@safe@activestrue\org@ref{#1}\@safe@activesfalse}
\bbl@redefinerobust\pageref#1{%
  \@safe@activestrue\org@pageref{#1}\@safe@activesfalse}
```

\@citex The macro used to cite from a bibliography, \cite uses an internal macro, \@citex. It is this internal macro that picks up the argument, so we redefine this internal macro and leave \cite alone.

```
\bbl@redefine\@citex[#1]#2{%
  \@safe@activestrue\org@@citex[#1]{#2}\@safe@activesfalse}
```

\nocite The macro \nocite which is used to instruct BiBTeX to extract uncited references from the database.

```
\bbl@redefine\nocite#1{%
  \@safe@activestrue\org@nocite{#1}\@safe@activesfalse}
```

\bibcite The macro that is used in the .aux file to define citation labels.

```
%\bbl@redefine\bibcite#1#2{%
%  \@safe@activestrue\org@bibcite{#1}{#2}\@safe@activesfalse}
```

\@bibitem One of the two internal LaTeX macros called by \bibitem that write the citation label on the .aux file.

```
\bbl@redefine\@bibitem#1{%
  \@safe@activestrue\org@@bibitem{#1}\@safe@activesfalse}
```

\@lbibitem The other of the two internal LaTeX macros called by \bibitem that write the citation label on the .aux file.

```
\bbl@redefine\@lbibitem[#1]#2{%
```

```
  \@safe@activestrue\org@@lbibitem[#1]{#2}\@safe@activesfalse}
⟨/core | shorthands⟩
```

`\ifthenelse` Sometimes a document writer wants to create a special effect depending on the page a certain fragment of text appears on. This can be acheived by the following peice of code:

```
    \ifthenelse{\isodd{pageref{some:label}}}
               {code for odd pages}
               {code for even pages}
```

In order for this to work the argument of `\isodd` needs to be fully expandable. with the above redefinition of `\pageref` it is not in the case of this example. To overcome that we add some code to the definition of `\ifthenelse` to make things work.

The first thing we need to do is check if the package `ifthen` is loaded. This should be done at `\begin{focument}` time.

```
⟨*package⟩
\AtBeginDocument{%
  \@ifpackageloaded{ifthen}{%
```

Then we can redefine `\ifthenelse`:

```
    \bbl@redefine@long\ifthenelse#1#2#3{%
```

We want to revert the definition of `\pageref` to its orginal definition for the duration of `\ifthenelse`, so we first need to store its current meaning.

```
      \let\bbl@tempa\pageref
      \let\pageref\org@pageref
```

Then we can set the `\@safe@actives` switch and call the original `\ifthenelse`.

```
      \@safe@activestrue\org@ifthenelse{#1}{#2}{#3}%
      \@safe@activesfalse
```

Now we need to re-install the stored definition of `\pageref`.

```
      \let\pageref\bbl@tempa
      }%
```

When the package wasn't loaded we do nothing.

```
    }{}%
  }
```

`\@@vpageref` When the package varioref is in use we need to modify its internal command `\@@vpageref` in order to prevent problems when an active character ends up in the argument of `\vref`.

```
\AtBeginDocument{%
  \@ifpackageloaded{varioref}{%
    \bbl@redefinerobust\@@vpageref#1[#2]#3{%
      \@safe@activestrue
      \org@@@vpageref{#1}[#2]{#3}%
      \@safe@activesfalse}%
    }{}%
  }
```

`\hhline` Dealying the activation of the shorthand charactes has introduced a problem with the `hhline` package. The reason is that it uses the ‘:’ character which is made active by the french support in babel. Therefore we need to *reload* the package when the ‘:’ is an active character.

So at `\begin{document}` we check whether `hhline` is loaded.

```
\AtBeginDocument{%
  \@ifpackageloaded{hhline}
```

Then we check whether the expansion of `\normal@char:` is not equal to `\relax`.

```
    {\expandafter\ifx\csname normal@char:\endcsname\relax
     \else
```

In that case we simply reload the package. Note that this happens *after* the category code of the @-sign has been changed to other, so we need to temporarily change it to letter again.

```
       \makeatletter
       \def\@currname{hhline}\input{hhline.sty}\makeatother
     \fi}
    {}}
⟨/package⟩
```

`\nfss@catcodes` LaTeX's font selection scheme sometimes wants to read font definition files in the middle of processing the document. In order to guard against any characters having the wrong `\catcode`'s it always calls `\nfss@catcodes` before loading a file. Unfortunately, the characters `"` and `'` are not dealt with. Therefore we have to add them untill LaTeXdoes that herself.

```
⟨*core | shorthands⟩
\ifx\nfss@catcodes\@undefined
\else
  \addto\nfss@catcodes{%
    \@makeother\'%
    \@makeother\"%
    }
\fi

⟨/core | shorthands⟩
```

11 Local Language Configuration

`\loadlocalcfg` At some sites it may be necessary to add site specific actions to a language definition file. This can be done by creating a file with the same name as the language defintion file, but with the extension `.cfg`. For instance the file `norsk.cfg` will be loaded when the language definition file `norsk.ldf` is loaded.

```
⟨*core⟩
```

For plain based formats we don't want to override the definition of `\loadlocalcfg` from `plain.def`.

```
\ifx\loadlocalcfg\@undefined
  \def\loadlocalcfg#1{%
    \InputIfFileExists{#1.cfg}
           {\typeout{*************************************^^J%
                     * Local config file #1.cfg used^^J%
```

```
                          *}%
                 }
               {}}
\fi
```

Just to be compatible with LaTeX 2.09 we add a few more lines of code:

```
\ifx\@unexpandable@protect\@undefined
  \def\@unexpandable@protect{\noexpand\protect\noexpand}
  \long\def \protected@write#1#2#3{%
        \begingroup
         \let\thepage\relax
         #2%
         \let\protect\@unexpandable@protect
         \edef\reserved@a{\write#1{#3}}%
         \reserved@a
        \endgroup
        \if@nobreak\ifvmode\nobreak\fi\fi
  }
\fi
⟨/core⟩
```

12 Driver files for the documented source code

Since babel version 3.4 all source files that are part of the babel system can be typeset separately. But in order to typeset them all in one document the file `babel.drv` can be used. If you only want the information on how to use the babel system and what goodies are provided by the language spcific files you can run the file `user.drv` through LaTeX to get a user guide.

```
⟨*driver⟩
\documentclass{ltxdoc}
\DoNotIndex{\!,\',\,,\.,\-,\:,\;,\?,\/,\^,\`,\@M}
\DoNotIndex{\@,\@ne,\@m,\@afterheading,\@date,\@endpart}
\DoNotIndex{\@hangfrom,\@idxitem,\@makeschapterhead,\@mkboth}
\DoNotIndex{\@oddfoot,\@oddhead,\@restonecolfalse,\@restonecoltrue}
\DoNotIndex{\@starttoc,\@unused}
\DoNotIndex{\accent,\active}
\DoNotIndex{\addcontentsline,\advance,\Alph,\arabic}
\DoNotIndex{\baselineskip,\begin,\begingroup,\bf,\box,\c@secnumdepth}
\DoNotIndex{\catcode,\centering,\char,\chardef,\clubpenalty}
\DoNotIndex{\columnsep,\columnseprule,\crcr,\csname}
\DoNotIndex{\day,\def,\dimen,\discretionary,\divide,\dp,\do}
\DoNotIndex{\edef,\else,\@empty,\end,\endgroup,\endcsname,\endinput}
\DoNotIndex{\errhelp,\errmessage,\expandafter,\fi,\filedate}
\DoNotIndex{\fileversion,\fmtname,\fnum@figure,\fnum@table,\fontdimen}
\DoNotIndex{\gdef,\global}
\DoNotIndex{\hbox,\hidewidth,\hfil,\hskip,\hspace,\ht,\Huge,\huge}
\DoNotIndex{\ialign,\if@twocolumn,\ifcase,\ifcat,\ifhmode,\ifmmode}
\DoNotIndex{\ifnum,\ifx,\immediate,\ignorespaces,\input,\item}
\DoNotIndex{\kern}
\DoNotIndex{\labelsep,\Large,\large,\labelwidth,\lccode,\leftmargin}
\DoNotIndex{\lineskip,\leavevmode,\let,\list,\ll,\long,\lower}
\DoNotIndex{\m@ne,\mathchar,\mathaccent,\markboth,\month,\multiply}
\DoNotIndex{\newblock,\newbox,\newcount,\newdimen,\newif,\newwrite}
\DoNotIndex{\nobreak,\noexpand,\noindent,\null,\number}
\DoNotIndex{\onecolumn,\or}
\DoNotIndex{\p@,par, \parbox,\parindent,\parskip,\penalty}
\DoNotIndex{\protect,\ps@headings}
\DoNotIndex{\quotation}
\DoNotIndex{\raggedright,\raise,\refstepcounter,\relax,\rm,\setbox}
\DoNotIndex{\section,\setcounter,\settowidth,\scriptscriptstyle}
\DoNotIndex{\sfcode,\sl,\sloppy,\small,\space,\spacefactor,\strut}
\DoNotIndex{\string}
\DoNotIndex{\textwidth,\the,\thechapter,\thefigure,\thepage,\thepart}
\DoNotIndex{\thetable,\thispagestyle,\titlepage,\tracingmacros}
\DoNotIndex{\tw@,\twocolumn,\typeout,\uppercase,\usecounter}
\DoNotIndex{\vbox,\vfil,\vskip,\vspace,\vss}
\DoNotIndex{\widowpenalty,\write,\xdef,\year,\z@,\z@skip}
```

Here `\dlqq` is defined so that an example of `"'` can be given.

```
\makeatletter
\gdef\dlqq{{\setbox\tw@=\hbox{,}\setbox\z@=\hbox{''}%
  \dimen\z@=\ht\z@ \advance\dimen\z@-\ht\tw@
  \setbox\z@=\hbox{\lower\dimen\z@\box\z@}\ht\z@=\ht\tw@
  \dp\z@=\dp\tw@ \box\z@\kern-.04em}}
```

The code lines are numbered within sections,

```
⟨*!user⟩
\@addtoreset{CodelineNo}{section}
\renewcommand\theCodelineNo{%
  \reset@font\scriptsize\thesection.\arabic{CodelineNo}}
```

which should also be visible in the index; hence this redefinition of a macro from `doc.sty`.

```
\renewcommand\codeline@wrindex[1]{\if@filesw
        \immediate\write\@indexfile
            {\string\indexentry{#1}%
            {\number\c@section.\number\c@CodelineNo}}\fi}
```

The glossary environment is used or the change log, but its definition needs changing for this document.

```
\renewenvironment{theglossary}{%
    \glossary@prologue%
    \GlossaryParms \let\item\@idxitem \ignorespaces}%
   {}
⟨/!user⟩
\makeatother
```

A few shorthands used in the documentation

```
\font\manual=logo10 % font used for the METAFONT logo, etc.
\newcommand*\MF{{\manual META}\-{\manual FONT}}
\newcommand*\TeXhax{\TeX hax}
\newcommand*\babel{\textsf{babel}}
\newcommand*\Babel{\textsf{Babel}}
\newcommand*\m[1]{\mbox{$\langle$\it#1\/$\rangle$}}
\newcommand*\langvar{\m{lang}}
```

Some more definitions needed in the documentation.

```
%\newcommand*\note[1]{\textbf{#1}}
\newcommand*\note[1]{}
\newcommand*\bsl{\protect\bslash}
\newcommand*\Lopt[1]{\textsf{#1}}
\newcommand*\file[1]{\texttt{#1}}
\newcommand*\cls[1]{\texttt{#1}}
\newcommand*\pkg[1]{\texttt{#1}}
\newcommand*\langdeffile[1]{%
⟨−user⟩  \clearpage
  \DocInput{#1}}
```

When a full index should be generated uncomment the line with `\EnableCrossres`. Beware, processing may take some time. Use `\DisableCrossrefs` when the index is ready.

```
%  \EnableCrossrefs
\DisableCrossrefs
```

Inlude the change log.

```
⟨−user⟩\RecordChanges
```

The index should use the linenumbers of the code.

```
⟨−user⟩\CodelineIndex
```

Set everything in `\MacroFont` instead of `\AltMacroFont`

```
\setcounter{StandardModuleDepth}{1}
```

For the user guide we only want the description parts of all the files.

```
⟨+user⟩\OnlyDescription
```

Here starts the document

```
\begin{document}
\DocInput{babel.dtx}
```

All the language definition files.

```
⟨+user⟩\clearpage
\langdeffile{esperant.dtx}
\langdeffile{dutch.dtx}
\langdeffile{english.dtx}
\langdeffile{germanb.dtx}
%
\langdeffile{breton.dtx}
\langdeffile{welsh.dtx}
\langdeffile{irish.dtx}
\langdeffile{scottish.dtx}
%
\langdeffile{greek.dtx}
%
\langdeffile{frenchb.dtx}
\langdeffile{italian.dtx}
\langdeffile{portuges.dtx}
\langdeffile{spanish.dtx}
\langdeffile{catalan.dtx}
\langdeffile{galician.dtx}
\langdeffile{romanian.dtx}
%
\langdeffile{danish.dtx}
\langdeffile{norsk.dtx}
\langdeffile{swedish.dtx}
%
\langdeffile{finnish.dtx}
\langdeffile{magyar.dtx}
\langdeffile{estonian.dtx}
%
\langdeffile{croatian.dtx}
\langdeffile{czech.dtx}
\langdeffile{polish.dtx}
\langdeffile{slovak.dtx}
\langdeffile{slovene.dtx}
\langdeffile{russianb.dtx}
%
\langdeffile{lsorbian.dtx}
\langdeffile{usorbian.dtx}
\langdeffile{turkish.dtx}
%
\langdeffile{bahasa.dtx}
\clearpage
\DocInput{bbplain.dtx}
```

Finally print the index and change log (not for the user guide).

```
⟨*!user⟩
\clearpage
```

```
\def\filename{index}
\PrintIndex
\clearpage
\def\filename{changes}
\PrintChanges
⟨/!user⟩
\end{document}
⟨/driver⟩
```

13 Conclusion

A system of document options has been presented that enable the user of LaTeX to adapt the standard document classes of LaTeX to the language he or she prefers to use. These options offer the possibility to switch between languages in one document. The basic interface consists of using ones option, which is the same for *all* standard document classes.

In some cases the language definition files provide macros that can be of use to plain TeX users as well as to LaTeX users. The babel system has been implemented in such a way that it can be used by both groups of users.

14 Acknowledgements

I would like to thank all who volunteered as β-testers for their time. I would like to mention Julio Sanchez who supplied the option file for the Spanish language and Maurizio Codogno who supplied the option file for the Italian language. Michel Goossens supplied contributions for most of the other languages. Nico Poppelier helped polishing the text of the documentation and supplied parts of the macros for the Dutch language. Paul Wackers and Werenfried Spit helped finding and repairing bugs.

During the further development of the babel system I received much help from Bernd Raichle, for which I am grateful.

References

[1] Donald E. Knuth, *The TeXbook*, Addison-Wesley, 1986.

[2] Leslie Lamport, *LaTeX, A document preparation System*, Addison-Wesley, 1986.

[3] K.F. Treebus. *Tekstwijzer, een gids voor het grafisch verwerken van tekst.* SDU Uitgeverij ('s-Gravenhage, 1988). A Dutch book on layout design and typography.

[4] Hubert Partl, *German TeX*, *TUGboat* 9 (1988) #1, p. 70–72.

[5] Leslie Lamport, in: TeXhax Digest, Volume 89, #13, 17 februari 1989.

[6] Johannes Braams, Victor Eijkhout and Nico Poppelier, *The development of national LaTeX styles*, *TUGboat* 10 (1989) #3, p. 401–406.

[7] Joachim Schrod, *International LaTeX is ready to use*, *TUGboat* 11 (1990) #1, p. 87–90.

15 The Esperanto language

The file `esperant.dtx`[10] defines all the language-specific macros for the Esperanto language.

For this language the character `^` is made active. In table 2 an overview is given of its purpose.

`^c`	gives ĉ with hyphenation in the rest of the word allowed, this works for c, C, g, G, H, J, s, S, z, Z
`^h`	prevents ĥ from becoming too tall
`^j`	gives ĵ
`^u`	gives ŭ, with hyphenation in the rest of the word allowed
`^U`	gives Ŭ, with hyphenation in the rest of the word allowed
`^|`	inserts a `\discretionary{-}{}{}`

Table 2: The functions of the active character for Esperanto.

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

```
15.1 ⟨*code⟩
15.2 \LdfInit{esperanto}\captionsesperanto
```

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

```
15.3 \ifx\l@esperanto\@undefined
15.4   \@nopatterns{Esperanto}
15.5   \adddialect\l@esperanto0\fi
```

The next step consists of defining commands to switch to the Esperanto language. The reason for this is that a user might want to switch back and forth between languages.

`\captionsesperanto` The macro `\captionsesperanto` defines all strings used in the four standard documentclasses provided with LaTeX.

```
15.6 \addto\captionsesperanto{%
15.7   \def\prefacename{Anta\u{u}parolo}%
15.8   \def\refname{Cita\^\j{}oj}%
15.9   \def\abstractname{Resumo}%
15.10  \def\bibname{Bibliografio}%
15.11  \def\chaptername{{\^C}apitro}%
15.12  \def\appendixname{Apendico}%
15.13  \def\contentsname{Enhavo}%
15.14  \def\listfigurename{Listo de figuroj}%
15.15  \def\listtablename{Listo de tabeloj}%
15.16  \def\indexname{Indekso}%
```

[10]The file described in this section has version number v1.4j and was last revised on 1997/01/06. A contribution was made by Ruiz-Altaba Marti (`ruizaltb@cernvm.cern.ch`). Code from the file `esperant.sty` by Jörg Knappen (`knappen@vkpmzd.kph.uni-mainz.de`) was included.

```
  \def\figurename{Figuro}%
  \def\tablename{Tabelo}%
  \def\partname{Parto}%
  \def\enclname{Aldono(j)}%
  \def\ccname{Kopie al}%
  \def\headtoname{Al}%
  \def\pagename{Pa\^go}%
  \def\subjectname{Temo}%
  \def\seename{vidu}%   a^u: vd.
  \def\alsoname{vidu anka\u{u}}% a^u vd. anka\u{u}
  \def\proofname{Pruvo}%
  }
```

`\dateesperanto` The macro `\dateesperanto` redefines the command `\today` to produce Esperanto dates.

```
\def\dateesperanto{%
\def\today{\number\day{--a}~de~\ifcase\month\or
  januaro\or februaro\or marto\or aprilo\or majo\or junio\or
  julio\or a\u{u}gusto\or septembro\or oktobro\or novembro\or
  decembro\fi,\space \number\year}}
```

`\extrasesperanto` `\noextrasesperanto` The macro `\extrasesperanto` performs all the extra definitions needed for the Esperanto language. The macro `\noextrasesperanto` is used to cancel the actions of `\extrasesperanto`.

For Esperanto the `^` character is made active. This is done once, later on its definition may vary.

```
\initiate@active@char{^}
\addto\extrasesperanto{\languageshorthands{esperanto}}
\addto\extrasesperanto{\bbl@activate{^}}
\addto\noextrasesperanto{\bbl@deactivate{^}}
```

And here are the uses of the active `^`:

```
\declare@shorthand{esperanto}{^c}{\^{c}\allowhyphens}
\declare@shorthand{esperanto}{^C}{\^{C}\allowhyphens}
\declare@shorthand{esperanto}{^g}{\^{g}\allowhyphens}
\declare@shorthand{esperanto}{^G}{\^{G}\allowhyphens}
\declare@shorthand{esperanto}{^h}{h\llap{\^{}}\allowhyphens}
\declare@shorthand{esperanto}{^H}{\^{H}\allowhyphens}
\declare@shorthand{esperanto}{^j}{\^{\j}\allowhyphens}
\declare@shorthand{esperanto}{^J}{\^{J}\allowhyphens}
\declare@shorthand{esperanto}{^s}{\^{s}\allowhyphens}
\declare@shorthand{esperanto}{^S}{\^{S}\allowhyphens}
\declare@shorthand{esperanto}{^u}{\u u\allowhyphens}
\declare@shorthand{esperanto}{^U}{\u U\allowhyphens}
\declare@shorthand{esperanto}{^|}{\discretionary{-}{}{}\allowhyphens}
```

`\Esper` `\esper` In `esperant.sty` Jörg Knappen provides the macros `\esper` and `\Esper` that can be used instead of `\alph` and `\Alph`. These macros are available in this file as well.

Their definition takes place in three steps. First the toplevel.

```
\def\esper#1{\@esper{\@nameuse{c@#1}}}
\def\Esper#1{\@Esper{\@nameuse{c@#1}}}
```

Then the first five occasions that are probably used the most.

```
\def\@esper#1{\ifcase#1\or a\or b\or c\or \^c\or d\else\@iesper{#1}\fi}
\def\@Esper#1{\ifcase#1\or A\or B\or C\or \^C\or D\else\@Iesper{#1}\fi}
```

And the 33 other cases.

```
\def\@iesper#1{\ifcase#1\or \or \or \or \or \or e\or f\or g\or \^g\or
    h\or h\llap{\^{}}\or i\or j\or \^\j\or k\orl\or m\or n\or o\or
    p\or s\or \^s\or t\or u\or \u{u}\or v\or z\else\@ctrerr\fi}
```

```
\def\@Iesper#1{\ifcase#1\or \or \or \or \or \or E\or F\or G\or \^G\or
    H\or \^H\or I\or J\or \^\J\or K\or L\or M\or N\or O\or
    P\or S\or \^S\or T\or U\or \u{U}\or V\or Z\else\@ctrerr\fi}
```

\hodiau
\hodiaun

In `esperant.sty` Jörg Knappen provides two alternative macros for `\today`, `\hodiau` and `\hodiaun`. The second macro produces an accusative version of the date in Esperanto.

```
\addto\dateesperanto{\def\hodiau{la \today}}
\def\hodiaun{la \number\day --an~de~\ifcase\month\or
  januaro\or februaro\or marto\or aprilo\or majo\or junio\or
  julio\or a\u{u}gusto\or septembro\or oktobro\or novembro\or
  decembro\fi, \space \number\year}
```

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{esperanto}
⟨/code⟩
```

16 The Dutch language

The file `dutch.dtx`[11] defines all the language-specific macros for the Dutch language and the 'Afrikaans' version[12] of it.

For this language the character `"` is made active. In table 3 an overview is given of its purpose. One of the reasons for this is that in the Dutch language a word with a dieresis can be hyphenated just before the letter with the umlaut, but the dieresis has to disappear if the word is broken between the previous letter and the accented letter.

In [3] the quoting conventions for the Dutch language are discussed. The preferred convention is the single-quote Anglo-American convention, i.e. 'This is a quote'. An alternative is the slightly old-fashioned Dutch method with initial double quotes lowered to the baseline, „This is a quote", which should be typed as `"‘This is a quote"’`.

`"a`	`\"a` which hyphenates as `-a`; also implemented for the other letters.
`"y`	puts a negative kern between `i` and `j`
`"Y`	puts a negative kern between `I` and `J`
`"|`	disable ligature at this position.
`"-`	an explicit hyphen sign, allowing hyphenation in the rest of the word.
`"~`	to produce a hyphencharcter without the following `\discretionary{}{}{}`.
`""`	to produce an invisible 'breakpoint'.
`"‘`	lowered double left quotes (see example below).
`"’`	normal double right quotes.
`\-`	like the old `\-`, but allowing hyphenation in the rest of the word.

Table 3: The extra definitions made by `dutch.ldf`

```
% \changes{dutch-3.8a}{1996/10/04}{made check dependant on
%    \cs{CurrentOption}}
%
%    The macro |\LdfInit| takes care of preventing that this file is
%    loaded more than once, checking the category code of the
%    \texttt{@} sign, etc.
% \changes{dutch-3.8a}{1996/10/30}{Now use \cs{LdfInit} to perform
%    initial checks}
%    \begin{macrocode}
⟨*code⟩
\LdfInit\CurrentOption{captions\CurrentOption}
```

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

[11]The file described in this section has version number v3.8c, and was last revised on 1997/01/14.

[12]contributed by Stoffel Lombard (`lombc@b31pc87.up.ac.za`)

First we try to establish with which option we are being processed.

```
\def\bbl@tempa{dutch}
\ifx\CurrentOption\bbl@tempa
```

If it is `dutch` then we first check if the Dutch hyphenation patterns wer loaded,

```
  \ifx\l@dutch\undefined
```

if no we issue a warning and make dutch a 'dialect' of either the hyphenation patterns that were loaded in slot 0 or of 'afrikaans' when it is available.

```
    \@nopatterns{Dutch}
    \ifx\l@afrikaans\undefined
      \adddialect\l@dutch0
    \else
      \adddialect\l@dutch\l@afrikaans
    \fi
  \fi
```

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

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

```
  \begingroup
    \catcode`\"\active
    \def\x{\endgroup
      \def\captionsdutch{%
        \def\prefacename{Voorwoord}%
        \def\refname{Referenties}%
        \def\abstractname{Samenvatting}%
        \def\bibname{Bibliografie}%
        \def\chaptername{Hoofdstuk}%
        \def\appendixname{B"ylage}%
        \def\contentsname{Inhoudsopgave}%
        \def\listfigurename{L"yst van figuren}%
        \def\listtablename{L"yst van tabellen}%
        \def\indexname{Index}%
        \def\figurename{Figuur}%
        \def\tablename{Tabel}%
        \def\partname{Deel}%
        \def\enclname{B"ylage(n)}%
        \def\ccname{cc}%
        \def\headtoname{Aan}%
        \def\pagename{Pagina}%
        \def\seename{zie}%
        \def\alsoname{zie ook}%
        \def\proofname{Bew"ys}%
        }
      }\x
```

`\datedutch` The macro `\datedutch` redefines the command `\today` to produce Dutch dates.

```
  \def\datedutch{%
    \def\today{\number\day~\ifcase\month\or
      januari\or februari\or maart\or april\or mei\or juni\or
      juli\or augustus\or september\or oktober\or november\or
    december\fi
    \space \number\year}}
```

When the option with which this file is being process was not dutch we assume it was afrikaans. We perform a similar check on the availability of the hyphenation paterns.

```
\else
  \ifx\l@afrikaans\undefined
    \@nopatterns{Afrikaans}
    \ifx\l@dutch\undefined
      \adddialect\l@afrikaans0
    \else
      \adddialect\l@afrikaans\l@dutch
    \fi
  \fi
```

`\captionsafrikaans` Now is the time to define the words for 'Afrikaans'.

```
  \def\captionsafrikaans{%
    \def\prefacename{Voorwoord}%
    \def\refname{Verwysings}%
    \def\abstractname{Samevatting}%
    \def\bibname{Bibliografie}%
    \def\chaptername{Hoofstuk}%
    \def\appendixname{Bylae}%
    \def\contentsname{Inhoudsopgawe}%
    \def\listfigurename{Lys van figure}%
    \def\listtablename{Lys van tabelle}%
    \def\indexname{Inhoud}%
    \def\figurename{Figuur}%
    \def\tablename{Tabel}%
    \def\partname{Deel}%
    \def\enclname{Bylae(n)}%
    \def\ccname{a.a.}%
    \def\headtoname{Aan}%
    \def\pagename{Bladsy}%
    \def\seename{sien}%
    \def\alsoname{sien ook}%
    \def\proofname{Bewys}%
    }
```

`\dateafrikaans` Here is the 'Afrikaans' version of the date macro.

```
  \def\dateafrikaans{%
    \def\today{\number\day~\ifcase\month\or
      Januarie\or Februarie\or Maart\or April\or Mei\or Junie\or
      Julie\or  Augustus\or September\or Oktober\or November\or
      Desember\fi
      \space \number\year}}
\fi
```

`\extrasdutch` `\extrasafrikaans` `\noextrasdutch` `\noextrasafrikaans` The macros `\extrasdutch` and `\captionsafrikaans` will perform all the extra definitions needed for the Dutch language. The macros `\noextrasdutch` and `noextrasafrikaans` is used to cancel the actions of `\extrasdutch` and `\captionsafrikaans`.

For Dutch the " character is made active. This is done once, later on its definition may vary. Other languages in the same document may also use the " character for shorthands; we specify that the dutch group of shorthands should be used.

```
\initiate@active@char{"}
```

Both version of the language use the same set of shorthand definitions althoug the 'ij' is not used in Afrikaans.

```
\@namedef{extras\CurrentOption}{\languageshorthands{dutch}}
\expandafter\addto\csname extras\CurrentOption\endcsname{%
  \bbl@activate{"}}
```

The 'umlaut' character should be positioned lower on *all* vowels in Dutch texts.

```
\expandafter\addto\csname extras\CurrentOption\endcsname{%
  \umlautlow\umlautelow}
\@namedef{noextras\CurrentOption}{%
  \umlauthigh}
```

`\dutchhyphenmins` `\afrikaanshyphenmins` The dutch hyphenation patterns can be used with `\lefthyphenmin` set to 2 and `\righthyphenmin` set to 3.

```
\def\dutchhyphenmins{\tw@\thr@@}
\def\afrikaanshyphenmins{\tw@\thr@@}
```

`\@trema` In the Dutch language vowels with a trema are treated specially. If a hyphenation occurs before a vowel-plus-trema, the trema should disappear. To be able to do this we could first define the hyphenation break behaviour for the five vowels, both lowercase and uppercase, in terms of `\discretionary`. But this results in a large `\if`-construct in the definition of the active ". Because we think a user should not use " when he really means something like `''` we chose not to distinguish between vowels and consonants. Therefore we have one macro `\@trema` which specifies the hyphenation break behaviour for all letters.

```
\def\@trema#1{\allowhyphens\discretionary{-}{#1}{\"{#1}}\allowhyphens}
```

Now we can define the doublequote macros: the tremas,

```
\declare@shorthand{dutch}{"a}{\textormath{\@trema a}{\ddot a}}
\declare@shorthand{dutch}{"e}{\textormath{\@trema e}{\ddot e}}
\declare@shorthand{dutch}{"i}{\textormath
  {\allowhyphens\discretionary{-}{i}{\"{\i}}\allowhyphens}%
  {\ddot \imath}}
\declare@shorthand{dutch}{"o}{\textormath{\@trema o}{\ddot o}}
\declare@shorthand{dutch}{"u}{\textormath{\@trema u}{\ddot u}}
```

dutch quotes,

```
\declare@shorthand{dutch}{"`}{%
  \textormath{\quotedblbase{}}{\mbox{\quotedblbase}}}
\declare@shorthand{dutch}{"'}{%
  \textormath{\textquotedblright{}}{\mbox{\textquotedblright}}}
```

and some additional commands:

```
\declare@shorthand{dutch}{"-}{\allowhyphens-\allowhyphens}
\declare@shorthand{german}{"~}{\textormath{\leavevmode\hbox{-}}{-}}
\declare@shorthand{dutch}{"|}{%
  \textormath{\discretionary{-}{}{\kern.03em}}{}}
```

```
\declare@shorthand{dutch}{""}{\hskip\z@skip}
\declare@shorthand{dutch}{"y}{\textormath{\ij{}}{\ddot y}}
\declare@shorthand{dutch}{"Y}{\textormath{\IJ{}}{\ddot Y}}
```

`\-` All that is left now is the redefinition of `\-`. The new version of `\-` should indicate an extra hyphenation position, while allowing other hyphenation positions to be generated automatically. The standard behaviour of TeX in this respect is very unfortunate for languages such as Dutch and German, where long compound words are quite normal and all one needs is a means to indicate an extra hyphenation position on top of the ones that TeX can generate from the hyphenation patterns.

```
\expandafter\addto\csname extras\CurrentOption\endcsname{%
  \babel@save\-}
\expandafter\addto\csname extras\CurrentOption\endcsname{%
  \def\-{\allowhyphens\discretionary{-}{}{}\allowhyphens}}
```

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\CurrentOption
⟨/code⟩
```

17 The English language

The file `english.dtx`[13] defines all the language definition macros for the English language as well as for the American version of this language.

For this language currently no special definitions are needed or available.

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

```
⟨*code⟩
\LdfInit\CurrentOption{date\CurrentOption}
```

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

We allow for the british english patterns to be loaded as either 'english', 'british', or 'UKenglish'

```
\ifx\l@english\@undefined
  \ifx\l@UKenglish\@undefined
    \ifx\l@british\@undefined
      \@nopatterns{English}
      \adddialect\l@english0
    \else
      \let\l@english\l@british
    \fi
  \else
    \let\l@english\l@UKenglish
  \fi
\fi
```

Because we allow 'british' to be used as the babel option we need to make sure that it will be recognised by `\selectlanguage`. In the code above we have made sure that `\l@english` has a sensible value; now we make `\l@british` equal to that.

```
\ifx\l@british\@undefined
  \let\l@british\l@english
\fi
```

'American' is a version of 'English' which can have its own hyphenation patterns. The default english patterns are in fact for american english. We allow for the patterns to be loaded as 'english' 'american' or 'USenglish'.

```
\ifx\l@american\@undefined
  \ifx\l@USenglish\@undefined
```

When the patterns are not know as 'american' or 'USenglish' we add a "dialect".

```
    \adddialect\l@american\l@english
  \else
    \let\l@american\l@USenglish
  \fi
\fi
```

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

[13]The file described in this section has version number v3.3h and was last revised on 1996/12/23.

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

```
\@namedef{captions\CurrentOption}{%
  \def\prefacename{Preface}%
  \def\refname{References}%
  \def\abstractname{Abstract}%
  \def\bibname{Bibliography}%
  \def\chaptername{Chapter}%
  \def\appendixname{Appendix}%
  \def\contentsname{Contents}%
  \def\listfigurename{List of Figures}%
  \def\listtablename{List of Tables}%
  \def\indexname{Index}%
  \def\figurename{Figure}%
  \def\tablename{Table}%
  \def\partname{Part}%
  \def\enclname{encl}%
  \def\ccname{cc}%
  \def\headtoname{To}%
  \def\pagename{Page}%
  \def\seename{see}%
  \def\alsoname{see also}%
  \def\proofname{Proof}%
  }
```

`\dateenglish` The macro `\dateenglish` redefines the command `\today` to produce English dates.

```
\@namedef{date\CurrentOption}{%
\def\today{\ifcase\day\or
  1st\or 2nd\or 3rd\or 4th\or 5th\or
  6th\or 7th\or 8th\or 9th\or 10th\or
  11th\or 12th\or 13th\or 14th\or 15th\or
  16th\or 17th\or 18th\or 19th\or 20th\or
  21st\or 22nd\or 23rd\or 24th\or 25th\or
  26th\or 27th\or 28th\or 29th\or 30th\or
  31st\fi~\ifcase\month\or
  January\or February\or March\or April\or May\or June\or
  July\or August\or September\or October\or November\or December\fi
  \space \number\year}}
```

`\dateamerican` The macro `\dateamerican` redefines the command `\today` to produce American dates.

```
\def\dateamerican{%
\def\today{\ifcase\month\or
  January\or February\or March\or April\or May\or June\or
  July\or August\or September\or October\or November\or December\fi
  \space\number\day, \number\year}}
```

`\extrasenglish`
`\noextrasenglish` The macro `\extrasenglish` will perform all the extra definitions needed for the English language. The macro `\extrasenglish` is used to cancel the actions of `\extrasenglish`. For the moment these macros are empty but they are defined for compatibility with the other language definition files.

```
\@namedef{extras\CurrentOption}{}
```

```
\@namedef{noextras\CurrentOption}{}
```

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\CurrentOption
⟨/code⟩
```

18 The German language

The file `germanb.dtx`[14] defines all the language definition macros for the German language as well as for the Austrian dialect of this language[15].

For this language the character `"` is made active. In table 4 an overview is given of its purpose. One of the reasons for this is that in the German language some character combinations change when a word is broken between the combination. Also the vertical placement of the umlaut can be controlled this way. The quotes

`"a`	`\"a`, also implemented for the other lowercase and uppercase vowels.
`"s`	to produce the German ß (like `\ss{}`).
`"z`	to produce the German ß (like `\ss{}`).
`"ck`	for `ck` to be hyphenated as `k-k`.
`"ff`	for `ff` to be hyphenated as `ff-f`, this is also implemented for l, m, n, p, r and t
`"S`	for `SS` to be `\uppercase{"s}`.
`"Z`	for `SZ` to be `\uppercase{"z}`.
`"|`	disable ligature at this position.
`"-`	an explicit hyphen sign, allowing hyphenation in the rest of the word.
`""`	like `"-`, but producing no hyphen sign (for compund words with hyphen, e.g. `x-""y`).
`"~`	for a compound word mark without a breakpoint.
`"=`	for a compound word mark with a breakpoint, allowing hyphenation in the composing words.
`"‘`	for German left double quotes (looks like ,,).
`"’`	for German right double quotes.
`"<`	for French left double quotes (similar to <<).
`">`	for French right double quotes (similar to >>).

Table 4: The extra definitions made by `german.ldf`

in table 4 can also be typeset by using the commands in table 5.

When this file was read through the option germanb we make it behave as if german was specified.

```
\def\bbl@tempa{germanb}
\ifx\CurrentOption\bbl@tempa
  \def\CurrentOption{german}
\fi
```

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

```
⟨*code⟩
\LdfInit\CurrentOption{captions\CurrentOption}
```

When this file is read as an option, i.e., by the `\usepackage` command, `german` will be an 'unknown' language, so we have to make it known. So we check for the

[14]The file described in this section has version number v2.6d and was last revised on 1996/12/23.

[15]This file is a re-implementation of Hubert Partl's `german.sty` version 2.5b, see [4].

`\glqq`	for German left double quotes (looks like „).
`\grqq`	for German right double quotes (looks like “).
`\glq`	for German left single quotes (looks like ‚).
`\grq`	for German right single quotes (looks like ‘).
`\flqq`	for French left double quotes (similar to <<).
`\frqq`	for French right double quotes (similar to >>).
`\flq`	for (French) left single quotes (similar to <).
`\frq`	for (French) right single quotes (similar to >).
`\dq`	the original quotes character (").

Table 5: More commands which produce quotes, defined by `german.ldf`

existence of `\l@german` to see whether we have to do something here.

```
\ifx\l@german\@undefined
  \@nopatterns{German}
  \adddialect\l@german0
\fi
```

For the Austrian version of these definitions we just add another language.

```
\adddialect\l@austrian\l@german
```

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

`\captionsgerman` `\captionsaustrian` Either the macro `\captionsgerman` or the macro `\captionsaustrian` will define all strings used in the four standard document classes provided with LaTeX.

```
\@namedef{captions\CurrentOption}{%
  \def\prefacename{Vorwort}%
  \def\refname{Literatur}%
  \def\abstractname{Zusammenfassung}%
  \def\bibname{Literaturverzeichnis}%
  \def\chaptername{Kapitel}%
  \def\appendixname{Anhang}%
  \def\contentsname{Inhaltsverzeichnis}%    % oder nur: Inhalt
  \def\listfigurename{Abbildungsverzeichnis}%
  \def\listtablename{Tabellenverzeichnis}%
  \def\indexname{Index}%
  \def\figurename{Abbildung}%
  \def\tablename{Tabelle}%                  % oder: Tafel
  \def\partname{Teil}%
  \def\enclname{Anlage(n)}%                 % oder: Beilage(n)
  \def\ccname{Verteiler}%                   % oder: Kopien an
  \def\headtoname{An}%
  \def\pagename{Seite}%
  \def\seename{siehe}%
  \def\alsoname{siehe auch}%
  \def\proofname{Beweis}%
  }
```

`\dategerman` The macro `\dategerman` redefines the command `\today` to produce German dates.

```
\def\month@german{\ifcase\month\or
```

```
  Januar\or Februar\or M\"arz\or April\or Mai\or Juni\or
  Juli\or August\or September\or Oktober\or November\or Dezember\fi}
\def\dategerman{\def\today{\number\day.~\month@german
  \space\number\year}}
```

`\dateaustrian` The macro `\dateaustrian` redefines the command `\today` to produce Austrian version of the German dates.

```
\def\dateaustrian{\def\today{\number\day.~\ifnum1=\month
  J\"anner\else \month@german\fi \space\number\year}}
```

`\extrasgerman` `\extrasaustrian` `\noextrasgerman` `\noextrasaustrian` Either the macro `\extrasgerman` or the macros `\extrasaustrian` will perform all the extra definitions needed for the German language. The macro `\noextrasgerman` is used to cancel the actions of `\extrasgerman`.

For German (as well as for Dutch) the `"` character is made active. This is done once, later on its definition may vary.

```
\initiate@active@char{"}
\@namedef{extras\CurrentOption}{%
  \languageshorthands{german}}
\expandafter\addto\csname extras\CurrentOption\endcsname{%
  \bbl@activate{"}}
%\addto\noextrasgerman{\bbl@deactivate{"}}
```

In order for TeX to be able to hyphenate German words which contain 'ß' (in the `OT1` position `^^Y`) we have to give the character a nonzero `\lccode` (see Appendix H, the TeXbook).

```
\expandafter\addto\csname extras\CurrentOption\endcsname{%
  \babel@savevariable{\lccode25}%
  \lccode25=25}
```

The umlaut accent macro `\"` is changed to lower the umlaut dots. The redefinition is done with the help of `\umlautlow`.

```
\expandafter\addto\csname extras\CurrentOption\endcsname{%
  \babel@save\"\umlautlow}
\@namedef{noextras\CurrentOption}{\umlauthigh}
```

The german hyphenation patterns can be used with `\lefthyphenmin` and `\righthyphenmin` set to 2.

```
\def\germanhyphenmins{\tw@\tw@}
```

The code above is necessary because we need an extra active character. This character is then used as indicated in table 4.

To be able to define the function of `"`, we first define a couple of 'support' macros.

`\dq` We save the original double quote character in `\dq` to keep it available, the math accent `\"` can now be typed as `"`.

```
\begingroup \catcode`\"12
\def\x{\endgroup
  \def\@SS{\mathchar"7019 }
  \def\dq{"}}
\x
```

Now we can define the doublequote macros: the umlauts,

```
\declare@shorthand{german}{"a}{\textormath{\"{a}\allowhyphens}{\ddot a}}
```

```
\declare@shorthand{german}{"o}{\textormath{\"{o}\allowhyphens}{\ddot o}}
\declare@shorthand{german}{"u}{\textormath{\"{u}\allowhyphens}{\ddot u}}
\declare@shorthand{german}{"A}{\textormath{\"{A}\allowhyphens}{\ddot A}}
\declare@shorthand{german}{"O}{\textormath{\"{O}\allowhyphens}{\ddot O}}
\declare@shorthand{german}{"U}{\textormath{\"{U}\allowhyphens}{\ddot U}}
```

tremas,

```
\declare@shorthand{german}{"e}{\textormath{\"{e}}{\ddot e}}
\declare@shorthand{german}{"E}{\textormath{\"{E}}{\ddot E}}
\declare@shorthand{german}{"i}{\textormath{\"{\i}}%
                              {\ddot\imath}}
\declare@shorthand{german}{"I}{\textormath{\"{I}}{\ddot I}}
```

german es-zet (sharp s),

```
\declare@shorthand{german}{"s}{\textormath{\ss{}}{\@SS{}}}
\declare@shorthand{german}{"S}{SS}
\declare@shorthand{german}{"z}{\textormath{\ss{}}{\@SS{}}}
\declare@shorthand{german}{"Z}{SZ}
```

german and french quotes,

```
\declare@shorthand{german}{"`}{\glqq}
\declare@shorthand{german}{"'}{\grqq}
\declare@shorthand{german}{"<}{\flqq}
\declare@shorthand{german}{">}{\frqq}
```

discretionary commands

```
\declare@shorthand{german}{"c}{\textormath{\bbl@disc ck}{c}}
\declare@shorthand{german}{"C}{\textormath{\bbl@disc CK}{C}}
\declare@shorthand{german}{"f}{\textormath{\bbl@disc f{ff}}{f}}
\declare@shorthand{german}{"F}{\textormath{\bbl@disc F{FF}}{F}}
\declare@shorthand{german}{"l}{\textormath{\bbl@disc l{ll}}{l}}
\declare@shorthand{german}{"L}{\textormath{\bbl@disc L{LL}}{L}}
\declare@shorthand{german}{"m}{\textormath{\bbl@disc m{mm}}{m}}
\declare@shorthand{german}{"M}{\textormath{\bbl@disc M{MM}}{M}}
\declare@shorthand{german}{"n}{\textormath{\bbl@disc n{nn}}{n}}
\declare@shorthand{german}{"N}{\textormath{\bbl@disc N{NN}}{N}}
\declare@shorthand{german}{"p}{\textormath{\bbl@disc p{pp}}{p}}
\declare@shorthand{german}{"P}{\textormath{\bbl@disc P{PP}}{P}}
\declare@shorthand{german}{"r}{\textormath{\bbl@disc r{rr}}{r}}
\declare@shorthand{german}{"R}{\textormath{\bbl@disc R{RR}}{R}}
\declare@shorthand{german}{"t}{\textormath{\bbl@disc t{tt}}{t}}
\declare@shorthand{german}{"T}{\textormath{\bbl@disc T{TT}}{T}}
```

and some additional commands:

```
\declare@shorthand{german}{"-}{\penalty\@M\-\allowhyphens}
\declare@shorthand{german}{"|}{%
  \textormath{\penalty\@M\discretionary{-}{}{\kern.03em}%
              \allowhyphens}{}}
\declare@shorthand{german}{""}{\hskip\z@skip}
\declare@shorthand{german}{"~}{\textormath{\leavevmode\hbox{-}}{-}}
\declare@shorthand{german}{"=}{\penalty\@M-\hskip\z@skip}
```

`\mdqon` `\mdqoff` `\ck` All that's left to do now is to define a couple of commands for reasons of compatibility with `german.sty`.

```
\def\mdqon{\bbl@activate{"}}
\def\mdqoff{\bbl@deactivate{"}}
\def\ck{\allowhyphens\discretionary{k-}{k}{ck}\allowhyphens}
```

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\CurrentOption
⟨/code⟩
```

19 The Breton language

The file `breton.dtx`[16] defines all the language-specific macros for the Breton language.

There are not really typographic rules for the Breton language. It is a local language (it's one of the celtic languages) which is spoken in Brittany (West of France). So we have a synthesis between french typographic rules and english typographic rules. The characters `:`, `;`, `!` and `?` are made active in order to get a whitespace automatically before these characters.

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

```
⟨*code⟩
\LdfInit{breton}\captionsbreton
```

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

```
\ifx\l@breton\@undefined
    \@nopatterns{Breton}
    \adddialect\l@breton0\fi
```

The next step consists of defining commands to switch to the English language. The reason for this is that a user might want to switch back and forth between languages.

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

```
\addto\captionsbreton{%
  \def\prefacename{Rakskrid}%
  \def\refname{Daveenno\`u}%
  \def\abstractname{Dvierra\~n}%
  \def\bibname{Lennadurezh}%
  \def\chaptername{Pennad}%
  \def\appendixname{Stagadenn}%
  \def\contentsname{Taolenn}%
  \def\listfigurename{Listenn ar Figurenno\`u}%
  \def\listtablename{Listenn an taolenno\`u}%
  \def\indexname{Meneger}%
  \def\figurename{Figurenn}%
  \def\tablename{Taolenn}%
  \def\partname{Lodenn}%
  \def\enclname{Diello\`u kevret}%
  \def\ccname{Eilskrid da}%
  \def\headtoname{evit}
  \def\pagename{Pajenn}%
  \def\seename{Gwelout}%
  \def\alsoname{Gwelout ivez}%
  \def\proofname{Proof}%  <-- needs translation
}
```

[16]The file described in this section has version number v1.0e and was last revised on 1996/12/23.

`\datebreton` The macro `\datebreton` redefines the command `\today` to produce Breton dates.

```
\def\datebreton{%
\def\today{\ifnum\day=1\relax 1\/$^{\rm a\tilde{n}}$\else
  \number\day\fi \space a\space viz\space\ifcase\month\or
  Genver\or C'hwevrer\or Meurzh\or Ebrel\or Mae\or Mezheven\or
  Gouere\or Eost\or Gwengolo\or Here\or Du\or Kerzu\fi
  \space\number\year}}
```

`\extrasbreton` `\noextrasbreton` The macro `\extrasbreton` will perform all the extra definitions needed for the Breton language. The macro `\noextrasbreton` is used to cancel the actions of `\extrasbreton`.

The category code of the characters `:`, `;`, `!` and `?` is made `\active` to insert a little white space.

```
\initiate@active@char{:}
\initiate@active@char{;}
\initiate@active@char{!}
\initiate@active@char{?}
```

We specify that the breton group of shorthands should be used.

```
\addto\extrasbreton{\languageshorthands{breton}}
```

These characters are 'turned on' once, later their definition may vary.

```
\addto\extrasbreton{%
  \bbl@activate{:}\bbl@activate{;}%
  \bbl@activate{!}\bbl@activate{?}}
%\addto\noextrasbreton{%
%  \bbl@deactivate{:}\bbl@deactivate{;}%
%  \bbl@deactivate{!}\bbl@deactivate{?}}
```

The last thing `\extrasbreton` needs to do is to make sure that `\frenchspacing` is in effect. If this is not the case the execution of `\noextrasbreton` will switch it of again.

```
\addto\extrasbreton{\bbl@frenchspacing}
\addto\noextrasbreton{\bbl@nonfrenchspacing}
```

`\breton@sh@;@` We have to reduce the amount of white space before `;`, `:` and `!` when the user types a space in front of these characters. This should only happen outside mathmode, hence the test with `\ifmmode`.

```
\declare@shorthand{breton}{;}{%
    \ifmmode
      \string;\space
    \else\relax
```

In horizontal mode we check for the presence of a 'space' and replace it by a `\thinspace`.

```
      \ifhmode
        \ifdim\lastskip>\z@
          \unskip\penalty\@M\thinspace
        \fi
      \fi
      \string;\space
    \fi}%
```

`\breton@sh@:@` `\breton@sh@!@` Because these definitions are very similar only one is displayed in a way that the definition can be easily checked.

```
\declare@shorthand{breton}{:}{%
  \ifmmode\string:\space
  \else\relax
    \ifhmode
      \ifdim\lastskip>\z@\unskip\penalty\@M\thinspace\fi
    \fi
    \string:\space
  \fi}
\declare@shorthand{breton}{!}{%
  \ifmmode\string!\space
  \else\relax
    \ifhmode
      \ifdim\lastskip>\z@\unskip\penalty\@M\thinspace\fi
    \fi
    \string!\space
  \fi}
```

`\breton@sh@?@` For the question mark something different has to be done. In this case the amount of white space that replaces the space character depends on the dimensions of the font.

```
\declare@shorthand{breton}{?}{%
  \ifmmode
    \string?\space
  \else\relax
    \ifhmode
      \ifdim\lastskip>\z@
        \unskip
        \kern\fontdimen2\font
        \kern-1.4\fontdimen3\font
      \fi
    \fi
    \string?\space
  \fi}
```

All that is left to do now is provide the breton user with some extra utilities. Some definitions for special characters.

```
\DeclareTextSymbol{\at}{OT1}{64}
\DeclareTextSymbol{\at}{T1}{64}
\DeclareTextSymbolDefault{\at}{OT1}
\DeclareTextSymbol{\boi}{OT1}{92}
\DeclareTextSymbol{\boi}{T1}{16}
\DeclareTextSymbolDefault{\boi}{OT1}
\DeclareTextSymbol{\circonflexe}{OT1}{94}
\DeclareTextSymbol{\circonflexe}{T1}{2}
\DeclareTextSymbolDefault{\circonflexe}{OT1}
\DeclareTextSymbol{\tild}{OT1}{126}
\DeclareTextSymbol{\tild}{T1}{3}
\DeclareTextSymbolDefault{\tild}{OT1}
\DeclareTextSymbol{\degre}{OT1}{23}
\DeclareTextSymbol{\degre}{T1}{6}
\DeclareTextSymbolDefault{\degre}{OT1}
```

The following macros are used in the redefinition of `\^` and `\"` to handle the letter i.

```
\AtBeginDocument{%
  \DeclareTextCompositeCommand{\^}{OT1}{i}{\^\i}
  \DeclareTextCompositeCommand{\"}{OT1}{i}{\"\i}}
```

And some more macros for numbering.

```
\def\kentan{1\/${}^{\rm a\tilde{n}}$}
\def\eil{2\/${}^{\rm l}$}
\def\re{\/${}^{\rm re}$}
\def\trede{3\re}
\def\pevare{4\re}
\def\vet{\/${}^{\rm vet}$}
\def\pempvet{5\vet}
```

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{breton}
⟨/code⟩
```

20 The Welsh language

The file `welsh.dtx`[17] defines all the language definition macros for the Welsh language as well as for the ¡Dialect¿ version of this language.

For this language currently no special definitions are needed or available.

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

```
⟨*code⟩
\LdfInit{welsh}{captionswelsh}
```

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

```
\ifx\undefined\l@welsh
  \@nopatterns{welsh}
  \adddialect\l@welsh0\fi
```

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

`\welshhyphenmins` This macro is used to store the correct values of the hyphenation parameters `\lefthyphenmin` and `\righthyphenmin`.

```
\def\welshhyphenmins{23}
```

`\captionswelsh` The macro `\captionswelsh` defines all strings used in the four standard documentclasses provided with LaTeX.

```
\def\captionswelsh{%
  \def\prefacename{Rhagair}%
  \def\refname{Cyfeiriadau}%
  \def\abstractname{Crynodeb}%
  \def\bibname{Llyfryddiaeth}%
  \def\chaptername{Pennod}%
  \def\appendixname{Atodiad}%
  \def\contentsname{Cynnwys}%
  \def\listfigurename{Rhestr Ddarluniau}%
  \def\listtablename{Rhestr Dablau}%
  \def\indexname{Mynegai}%
  \def\figurename{Darlun}%
  \def\tablename{Taflen}%
  \def\partname{Rhan}%
  \def\enclname{amgae\"edig}%
  \def\ccname{cop\"\i au}%
  \def\headtoname{At}%  % 'at' on letters meaning 'to ( a person)'
                        % 'to (a place)' is 'i' in Welsh
  \def\pagename{tudalen}%
  \def\seename{gweler}%
  \def\alsoname{gweler hefyd}%
  \def\proofname{Prawf}%
  }
```

[17]The file described in this section has version number v1.0a and was last revised on 1996/12/23.

\datewelsh The macro `\datewelsh` redefines the command `\today` to produce welsh dates.

```
\def\datewelsh{%
  \def\today{\ifnum\day=1\relax 1\/$^{\mathrm{a\tilde{n}}}$\else
    \number\day\fi \space a\space viz\space\ifcase\month\or
    Ionawr\or Chwefror\or Mawrth\or Ebrill\or
    Mai\or Mehefin\or Gorffennaf\or Awst\or
    Medi\or Hydref\or Tachwedd\or Rhagfyr\fi
  \space\number\year}}
```

\extraswelsh \noextraswelsh The macro `\extraswelsh` will perform all the extra definitions needed for the welsh language. The macro `\noextraswelsh` is used to cancel the actions of `\extraswelsh`. For the moment these macros are empty but they are defined for compatibility with the other language definition files.

```
\addto\extraswelsh{}
\addto\noextraswelsh{}
```

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{welsh}
⟨/code⟩
```

21 The Irish language

The file `irish.dtx`[18] defines all the language definition macros for the Irish language.

For this language currently no special definitions are needed or available.

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

```
⟨*code⟩
\LdfInit{irish}\captionsirish
```

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

```
\ifx\l@irish\@undefined
  \@nopatterns{irish}
  \adddialect\l@irish0\fi
```

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

`\captionsirish` The macro `\captionsirish` defines all strings used in the four standard documentclasses provided with LaTeX.

```
\addto\captionsirish{%
  \def\prefacename{Preface}%    <-- needs translation
  \def\refname{Tagairt\'{\i}}%
  \def\abstractname{Achoimre}%
  \def\bibname{Leabharliosta}%
  \def\chaptername{Caibidil}%
  \def\appendixname{Aguis\'{\i}n}%
  \def\contentsname{Cl\'ar \'Abhair}%
  \def\listfigurename{L\'ear\'aid\'{\i}}%
  \def\listtablename{T\'abla\'{\i}}%
  \def\indexname{Inn\'eacs}%
  \def\figurename{L\'ear\'aid}%
  \def\tablename{T\'abla}%
  \def\partname{Cuid}%
  \def\enclname{faoi iamh}%
  \def\ccname{cc}%                abrv. 'c\'oip chuig'
  \def\headtoname{Go}%
  \def\pagename{Leathanach}%
  \def\seename{see}%    <-- needs translation
  \def\alsoname{see also}%    <-- needs translation
  \def\proofname{Proof}%    <-- needs translation
  }
```

`\dateirish` The macro `\dateirish` redefines the command `\today` to produce Irish dates.

```
\def\dateirish{\def\today{%
  \number\day\space \ifcase\month\or
  Ean\'air\or Feabhra\or M\'arta\or Aibre\'an\or
  Bealtaine\or Meitheamh\or I\'uil\or L\'unasa\or
  Me\'an F\'omhair\or Deireadh F\'omhair\or
```

[18]The file described in this section has version number v1.0f and was last revised on 1998/05/05. A contribution was made by Marion Gunn.

```
  M\'{\i} na Samhna\or M\'{\i} na Nollag\fi
  \space \number\year}}
```

`\extrasirish` `\noextrasirish` The macro `\extrasirish` will perform all the extra definitions needed for the Irish language. The macro `\noextrasirish` is used to cancel the actions of `\extrasirish`. For the moment these macros are empty but they are defined for compatibility with the other language definition files.

```
\addto\extrasirish{}
\addto\noextrasirish{}
```

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{irish}
⟨/code⟩
```

22 The Scottish language

The file `scottish.dtx`[19] defines all the language definition macros for the Scottish language.

For this language currently no special definitions are needed or available.

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

```
⟨*code⟩
\LdfInit{scottish}\captionsscottish
```

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

```
\ifx\l@scottish\@undefined
  \@nopatterns{scottish}
  \adddialect\l@scottish0\fi
```

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

`\captionsscottish` The macro `\captionsscottish` defines all strings used in the four standard documentclasses provided with LaTeX.

```
\addto\captionsscottish{%
  \def\prefacename{Preface}%    <-- needs translation
  \def\refname{Iomraidh}%
  \def\abstractname{Br\'{\i}gh}%
  \def\bibname{Leabhraichean}%
  \def\chaptername{Caibideil}%
  \def\appendixname{Ath-sgr'{\i}obhadh}%
  \def\contentsname{Cl\'ar-obrach}%
  \def\listfigurename{Liosta Dhealbh }%
  \def\listtablename{Liosta Chl\'ar}%
  \def\indexname{Cl\'ar-innse}%
  \def\figurename{Dealbh}%
  \def\tablename{Cl\'ar}%
  \def\partname{Cuid}%
  \def\enclname{a-staigh}%
  \def\ccname{lethbhreac gu}%
  \def\headtoname{gu}%
  \def\pagename{t.d.}%             abrv. 'taobh duilleag'
  \def\seename{see}%    <-- needs translation
  \def\alsoname{see also}%    <-- needs translation
  \def\proofname{Proof}%    <-- needs translation
}
```

`\datescottish` The macro `\datescottish` redefines the command `\today` to produce Scottish dates.

```
\def\datescottish{%
  \number\day\space \ifcase\month\or
  am Faoilteach\or an Gearran\or am M\'art\or an Giblean\or
```

[19]The file described in this section has version number v1.0d and was last revised on 1996/12/23. A contribution was made by Fraser Grant (FRASER@CERNVM).

```
  an C\`eitean\or an t-\`Og mhios\or an t-Iuchar\or
  L\`unasdal\or an Sultuine\or an D\`amhar\or
  an t-Samhainn\or an Dubhlachd\fi
  \space \number\year}}
```

`\extrasscottish` `\noextrasscottish` The macro `\extrasscottish` will perform all the extra definitions needed for the Scottish language. The macro `\noextrasscottish` is used to cancel the actions of `\extrasscottish`. For the moment these macros are empty but they are defined for compatibility with the other language definition files.

```
\addto\extrasscottish{}
\addto\noextrasscottish{}
```

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{scottish}
⟨/code⟩
```

23 The Greek language

The file `greek.dtx`[20] defines all the language definition macros for the Greek language.

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

```
⟨*code⟩
\LdfInit{greek}\captionsgreek
```

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

```
\ifx\l@greek\@undefined
  \@nopatterns{greek}
  \adddialect\l@greek0\fi
```

Tyesetting Greek texts implies that a special set of fonts needs to be used. The current support for greek uses a set of fonts that more or less 'match' with the computer modern fonts. Therefore we define the Local GReek encoding (LGR, see the file `greek.fdd`). We make sure that this encoding is known to LaTeX.

```
\input{LGRenc.def}
```

`\latinencoding` We need to know the encoding for text that is supposed to be typeset in latin text. We assume that it will be the encoding which is active at the end of the babel package. If the `fontenc` package is loaded later, then... too bad!

```
\AtEndOfPackage{\edef\latinencoding{\cf@encoding}}
```

Now we define two commands that offer the possibility to switch between greek and roman encodings.

`\greektext` `\latintext` The command `\greektext` will switch from latin font encoding to the greek font encoding, the command `\latintext` switches back. This assumes that the 'normal' font encoding is a latin one. These commands are *declarations*, for shorter peaces of text the commands `\textlatin` and `\textgreek` can be used.

```
\DeclareRobustCommand{\greektext}{%
  \fontencoding{LGR}\selectfont
  \def\encodingdefault{LGR}}
\DeclareRobustCommand{\latintext}{%
  \fontencoding{\latinencoding}\selectfont
  \def\encodingdefault{\latinencoding}}
```

`\textgreek` `\textlatin` These commands take an argument which is then typeset using the requested font encoding. In order to avoid many encoding switches both commands operate in a local scope.

```
\DeclareRobustCommand{\textgreek}[1]{{\greektext #1}}
\DeclareRobustCommand{\textlatin}[1]{{\latintext #1}}
```

`\textol` A last aspect of the set of fonts provided with this version of support for typesetting Greek texts is that it contains an outline font. In order to make it available we define the command `\textol`.

[20]The file described in this section has version number v1.0b and was last revised on 1996/12/23. The original author is Apostolos Syropoulos (`apostolo@platon.ee.duth.gr`).

```
\def\outlfamily{\usefont{LGR}{cmro}{m}{n}}
\DeclareTextFontCommand{\textol}{\outlfamily}
```

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

`\greekhyphenmins` This macro is used to store the correct values of the hyphenation parameters `\lefthyphenmin` and `\righthyphenmin`.

```
\def\greekhyphenmins{23}
```

`\captionsgreek` The macro `\captionsgreek` defines all strings used in the four standard documentclasses provided with LaTeX.

```
\addto\captionsgreek{%
  \def\prefacename{Pr'ologoc}%
  \def\refname{Anafor'ec}%
  \def\abstractname{Per'ilhyh}%
  \def\bibname{Bibliograf'ia}%
  \def\chaptername{Kef'alaio}%
  \def\appendixname{Par'arthma}%
  \def\contentsname{Perieq'omena}%
  \def\listfigurename{Kat'alogoc Sqhm'atwn}%
  \def\listtablename{Kat'alogoc Pin'akwn}%
  \def\indexname{Euret'hrio}%
  \def\figurename{Sq'hma}%
  \def\tablename{P'inakac}%
  \def\partname{M'eroc}%
  \def\enclname{Sunhmm'ena}%
  \def\ccname{Koinopo'ihsh}%
  \def\headtoname{Proc}%
  \def\pagename{Sel'ida}%
  \def\seename{bl'epe}%
  \def\alsoname{bl'pe ep'ishc}%
  }
```

`\dategreek` The macro `\dategreek` redefines the command `\today` to produce greek dates.

```
\def\dategreek{%
  \def\today{\number\day \space%
    \ifcase\month\or
    Ianouar'iou\or Februar'iou\or Mart'iou\or April'iou\or
    Ma'"iou\or Ioun'iou\or Ioul'iou\or Augo'ustou\or
    Septembr'iou\or Oktobr'iou\or Noembr'iou\or Dekembr'iou\fi
    \space \number\year}}
```

`\extrasgreek` `\noextrasgreek` The macro `\extrasgreek` will perform all the extra definitions needed for the Greek language. The macro `\noextrasgreek` is used to cancel the actions of `\extrasgreek`. For the moment these macros switch the fontencoding used and the definition of the internal macros `\@alph` and `\@Alph` because the greek alpabetisation differs considerably from the roman alpabetisation..

```
\addto\extrasgreek{\greektext}
\addto\noextrasgreek{\latintext}
```

`\greek@alph` `\greek@Alph` With the latin transcription used to create documents that will be typeset in the greek language we need to adopt the alphabetisation considerably. Therefore we have to redefine the internal LaTeX commands `\@alph` and `\@Alph`.

We need to be able to switch between the original definitions and the greek ones, so we first 'save' the orginal definitions.

```
\let\latin@alph\@alph
\let\latin@Alph\@Alph
```

Then we define the greek versions

```
\def\greek@alph#1{%
  \ifcase #1\or a\or b\or g\or d\or e\or st\or z\or h\or j\or i\or
    ia\or ib\or ig\or id\or ie\or ist\or iz\or ih\or ij\or k\or ka\or
    kb\or kg\or \kd \or ke\or kst
  \else
    \@ctrerr
  \fi
  $'$}
\def\greek@Alph#1{%
   \ifcase #1\or A\or B\or G\or D\or E\or St\or Z\or H\or J\or I\or
    IA\or IB\or IG\or ID\or IE\or IST\or IZ\or IH\or IJ\or K\or KA\or
    KB\or KG\or \KD \or KE\or KST
  \else
    \@ctrerr
  \fi
  $'$}
```

Now we can set up the switching.

```
\addto\extrasgreek{%
  \let\@alph\@greek@alph
  \let\@Alph\@greek@Alph}
\addto\noextrasgreek{%
  \let\@alph\latin@alph
  \let\@Alph\latin@Alph}
```

`\greek@roman` `\greek@Roman` To prevent roman numerals being typeset in greek letters we need to adopt the internal LaTeX commands `\@roman` and `\@Roman`.

```
\let\latin@roman\@roman
\let\latin@Roman\@Roman
\def\greek@roman#1{\textlatin{\latin@roman{#1}}}
\def\greek@Roman#1{\textlatin{\latin@Roman{#1}}}
\addto\extrasgreek{%
  \let\@roman\@greek@roman
  \let\@Roman\@greek@Roman}
\addto\noextrasgreek{%
  \let\@roman\latin@roman
  \let\@Roman\latin@Roman}
```

We provide access to a few extra greek characters. They are only available in one particular font, therefore we first define a 'helper' macro to select the correct font.

```
\def\greek@char#1{{%
    \fontfamily\rmdefault
    \fontseries\mddefault
    \fontshape\scdefault
    \selectfont\char#1}}

\DeclareTextCommand{\tao}{LGR}{\greek@char{"7F}}
\DeclareTextCommand{\Qoppa}{LGR}{\greek@char{"43}}
```

```
\DeclareTextCommand{\qoppa}{LGR}{\greek@char{"5B}}
\DeclareTextCommand{\varqoppa}{LGR}{\greek@char{"5C}}
\DeclareTextCommand{\Sampi}{LGR}{\greek@char{"5F}}
\DeclareTextCommand{\sampi}{LGR}{\greek@char{"5E}}
\DeclareTextCommand{\Digamma}{LGR}{\greek@char{"17}}
```

The amssymb package defines a `\digamma` command, so in order to avoid problems we spell here digamma intentionally erroneous with a double d.

```
\DeclareTextCommand{\ddigamma}{LGR}{\greek@char{"60}}
\DeclareTextCommand{\vardigamma}{LGR}{\greek@char{"5D}}
```

Now make sure that these commands can also be used outside of the greek font encoding.

```
\ProvideTextCommandDefault{\tao}{\UseTextSymbol{LGR}{\tao}}
\ProvideTextCommandDefault{\Qoppa}{\UseTextSymbol{LGR}{\Qoppa}}
\ProvideTextCommandDefault{\qoppa}{\UseTextSymbol{LGR}{\qoppa}}
\ProvideTextCommandDefault{\varqoppa}{\UseTextSymbol{LGR}{\varqoppa}}
\ProvideTextCommandDefault{\Sampi}{\UseTextSymbol{LGR}{\Sampi}}
\ProvideTextCommandDefault{\sampi}{\UseTextSymbol{LGR}{\sampi}}
\ProvideTextCommandDefault{\Digamma}{\UseTextSymbol{LGR}{\Digamma}}
\ProvideTextCommandDefault{\ddigamma}{\UseTextSymbol{LGR}{\ddigamma}}
\ProvideTextCommandDefault{\vardigamma}{%
  \UseTextSymbol{LGR}{\vardigamma}}
```

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{greek}
⟨/code⟩
```

24 The French Language

24.1 About French typography

The file `frenchb.dtx`[21], derived from `frenchy.sty`, defines all the language definition macros for the French language.

Customization for the French language is achieved following the book "Lexique des règles typographiques en usage à l'Imprimerie nationale" troisième édition (1994), ISBN-2-11-081075-0.

This file has been designed to be used with LaTeX 2ε, LaTeX-2.09 and PlainTeX formats. If you are still using LaTeX-2.09, you *should* consider switching to LaTeX 2ε!

Any of the commands `\selectlanguage{french}`, `\selectlanguage{francais}`, or `\selectlanguage{frenchb}` switches to the French language with the following effects:

1. French hyphenation patterns are made active;
2. 'double punctuation' is made active for correct spacing in French;
3. `\today` prints the date in French;
4. the caption names are translated into French (LaTeX only);
5. the list items are set to '–' instead of • (LaTeX only);
6. the vertical spacing in lists is shortened (LaTeX only);
7. the first paragraph of each section is indented (LaTeX only);
8. French quotation marks can be typeset using the commands `\og` and `\fg` which work in LaTeX 2ε, LaTeX-2.09 and PlainTeX, their appearance depending on what is available to draw them; if you use LaTeX 2ε with `T1`-encoding you can also enter them as `<<~French quotation marks~>>` but then *don't forget* the unbreakable spaces, (`\og` and `\fg` provide for correct line breaks);
9. a command `\up` is provided to typeset superscripts like `M\up{me}` (abbreviation for "Madame"), `1\up{er}` (for "premier");
10. family names should be typeset in small capitals and never be hyphenated, the macro `\bsc` (boxed small caps) does this, e.g., `Leslie~\bsc{Lamport}` will produce Leslie LAMPORT;
11. commands `\primo`, `\secundo`, `\tertio` and `\quarto` may be used to enumerate in lists;
12. abbreviations for "Numéro" and "numéro" are obtained via the commands `\No`, `\no`;
13. two commands are provided to typeset abbreviations for "degré": `\degre` prints the raw character and `\degres` should be used to typeset temperatures (e.g., "`20~\degres C`" with an unbreakable space), or for alcohols' strengths (e.g., "`45\degres`" with *no* space in French);

[21]The file described in this section has version number v1.2a and was last revised on 1997/01/11.

14. an command `\nombre` is provided to ease the typesetting of numbers: it works both in text and in math-mode: inputting `\nombre{3141,592653}` will format this number properly according to the current language (French or non-French) [22]. The command `\nombre` is a contribution of Vincent Jalby using ideas of David Carlisle in comma.sty.

All commands previously available in `francais.ldf` have been included in `frenchb.ldf` for compatibility, sometimes with updated definitions.

The `french` package, by Bernard GAULLE, was not designed to run with babel (although the latest versions claim to be babel compatible), but rather as a stand-alone package for the French language. It provides many more functionalities (like `\lettrine`, `\sommaire`…) not available in `frenchb`, at the cost of a much greater complexity and possible incompatibilities with other languages.

As `french` is known to produce the best layout available for French typography, I have borrowed many ideas from Bernard's file. I did my best to help users of both packages (`french` and `frenchb`) to exchange their sources files easily, with one exception which affects the way French quotation marks are entered: `frenchb` uses *macros* (`\og` and `\fg`) while `french` uses active characters (`<<` and `>>`).

French typographic rules specify that some white space should be present before 'double punctuation' characters. These characters are `; ! ?` and `:`. In order to get this white space automatically, the category code of these characters is made `\active`. In French, the user *should* input these four characters preceded with a space, but as many people forget about it (even among native French writers!), the default behaviour of `frenchb` is to automatically add a `\thinspace` before '`;`' '`!`' '`?`' and a normal (unbreakable) space before '`:`' (this is the rule in French typography). It's up to the user to add or not a space *after* 'double punctuation' characters: usually a space is necessary, but not always (before a full point or a closing brace for instance), so this cannot done automatically.

In (rare) cases where no space should be added before a 'double punctuation', either use `\string; \string: \string! \string?` instead of `; : ! ?`, or switch locally to `english`. For instance you can type `C\string:TEX` or `\begin{otherlanguage}{english}{C:TEX}\end{otherlanguage}` to avoid the space before `:` in a MS-DOS path.

Some users dislike this automatic insertion of a space before 'double punctuation', and prefer to decide themselves whether a space should be added or not; so a hook `\NoAutoSpaceBeforeFDP` is provided: if this command is added (in file `frenchb.cfg`, or anywhere in a document) `frenchb` will respect your typing, and introduce a suitable space before 'double punctuation' *if and only if* a space is typed in the source file before those signs.

The command `\AutoSpaceBeforeFDP` switches back to the default behavior of `frenchb`.

Once you have built your format, a good precaution would be to perform some basic tests about hyphenation in French. For LaTeX 2ε I suggest this:

[22] In math-mode the comma has to be surrounded with braces to avoid a spurious space being inserted after it (see the TeXbook p. 134). Besides this, each slice of three digits should be separated either with a comma in English or with a space in French.

- run the following file, with the encoding suitable for your machine (*my-encoding* will be `latin1` for UNIX machines and PCs running Windows, `applemac` for Macintoshs, or `cp850` for PCs running DOS. If you are using MlTEX together with CMR fonts, comment out the line `\usepackage[`*my-encoding*`]{inputenc}`.

  ```
  %%% Test file for French hyphenation.
  \documentclass{article}
  \usepackage[my-encoding]{inputenc}
  \usepackage[francais]{babel}
  \begin{document}
  \showhyphens{signal, container, \'ev\'enement, alg\`ebre}
  \showhyphens{signal, container,événement, algèbre}
  \end{document}
  ```

- check the hyphenations proposed by TEX in your log-file; in French you should get with both 7-bit and 8-bit encodings
  `si-gnal, contai-ner, évé-ne-ment, al-gèbre`.
  Do not care about how accented characters are displayed in the log-file, what matters is the position of the '`-`' hyphen signs *only*.

If they are all correct, your installation (probably) works fine, if one (or more) is (are) wrong, ask a local wizard to see what's going wrong and perform the test again (or e-mail me about what happens).
Frequent mismatches:

- you get `sig-nal, con-tainer`, this probably means that the hyphenation patterns you are using are for USenglish, not for French;
- you get no hyphen at all in `évé-ne-ment`, this probably means that you are using CMR fonts and the macro `\accent` to produce accented characters. Consider switching to DC/EC fonts and T1-encoding or use MlTEX.

`frenchb` has been improved using helpful suggestions from many people, the main contributions came from Vincent Jalby. Thanks to all of them!

First version released: 1.1 as of 1996/05/31 part of babel-3.6beta.
Changes in version 1.1b: update for babel-3.6.
Changes in version 1.2: new command `\nombre` to format numbers; removed command `\fup` borrowed from the `french` package (`\up` does a better job in LaTeX 2ε); also removed aliases `\french` and `\english` (frenchb.cfg is a better place for these).

24.2 TEXnical details

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

```
⟨*code⟩
%% Please report errors to: Daniel Flipo, GUTenberg
%%                          Daniel.Flipo@univ-lille1.fr
%%
\LdfInit{frenchb}\NoAutoSpaceBeforeFDP
```

Check if hyphenation patterns for the French language have been loaded in language.dat: requested name 'french' or 'francais'.

```
\ifx\l@french\@undefined
  \ifx\l@francais\@undefined
    \@nopatterns{French}
    \adddialect\l@french0
  \fi
\fi
```

Regardless of `\CurrentOption` the internal name for the French language will be 'frenchb'; 'francais' and 'french' will be synonymous for 'frenchb': first let both names use the same hyphenation patterns. Later we will have to set aliases for `\captionsfrenchb`, `\datefrenchb`, `\extrasfrenchb` and `\noextrasfrenchb`. As French uses the standard values of `\lefthyphenmin` (2) and `\righthyphenmin` (3), no special setting is required here.

```
\def\CurrentOption{frenchb}
\ifx\l@francais\@undefined
  \let\l@francais\l@french
\else
  \let\l@french\l@francais
\fi
\let\l@frenchb\l@french
```

To check the format in use (plain or LaTeX), we'll need macros to hold the names of the plain and $\LaTeX\,2_\varepsilon$ formats.

```
\def\PlainFmtName{plain}
\def\LaTeXeFmtName{LaTeX2e}
```

`\if@Two@E` We will need a new 'if' : `\if@Two@E` is true if and only if $\LaTeX\,2_\varepsilon$ is running *not* in compatibility mode. It is used in the definitions of the command `\nombre` and `\up`. The definition is somewhat complicated, due to the fact that `\if@compatibility` is not recognized as a `\if` in LaTeX-2.09 based formats.

```
\newif\if@Two@E \@Two@Etrue
\def\@FI@{\fi}
\ifx\@compatibilitytrue\@undefined
  \@Two@Efalse \def\@FI@{\relax}
\else
  \if@compatibility \@Two@Efalse \fi
\@FI@
```

`\extrasfrenchb`
`\noextrasfrenchb` The macro `\extrasfrenchb` will perform all the extra definitions needed for the French language. The macro `\noextrasfrenchb` is used to cancel the actions of `\extrasfrenchb`.

In French "apostrophe" is used in hyphenation in expressions like `l'ambulance` (French patterns provide entries for this kind of words). This means that the `\lccode` of "apostrophe" has to be non null in French for proper hyphenation of those expressions, and to be reset to null when exiting French.

```
\@namedef{extras\CurrentOption}{\lccode`\'=`\'}
\@namedef{noextras\CurrentOption}{\lccode`\'=0}
\def\extrasfrancais{\extrasfrenchb}
\def\extrasfrench{\extrasfrenchb}
\def\noextrasfrancais{\noextrasfrenchb}
\def\noextrasfrench{\noextrasfrenchb}
```

It is best to use LaTeX 2ε's font changing commands, and to emulated those we need when they are not available, as in PlainTeX or LaTeX-2.09. Be aware that old commands `\sc`, `\it`, *etc.* exist in LaTeX 2ε, but they behave like they did in LaTeX-2.09 (i. e., they switch back to `\normalfont` instead of keeping the other font attributes unchanged).

```
\ifx\scshape\@undefined
  \ifx\sc\@undefined
     \let\scshape\relax
  \else
     \let\scshape\sc
  \fi
\fi
\ifx\emph\@undefined
  \ifx\em\@undefined
     \let\emph\relax
  \else
     \def\emph#1{\em #1}
  \fi
\fi
```

24.3 Captionnames and date

The next step consists of defining the French equivalents for the LaTeX captionnames.

`\captionsfrenchb` The macro `\captionsfrenchb` defines all strings used in the four standard document classes provided with LaTeX. Some authors do not like some of these names; it is easy to change them in the preamble *after* loading `frenchb` (or in your file `frenchb.cfg`), e.g `\addto\captionsfrenchb{\def\figurename{Figure}}` will print 'Figure' in roman instead of 'Fig.'.

```
\ifx\fmtname\PlainFmtName
\else
\@namedef{captions\CurrentOption}{%
   \def\refname{R\'ef\'erences}%
   \def\abstractname{R\'esum\'e}%
   \def\bibname{Bibliographie}%
   \def\prefacename{Pr\'eface}%
   \def\chaptername{Chapitre}%
   \def\appendixname{Annexe}%
   \def\contentsname{Table des mati\`eres}%
   \def\listfigurename{Table des figures}%
   \def\listtablename{Liste des tableaux}%
   \def\indexname{Index}%
   \def\figurename{{\scshape Fig.}}%
   \def\tablename{{\scshape Tab.}}%
```

"Première partie" instead of "Part I"

```
   \def\partname{\protect\@Fpt partie}%
   \def\@Fpt{{\ifcase\value{part}\or Premi\`ere\or Deuxi\`eme\or
   Troisi\`eme\or Quatri\`eme\or Cinqui\`eme\or Sixi\`eme\or
   Septi\`eme\or Huiti\`eme\or Neuvi\`eme\or Dixi\`eme\or Onzi\`eme\or
   Douzi\`eme\or Treizi\`eme\or Quatorzi\`eme\or Quinzi\`eme\or
   Seizi\`eme\or Dix-septi\`eme\or Dix-huiti\`eme\or Dix-neuvi\`eme\or
```

```
  Vingti\`eme\fi}\space\def\thepart{}}%
  \def\pagename{page}%
  \def\seename{{\emph{voir}}}%
  \def\alsoname{{\emph{voir aussi}}}%
  \def\enclname{P.~J. }%
  \def\ccname{Copie \`a }%
  \def\headtoname{}%
  \def\proofname{D\'emonstration}% for AMS-\LaTeX
  }
  \def\captionsfrench{\captionsfrenchb}
  \def\captionsfrancais{\captionsfrenchb}
\fi
```

\datefrenchb The macro \datefrenchb redefines the command \today to produce French dates.

```
\@namedef{date\CurrentOption}{%
 \def\today{\number\day
   \ifnum1=\day \ier\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 novembre\or
   d\'ecembre\fi
   \space \number\year}}
\def\datefrench{\datefrenchb}
\def\datefrancais{\datefrenchb}
```

24.4 Punctuation

The 'double punctuation' characters (`;` `!` `?` and `:`) have to be made `\active` for an automatic control of the amount of space to insert before them.

```
\initiate@active@char{:}
\initiate@active@char{;}
\initiate@active@char{!}
\initiate@active@char{?}
```

We specify that the French group of shorthands should be used.

```
\expandafter\addto\csname extras\CurrentOption\endcsname{%
 \languageshorthands{frenchb}}
```

These characters are 'turned on' once, later their definition may vary.

```
\expandafter\addto\csname extras\CurrentOption\endcsname{%
 \bbl@activate{:}\bbl@activate{;}%
 \bbl@activate{!}\bbl@activate{?}}
\expandafter\addto\csname noextras\CurrentOption\endcsname{%
 \bbl@deactivate{:}\bbl@deactivate{;}%
 \bbl@deactivate{!}\bbl@deactivate{?}}
```

One more thing `\extrasfrenchb` needs to do is to make sure that `\frenchspacing` is in effect. If this is not the case the execution of `\noextrasfrenchb` will switch it off again.

```
\expandafter\addto\csname extras\CurrentOption\endcsname{%
 \bbl@frenchspacing}
\expandafter\addto\csname noextras\CurrentOption\endcsname{%
 \bbl@nonfrenchspacing}
```

`\frenchb@sh@;@` We have to tune the amount of white space before `;` `!` `?` and `:`. This should only happen in horizontal mode, hence the test `\ifhmode`. In horizontal mode, if a space has been typed before ';' we remove it and put an unbreakable `\thinspace` instead. If no space has been typed, we add `\FDP@thinspace` which will be defined, up to the user's wishes, as an automatic added thinspace, or as `\@empty`.

```
\declare@shorthand{frenchb}{;}{%
   \ifhmode
     \ifdim\lastskip>\z@
       \unskip\penalty\@M\thinspace
     \else
       \FDP@thinspace
     \fi
   \fi
```

Now we can insert a `;` character.

```
   \string;}
```

`\frenchb@sh@!@` `\frenchb@sh@?@` Because these definitions are very similar only one is displayed in a way that the definition can be easily checked.

```
\declare@shorthand{frenchb}{!}{%
   \ifhmode
     \ifdim\lastskip>\z@
       \unskip\penalty\@M\thinspace
     \else
       \FDP@thinspace
     \fi
   \fi
   \string!}
```

```
\declare@shorthand{frenchb}{?}{%
   \ifhmode
     \ifdim\lastskip>\z@
       \unskip\penalty\@M\thinspace
     \else
       \FDP@thinspace
     \fi
   \fi
   \string?}
```

`\frenchb@sh@:@` The ':' requires a normal space before it, instead of a `\thinspace`.

```
\declare@shorthand{frenchb}{:}{%
   \ifhmode
     \ifdim\lastskip>\z@
       \unskip\penalty\@M\
     \else
       \FDP@space
     \fi
   \fi
   \string:}
```

`\AutoSpaceBeforeFDP` `\NoAutoSpaceBeforeFDP` `\FDP@thinspace` and `\FDP@space` are defined as unbreakable spaces by `\AutoSpaceBeforeFDP` or as `\@empty` by `\NoAutoSpaceBeforeFDP`.
Default is `\AutoSpaceBeforeFDP`.

```
\def\AutoSpaceBeforeFDP{%
      \def\FDP@thinspace{\penalty\@M\thinspace}%
      \def\FDP@space{\penalty\@M\ }}
\def\NoAutoSpaceBeforeFDP{\let\FDP@thinspace\@empty
                          \let\FDP@space\@empty}
\AutoSpaceBeforeFDP
```

`\system@sh@:@` `\system@sh@!@` `\system@sh@?@` `\system@sh@;@`

When the active characters appear in an environment where their French behaviour is not wanted they should give an 'expected' result. Therefore we define shorthands at system level as well.

```
\declare@shorthand{system}{:}{\string:}
\declare@shorthand{system}{!}{\string!}
\declare@shorthand{system}{?}{\string?}
\declare@shorthand{system}{;}{\string;}
```

24.5 French quotation marks

Several shapes of French quotation marks are provided for use with CMR or EC/DC fonts, or PostScript fonts. CMR fonts have no quotation marks built-in, so we have to emulate them using math symbols, either LaTeX's 'lasy' font if available, or TeX symbols `\ll` and `\gg` otherwise. EC/DC fonts and PostScript fonts have built-in quotation marks, so we will of course use them.
The following definitions will take care for correct spacing of French quotation marks (a white space precedes and follows quotation marks but no line break is allowed neither *after* the opening one, nor *before* the closing one).

`\oPlainGuill` `\fPlainGuill`

For *Plain*TeX, we define `\oPlainGuill` and `\fPlainGuill` using math symbols `\ll` and `\gg`. In order to have the word following opening guillemets hyphenated properly we have to end the definitions with the TeX equivalent for `\allowhyphens` which is `\penalty\@M\hskip\z@skip`.

```
\def\oPlainGuill{\leavevmode\raise0.25ex%
                 \hbox{$\scriptscriptstyle\ll$\kern 0.15em}%
                 \penalty\@M\hskip\z@skip}
\def\fPlainGuill{\ifdim\lastskip>\z@\unskip\penalty\@M\fi
                 \leavevmode\raise0.25ex%
                 \hbox{\kern 0.15em$\scriptscriptstyle\gg$}}
```

`\oLasyGuill` `\fLasyGuill`

In LaTeX 2_ε better looking quotation marks are available via the 'lasy' font ('lasy' stands for LaTeX Symbol).

```
\ifx\fmtname\LaTeXeFmtName
  \def\oLasyGuill{\leavevmode
                  \hbox{\fontencoding{U}\fontfamily{lasy}\selectfont
                        (\kern-0.20em(\kern 0.20em}\allowhyphens}
  \def\fLasyGuill{\ifdim\lastskip>\z@\unskip\penalty\@M\fi\leavevmode
                  \hbox{\kern0.20em%
                        \fontencoding{U}\fontfamily{lasy}\selectfont
                        )\kern-0.20em)}}
\fi
```

`\oECGuill` `\fECGuill`

Now let's define French quotation marks for T1 encoding.

```
\def\oECGuill{\leavevmode\hbox{\guillemotleft\kern 0.15em}%
              \allowhyphens}
```

```
\def\fECGuill{\ifdim\lastskip>\z@\unskip\penalty\@M\fi
              \leavevmode\hbox{\kern 0.15em\guillemotright}}
```

`\og` `\fg` `\bbl@frenchguillemets` `\bbl@nonfrenchguillemets`

Now let's define which kind of French quotation marks will be used. The top macros for quotation marks will be called `\og` ("o̲uvrez g̲uillemets") and `\fg` ("f̲ermez g̲uillemets").
Make the top level definitions for French quotation marks available through the `\extrasfrenchb` `\noextrasfrenchb` mechanism.
As `\DeclareTextCommand` cannot be used after the `\begin{document}` we introduce internal definitions `\begin@guill` and `\end@guill`.
We'll try to be smart to users of D. Carlisle's `xspace` package: if this package is loaded there will be no need for `{}` or `\ ` to get a space after `\fg`.
In LaTeX 2_ε we provide a dummy definition for `\og` and `\fg`, just to display an error message in case `\og` or `\fg` have been defined elsewhere.

```
\ifx\fmtname\LaTeXeFmtName
   \newcommand{\og}{\@empty}
   \newcommand{\fg}{\@empty}
   \DeclareTextCommand{\begin@guill}{T1}{\oECGuill}
   \DeclareTextCommand{\end@guill}{T1}{\fECGuill}
   \DeclareTextCommand{\begin@guill}{OT1}{\oLasyGuill}
   \DeclareTextCommand{\end@guill}{OT1}{\fLasyGuill}
   \DeclareTextSymbolDefault{\begin@guill}{OT1}
   \DeclareTextSymbolDefault{\end@guill}{OT1}
\else
   \let\begin@guill\oPlainGuill
   \let\end@guill\fPlainGuill
\fi
\def\bbl@frenchguillemets{\ifx\xspace\@undefined\let\xspace\relax\fi
                          \def\og{\begin@guill}%
                          \def\fg{\end@guill\xspace}}
\def\bbl@nonfrenchguillemets{\def\og{``}%
                             \def\fg{\ifdim\lastskip>\z@\unskip\fi ''}}
\expandafter\addto\csname extras\CurrentOption\endcsname{%
  \bbl@frenchguillemets}
\expandafter\addto\csname noextras\CurrentOption\endcsname{%
  \bbl@nonfrenchguillemets}
```

24.6 French lists

`\bbl@frenchitems` `\bbl@nonfrenchitems` `\bbl@frenchlistspacing` `\bbl@nonfrenchlistspacing`

French lists are different from USenglish ones: the • is never used (long dash '–' is prefered for all levels), and vertical spacing between items, before and after the list, should be shorter in French texts than the defaults provided by LaTeX. Note that the easy way, just changing values of vertical spacing parameters when entering French and restoring them to their defaults on exit would not work, so we have to redefine `\@trivlist`.
The amount of vertical space before and after a list is given by `\topsep` + `\parskip` (+ `\partopsep` if the list starts a new paragraph). IMHO, `\parskip` should be added *only* when the list starts a new paragraph, so I subtract `\parskip` from `\topsep` and add it back to `\partopsep`; this will normally make no difference because `\parskip`'s default value is 0pt, but will be noticeable when `\parskip` is *not* null.

I would appreciate feedback from experts in French typography, about the (somewhat experimental) values set here for `\partopsep`, `\topsep`, `\itemsep` and `\parsep`.
Of course, this code is only for LaTeX.

```
\ifx\fmtname\PlainFmtName
\else
  \let\@ltiORI\labelitemi
  \let\@ltiiORI\labelitemii
  \let\@ltiiiORI\labelitemiii
  \let\@ltivORI\labelitemiv
  \def\bbl@frenchitems{%
    \def\labelitemi{--}%
    \def\labelitemii{--}%
    \def\labelitemiii{--}%
    \def\labelitemiv{--}}
  \def\bbl@nonfrenchitems{%
    \let\labelitemi\@ltiORI
    \let\labelitemii\@ltiiORI
    \let\labelitemiii\@ltiiiORI
    \let\labelitemiv\@ltivORI}
  \expandafter\addto\csname extras\CurrentOption\endcsname{%
    \bbl@frenchitems}
  \expandafter\addto\csname noextras\CurrentOption\endcsname{%
    \bbl@nonfrenchitems}
  \let\@trivlistORI\@trivlist
  \def\bbl@frenchlistspacing{%
    \def\@trivlist{\setlength{\itemsep}{0.4ex plus 0.2ex minus 0.2ex}%
                   \setlength{\parsep}{0.4ex plus 0.2ex minus 0.2ex}%
                   \setlength{\topsep}{0.8ex plus 0.4ex minus 0.4ex}%
                   \setlength{\partopsep}{0.4ex plus 0.2ex minus 0.2ex}%
                   \addtolength{\topsep}{-\parskip}%
                   \addtolength{\partopsep}{\parskip}%
                   \@trivlistORI}}
  \def\bbl@nonfrenchlistspacing{\let\@trivlist\@trivlistORI}
  \expandafter\addto\csname extras\CurrentOption\endcsname{%
    \bbl@frenchlistspacing}
  \expandafter\addto\csname noextras\CurrentOption\endcsname{%
    \bbl@nonfrenchlistspacing}
\fi
```

24.7 French indentation of sections

`\bbl@frenchindent`
`\bbl@nonfrenchindent`

In French the first paragraph of each section should be indented, this is another difference with USenglish. Add this code only in LaTeX.

```
\ifx\fmtname\PlainFmtName
\else
  \let\@aifORI\@afterindentfalse
  \def\bbl@frenchindent{\let\@afterindentfalse\@afterindenttrue
                        \@afterindenttrue}
  \def\bbl@nonfrenchindent{\let\@afterindentfalse\@aifORI
                           \@afterindentfalse}
  \expandafter\addto\csname extras\CurrentOption\endcsname{%
    \bbl@frenchindent}
  \expandafter\addto\csname noextras\CurrentOption\endcsname{%
    \bbl@nonfrenchindent}
\fi
```

24.8 Formatting numbers

In English the decimal part starts with a point and thousands should be separated by a comma: an approximation of 1000π should be inputed as `$3{,}141.592{,}653$` in math-mode and as `3,141.592,653` in text. In French the decimal part starts with a comma and thousands should be separated by a space; the same approximation of 1000π should be inputed as `$3\;141{,}592\;653$` in math-mode and as something like `3~141,592~653` in text. Remember braces are mandatory around the comma in math-mode, the reason is mentioned in the TeXbook p. 134: the comma is of type `\mathpunct` (thus normally followed by a space) while the point is of type `\mathord` (no space added).

Thierry Bouche suggested that a second type of comma, of type `\mathord` would be useful in math-mode, and proposed to introduce a command (named `\decimalsep` in this package), the expansion of which would depend on the current language.

Vincent Jalby suggested a command `\nombre` to conveniently typeset numbers: inputting `\nombre{3141,592653}` either in text or in math-mode will format this number properly according to the current language (French or non-French).

`\nombre` accepts an optional argument which happens to be useful with the extension ‘dcolumn’, it specifies the decimal separator used in the *source code*:

```
\newcolumntype{d}{D{,}{\decimalsep}{-1}}
\begin{tabular}{d}\hline
  3,14 \\
  \nombre[,]{123,4567} \\
  \nombre[,]{9876,543}\\\hline
\end{tabular}
```

will print a column of numbers aligned on the decimal point (comma or point depending on the current language), each slice of 3 digits being separated by a space or a comma according to the current language.

`\decimalsep` `\thousandsep` We need a internal definition, valid in both text and math-mode, for the comma (`\@comma@`) and another one for the unbreakable fixed length space (no glue) used in French (`\f@thousandsep`).

The commands `\decimalsep` and `\thousandsep` get default definitions (for the English language) when `frenchb` is loaded; these definitions will be updated when the current language is switched to or from French.

```
\mathchardef\m@comma="013B
\def\@comma@{\ifmmode\m@comma\else,\fi}
\def\f@thousandsep{\ifmmode\mskip5.5mu\else\penalty\@M\kern.3em\fi}
\newcommand{\decimalsep}{.}
\newcommand{\thousandsep}{\@comma@}
\expandafter\addto\csname extras\CurrentOption\endcsname{%
            \def\decimalsep{\@comma@}%
            \def\thousandsep{\f@thousandsep}}
\expandafter\addto\csname noextras\CurrentOption\endcsname{%
            \def\decimalsep{.}%
            \def\thousandsep{\@comma@}}
```

`\nombre` The decimal separator used when *inputing* a number with `\nombre` *has to be a comma.* `\nombre` splits the inputed number into two parts: what comes before the first comma will be formatted by `\@integerpart` while the rest (if not empty) will be formatted by `\@decimalpart`. Both parts, once formatted separately will be merged together with between them, either the decimal separator `\decimalsep` or (in LaTeX 2_ε *only*) the optional argument of `\nombre`.

```
\if@Two@E
  \newcommand{\nombre}[2][\decimalsep]{%
         \def\@decimalsep{#1}\@nombre#2\@empty,\@empty,\@nil}
\else
  \newcommand{\nombre}[1]{%
         \def\@decimalsep{\decimalsep}\@nombre#1\@empty,\@empty,\@nil}
\fi
\def\@nombre#1,#2,#3\@nil{%
       \ifx\@empty#2%
         \@integerpart{#1}%
       \else
         \@integerpart{#1}\@decimalsep\@decimalpart{#2}%
       \fi}
```

The easiest bit is the decimal part: We attempt to read the first four digits of the decimal part, if it has less than 4 digits, we just have to print them, otherwise `\thousandsep` has to be appended after the third digit, and the algorithm is applied recursively to the rest of the decimal part.

```
\def\@decimalpart#1{\@@decimalpart#1\@empty\@empty\@empty}
\def\@@decimalpart#1#2#3#4{#1#2#3%
  \ifx\@empty#4%
  \else
    \thousandsep\expandafter\@@decimalpart\expandafter#4%
  \fi}
```

Formatting the integer part is more difficult because the slices of 3 digits start from the *bottom* while the number is read from the top! This (tricky) code is borrowed from David Carlisle's comma.sty.

```
\def\@integerpart#1{\@@integerpart{}#1\@empty\@empty\@empty}
\def\@@integerpart#1#2#3#4{%
  \ifx\@empty#2%
    \@addthousandsep#1\relax
  \else
    \ifx\@empty#3%
      \@addthousandsep\@empty\@empty#1#2\relax
    \else
      \ifx\@empty#4%
        \@addthousandsep\@empty#1#2#3\relax
      \else
        \@@integerpartafterfi{#1#2#3#4}%
      \fi
    \fi
  \fi}
\def\@@integerpartafterfi#1\fi\fi\fi{\fi\fi\fi\@@integerpart{#1}}
\def\@addthousandsep#1#2#3#4{#1#2#3%
  \if#4\relax
  \else
```

```
    \thousandsep\expandafter\@addthousandsep\expandafter#4%
  \fi}
```

24.9 Extra utilities

All that is left to do now is to provide the French user with some extra utilities.

\up \ieme \up eases the typesetting of superscripts like ‘1$^{\text{er}}$’. \up relies on \textsuperscript when available (i. e., in LaTeX 2$_\varepsilon$).

\up@size The internal macro \up@size holds the size at which the superscript will be typeset. The reason for this is that we have to specify it differently for different formats.

```
\ifx\sevenrm\@undefined
  \ifx\@ptsize\@undefined
    \let\up@size\small
  \else
    \ifx\selectfont\@undefined
```

In this case the format is the original LaTeX-2.09:

```
      \ifcase\@ptsize
        \let\up@size\ixpt\or
        \let\up@size\xpt\or
        \let\up@size\xipt
      \fi
```

When \selectfont is defined we probably have NFSS available:

```
    \else
      \ifcase\@ptsize
        \def\up@size{\fontsize\@ixpt{10pt}\selectfont}\or
        \def\up@size{\fontsize\@xpt{11pt}\selectfont}\or
        \def\up@size{\fontsize\@xipt{12pt}\selectfont}
      \fi
    \fi
  \fi
\else
```

If we end up here it must be a plain based TeX format, so:

```
    \let\up@size\sevenrm
\fi
```

Now we can define \up. When LaTeX 2$_\varepsilon$ runs in compatibility mode (LaTeX-2.09 emulation), \textsuperscript is also defined, but does no good job, so we give two different definitions for \up using \if@Two@E.

```
\if@Two@E
  \DeclareRobustCommand*{\up}[1]{\textsuperscript{#1}}
\else
  \DeclareRobustCommand*{\up}[1]{\leavevmode\raise1ex\hbox{\up@size#1}}
\fi
```

\ieme is provided for compatibility with francais.sty, the other 5 for compatibility with french.sty:

```
\def\ieme{\up{\lowercase{e}}}
\def\iemes{\up{\lowercase{es}}}
\def\ier{\up{\lowercase{er}}}
```

```
\def\iers{\up{\lowercase{ers}}}
\def\iere{\up{\lowercase{re}}}
\def\ieres{\up{\lowercase{res}}}
```

\No \no \primo \fprimo)

And some more macros for numbering, first two support macros.

```
\DeclareRobustCommand*{\FrenchEnumerate}[1]{%
                          #1\up{\lowercase{o}}\kern+.3em}
\DeclareRobustCommand*{\FrenchPopularEnumerate}[1]{%
                          #1\up{\lowercase{o}})\kern+.3em}
```

Typing `\primo` should result in '1^{o} ',

```
\def\primo{\FrenchEnumerate1}
\def\secundo{\FrenchEnumerate2}
\def\tertio{\FrenchEnumerate3}
\def\quatro{\FrenchEnumerate4}
```

while typing `\fprimo)` gives '1^{o}) .

```
\def\fprimo){\FrenchPopularEnumerate1}
\def\fsecundo){\FrenchPopularEnumerate2}
\def\ftertio){\FrenchPopularEnumerate3}
\def\fquatro){\FrenchPopularEnumerate4}
```

Let's provide two macros for the common abbreviations of "Numéro".

```
\DeclareRobustCommand*{\No}{N\up{\lowercase{o}}\kern+.2em}
\DeclareRobustCommand*{\no}{n\up{\lowercase{o}}\kern+.2em}
```

\bsc

As family names should be written in small capitals and never be hyphenated, we provide a command (its name comes from Boxed Small Caps) to input them easily; this is a simpler implementation of commands `\fsc` and `\lsc` from `french.sty` : no automatic uppercase/lowercase conversion is performed. Usage: `Jean~\bsc{Duchemin}`.

```
\DeclareRobustCommand*{\bsc}[1]{\leavevmode\hbox{\scshape #1}}
```

Some definitions for special characters. The first eight are mandatory for `\oe` etc. to work properly in moving arguments, the others just for convenience. We won't define `\tilde` as a Text Symbol not to conflict with the macro `\tilde` for math-mode and use the name `\tild` instead. Note that `\boi` may *not* be used in math-mode, its name in math-mode is `\backslash`. `\degre` needs a special treatment: it is `\char6` in T1-encoding and `\char23` in OT1-encoding.

```
\ifx\fmtname\LaTeXeFmtName
  \DeclareTextSymbol{\ae}{T1}{230}
  \DeclareTextSymbol{\ae}{OT1}{26}
  \DeclareTextSymbol{\oe}{T1}{247}
  \DeclareTextSymbol{\oe}{OT1}{27}
  \DeclareTextSymbol{\AE}{T1}{198}
  \DeclareTextSymbol{\AE}{OT1}{29}
  \DeclareTextSymbol{\OE}{T1}{215}
  \DeclareTextSymbol{\OE}{OT1}{30}
  \DeclareTextSymbol{\degre}{T1}{6}
  \DeclareTextSymbol{\degre}{OT1}{23}
  \DeclareTextSymbol{\boi}{T1}{92}
  \DeclareTextCommand{\boi}{OT1}{{$\backslash$}}
  \DeclareTextSymbol{\at}{T1}{64}
  \DeclareTextSymbol{\at}{OT1}{64}
  \DeclareTextSymbol{\circonflexe}{T1}{94}
  \DeclareTextSymbol{\circonflexe}{OT1}{94}
  \DeclareTextSymbol{\tild}{T1}{126}
  \DeclareTextSymbol{\tild}{OT1}{126}
\else
  \def\T@one{T1}
  \ifx\f@encoding\T@one
    \newcommand{\degre}{\char6}
  \else
    \newcommand{\degre}{\char23}
  \fi
  \newcommand{\at}{\char64}
  \newcommand{\circonflexe}{\char94}
  \newcommand{\tild}{\char126}
  \newcommand{\boi}{{$\backslash$}}
\fi
```

`\degres` Macro for typesetting the abbreviation for 'degrees' (as in 'degrees Celsius'). As the bounding box of the character 'degree' has *very* different widths in CMR/DC and PostScript fonts, we fix the width of the bounding box of `\degres` to 0.3 em, this lets the symbol 'degree' stick to the preceding (e.g., `45\degres`) or following character (e.g., `20~\degres C`).

```
\DeclareRobustCommand*{\degres}{%
                            \leavevmode\hbox to 0.3em{\hss\degre\hss}}
```

The following macros are used in the redefinition of `\^` and `\"` to handle the letter i: they allow users to type simply `\^i` and `\"i` instead of `\^{\i}` and `\"{\i}`. MlTEX's macros dealing with accents conflict with those of LaTeX 2$_\varepsilon$, so we check whether `\csubinverse` is defined or not. If `\csubinverse` is *defined*, we are in MlTEX.

```
\ifx\fmtname\LaTeXeFmtName
  \AtBeginDocument{%
    \ifx\csubinverse\@undefined
      \DeclareTextCompositeCommand{\^}{OT1}{i}{\^\i}%
      \DeclareTextCompositeCommand{\"}{OT1}{i}{\"\i}%
    \fi}
\fi
```

Finally the macrospace used by some control sequences we do not need any longer, is freed.

```
\let\T@one\relax
\let\@FI@\relax
\let\PlainFmtName\relax
\let\LaTeXeFmtName\relax
```

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. The config file searched for will always be 'frenchb.cfg'. Remember that `\CurrentOption` has been set to 'frenchb', and that 'francais' and 'french' are aliases for 'frenchb'.

```
\ldf@finish\CurrentOption
⟨/code⟩
```

25 The Italian language

The file `italian.dtx`[23] It defines all the language-specific macros for the Italian language.

For this language the `\clubpenalty`, `\widowpenalty` and `\finalhyphendemerits` are set to rather high values.

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

```
⟨*code⟩
\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.

```
\ifx\l@italian\@undefined
    \@nopatterns{Italian}
    \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 documentclasses 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 abbreviation is preferred
  \def\seename{vedi}%
  \def\alsoname{vedi anche}%
  \def\proofname{Dimostrazione}%
  }
```

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

```
\def\dateitalian{%
\def\today{\number\day~\ifcase\month\or
```

[23] The file described in this section has version number v1.2j and was last revised on 1996/12/29. The original author is Maurizio Codogno, (`mau@beatles.cselt.stet.it`).

```
  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.

```
\def\italianhyphenmins{\tw@\tw@}
```

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

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

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 "begl'italiani" we give the `'` a non-zero lower case code. When we do that TEX finds the following hyphenation points `be-gl'i-ta-lia-ni` instead of none.

```
\addto\extrasitalian{%
  \lccode`'=`'}
\addto\noextrasitalian{%
  \lccode`'=0}
```

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⟩
```

26 The Portuguese language

The file `portuges.dtx`[24] defines all the language-specific macros for the Portuguese language as well as for the Brasilian version of this language.

For this language the character `"` is made active. In table 6 an overview is given of its purpose.

`"|`	disable ligature at this position.
`"-`	an explicit hyphen sign, allowing hyphenation in the rest of the word.
`""`	like `"-`, but producing no hyphen sign (for words that should break at some sign such as "entrada/salida."
`"<`	for French left double quotes (similar to <<).
`">`	for French right double quotes (similar to >>).
`\-`	like the old `\-`, but allowing hyphenation in the rest of the word.

Table 6: The extra definitions made by `portuges.ldf`

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

```
⟨*code⟩
\LdfInit\CurrentOption{captions\CurrentOption}
```

When this file is read as an option, i.e. by the `\usepackage` command, `portuges` will be an 'unknown' language in which case we have to make it known. So we check for the existence of `\l@portuges` to see whether we have to do something here. Since it is possible to load this file with any of the following four options to babel: portuges, portuguese, brazil and brazilian we also allow that the hyphenation patterns are loaded under any of these four names. We just have to find out which one was used.

```
\ifx\l@portuges\@undefined
  \ifx\l@portuguese\@undefined
    \ifx\l@brazil\@undefined
      \ifx\l@brazilian\@undefined
        \@nopatterns{Portuguese}
        \adddialect\l@portuges0
      \else
        \let\l@portuges\l@brazilian
      \fi
    \else
      \let\l@portuges\l@brazil
    \fi
  \else
    \let\l@portuges\portuguese
  \fi
\fi
```

By now `\l@portuges` is defined. When the language definition file was loaded under a different name we make sure that the hyphenation patterns can be found.

[24]The file described in this section has version number v1.2j and was last revised on 1996/12/23. Contributions were made by Jose Pedro Ramalhete (`JRAMALHE@CERNVM` or `Jose-Pedro_Ramalhete@MACMAIL`) and Arnaldo Viegas de Lima `arnaldo@VNET.IBM.COM`.

```
\expandafter\ifx\csname l@\CurrentOption\endcsname\relax
  \expandafter\let\csname l@\CurrentOption\endcsname\l@portuges
\fi
```

Now we have to decide whether this language definition file was loaded for Portuguese or Brasilian use. This can be done by checking the contents of `\CurrentOption`. When it doesn't contain either 'portuges' or 'portuguese' we make `\bbl@tempa` empty.

```
\def\bbl@tempa{portuges}
\ifx\CurrentOption\bbl@tempa
  \let\bbl@tempb\@empty
\else
  \def\bbl@tempa{portuguese}
  \ifx\CurrentOption\bbl@tempa
    \let\bbl@tempb\@empty
  \else
    \def\bbl@tempb{brazil}
  \fi
\fi
\ifx\bbl@tempb\@empty
```

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

`\captionsportuges` The macro `\captionsportuges` defines all strings used in the four standard documentclasses provided with LaTeX.

```
  \@namedef{captions\CurrentOption}{%
    \def\prefacename{Pref\'acio}%
    \def\refname{Refer\^encias}%
    \def\abstractname{Resumo}%
    \def\bibname{Bibliografia}%
    \def\chaptername{Cap\'{\i}tulo}%
    \def\appendixname{Ap\^endice}%
```

Some discussion took place around the correct translations for 'Table of Contents' and 'Index'. the translations differ for Portuguese and Brasilian based the following history:

> The whole issue is that some books without a real index at the end misused the term 'Índice' as table of contents. Then, what happens is that some books apeared with 'Índice' at the begining and a 'Índice Remissivo' at the end. Remissivo is a redundant word in this case, but was introduced to make up the difference. So in Brasil people started using 'Sumário' and 'Índice Remissivo'. In Portugal this seems not to be very common, therefore we chose 'Índice' instead of 'Índice Remissivo'.

```
    \def\contentsname{Conte\'udo}%
    \def\listfigurename{Lista de Figuras}%
    \def\listtablename{Lista de Tabelas}%
    \def\indexname{\'Indice}%
    \def\figurename{Figura}%
    \def\tablename{Tabela}%
    \def\partname{Parte}%
    \def\enclname{Anexo}%
    \def\ccname{Com c\'opia a}%
    \def\headtoname{Para}%
    \def\pagename{P\'agina}%
    \def\seename{ver}%
    \def\alsoname{ver tamb\'em}%
```

An alternate term for ‘Proof’ could be ‘Prova’.

```
    \def\proofname{Demonstra\c{c}\~ao}%
    }
```

\dateportuges The macro `\dateportuges` redefines the command `\today` to produce Portuguese dates.

```
  \@namedef{date\CurrentOption}{%
    \def\today{\number\day\space de\space\ifcase\month\or
      Janeiro\or Fevereiro\or Mar\c{c}o\or Abril\or Maio\or Junho\or
      Julho\or Agosto\or Setembro\or Outubro\or Novembro\or Dezembro
      \fi
      \space de\space\number\year}}
\else
```

For the Brasilian version of these definitions we just add a “dialect”.

```
  \expandafter\adddialect\csname l@\CurrentOption\endcsname\l@portuges
```

\captionsbrazil The “captions” are different for both versions of the language, so we define the macro `\captionsbrazil` here.

```
  \@namedef{captions\CurrentOption}{%
    \def\prefacename{Pref\'acio}%
    \def\refname{Refer\^encias}%
    \def\abstractname{Resumo}%
    \def\bibname{Refer\^encias Bibliogr\'aficas}%
    \def\chaptername{Cap\'{\i}tulo}%
    \def\appendixname{Ap\^endice}%
    \def\contentsname{Sum\'ario}%
    \def\listfigurename{Lista de Figuras}%
    \def\listtablename{Lista de Tabelas}%
    \def\indexname{\'Indice Remissivo}%
    \def\figurename{Figura}%
    \def\tablename{Tabela}%
    \def\partname{Parte}%
    \def\enclname{Anexo}%
    \def\ccname{C\'opia para}%
    \def\headtoname{Para}%
    \def\pagename{P\'agina}%
    \def\seename{veja}%
    \def\alsoname{veja tamb\'em}%
    \def\proofname{Demonstra\c{c}\~ao}%
    }
```

\datebrazil The macro `\datebrazil` redefines the command `\today` to produce Brasilian dates, for which the names of the months are not capitalized.

```
  \@namedef{date\CurrentOption}{%
    \def\today{\number\day\space de\space\ifcase\month\or
      janeiro\or fevereiro\or mar\c{c}o\or abril\or maio\or junho\or
      julho\or agosto\or setembro\or outubro\or novembro\or dezembro
      \fi
      \space de\space\number\year}}
\fi
```

\portugeshyphenmins

Set correct values for \lefthyphenmin and \righthyphenmin.

```
\@namedef{\CurrentOption hyphenmins}{\tw@\tw@}
```

\extrasportuges
\noextrasportuges

The macro \extrasportuges will perform all the extra definitions needed for the Portuguese language. The macro \noextrasportuges is used to cancel the actions of \extrasportuges.

For Portuguese the " character is made active. This is done once, later on its definition may vary. Other languages in the same document may also use the " character for shorthands; we specify that the portuguese group of shorthands should be used.

```
\initiate@active@char{"}
\@namedef{extras\CurrentOption}{\languageshorthands{portuges}}
\expandafter\addto\csname extras\CurrentOption\endcsname{%
  \bbl@activate{"}}
%\addto\noextrasportuges{\bbl@deactivate{"}}
```

First we define access to the guillemets for quotations,

```
\declare@shorthand{portuges}{"<}{%
  \textormath{\guillemotleft{}}{\mbox{\guillemotleft}}}
\declare@shorthand{portuges}{">}{%
  \textormath{\guillemotright{}}{\mbox{\guillemotright}}}
```

then we define two shorthands to be able to specify hyphenation breakpoints that behavew a little different from \-.

```
\declare@shorthand{portuges}{"-}{\allowhyphens-\allowhyphens}
\declare@shorthand{portuges}{""}{\hskip\z@skip}
```

And we want to have a shorthand for disabling a ligature.

```
\declare@shorthand{portuges}{"|}{%
  \textormath{\discretionary{-}{}{\kern.03em}}{}}
```

\-

All that is left now is the redefinition of \-. The new version of \- should indicate an extra hyphenation position, while allowing other hyphenation positions to be generated automatically. The standard behaviour of TEX in this respect is very unfortunate for languages such as Dutch and German, where long compound words are quite normal and all one needs is a means to indicate an extra hyphenation position on top of the ones that TEX can generate from the hyphenation patterns.

```
\expandafter\addto\csname extras\CurrentOption\endcsname{%
  \babel@save\-}
\expandafter\addto\csname extras\CurrentOption\endcsname{%
  \def\-{\allowhyphens\discretionary{-}{}{}\allowhyphens}}
```

\ord
\ro
\orda
\ra

We also provide an easy way to typeset ordinals, both in the male (\ord or \ro) and the female (orda or \ra) form.

```
\def\ord{$^{\rm o}$}
\def\orda{$^{\rm a}$}
\let\ro\ord\let\ra\orda
```

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.

```
26.114 \ldf@finish\CurrentOption
26.115 ⟨/code⟩
```

27 The Spanish language

The file `spanish.dtx`[25] defines all the language definition macro's for the Spanish[26] language.

This file[27] incorporates the result of discussions held in the Spanish-TEX[28] electronic mail list.

For this language the characters `'` `~` and `"` are made active. In table 7 an overview is given of their purpose. These active accent characters behave according

`'a`	an accent that allows hyphenation. Valid for all vowels uppercase and lowercase.
`'n`	a n with a tilde. This is included to improve compatibility with FTC. Works for uppercase too.
`"|`	disable ligature at this position.
`"-`	an explicit hyphen sign, allowing hyphenation in the rest of the word.
`""`	like `"-`, but producing no hyphen sign (for words that should break at some sign such as "entrada/salida."
`\-`	like the old `\-`, but allowing hyphenation in the rest of the word.
`"u`	a u with dieresis allowing hyphenation.
`"a`	feminine ordinal as in $1^{\underline{a}}$.
`"o`	masculine ordinal as in $1^{\underline{o}}$.
`"<`	for French left double quotes (similar to <<).
`">`	for French right double quotes (similar to >>).
`~n`	a n with tilde. Works for uppercase too.

Table 7: The extra definitions made by `spanish.ldf`

to their original definitions if not followed by one of the characters indicated in that table.

This option includes support for working with extended, 8-bit fonts, if available. Old versions of this file based this support on the existance of special macros with names as in Ferguson's ML-TEX. This is no longer the case. Support is now based on providing an appropriate definition for the accent macros on entry to the Spanish language. This is automatically done by LaTeX 2ε or NFSS2. If T1 encoding is chosen, and provided that adequate hyphenation patterns[29] exist, it

[25]The file described in this section has version number v3.4g and was last revised on 1997/01/15. The original author is Julio Sánchez, (`jsanchez@gmv.es`).

[26]Catalan used to be part of this file but is now on its own file.

[27]In writing this file, many ideas and actual coding solutions have been taken from a number of sources. The language definition files `dutch.sty` and `germanb.sty` are the main contributors and are not explicitly mentioned in the sequel. J. L. Braams and Bernd Raichle have given helpful advice. Another source of inspiration is the experience gained in the use of FTC, a software package written by José A. Mañas. The members of the Spanish-TEX list have helped clarify a number of issues. Other sources are explicitly acknowledged when used. If you think that you contributed something and you are not mentioned, please let me (`jsanchez@gmv.es`) know. I humbly apologize for any omission.

[28]`spanish-tex@goya.eunet.es`, subscription requests can be sent to the address `listserv@goya.eunet.es`. This list is devoted to discussions on support in TEX for Spanish. Comments on this language option are welcome there or directly to `jsanchez@gmv.es`.

[29]One source for such patterns is the archive at `ftp.eunet.es` that can be accessed by anonymous FTP or electronic mail to `ftpmail@goya.eunet.es`. They are in the `info` direc-

is possible to get better hyphenation for Spanish than before. The easiest way to use the new encoding with LaTeX 2ε to load the package `t1enc` with `\usepackage`. This must be done before loading `babel`.

If the combination of keyboard and TeX version that the user has is able to produce the accented characters in the T1 enconding, the user could see the accented characters in the editor, greatly improving the readability of the document source. As of today, this is not a recommended method for producing documents for distribution, although it is possible to mechanically translate the document so that the receiver can make use of it. If care is taken to define the encoding needed by the document, the results are pretty portable.

This option file will automatically detect if the T1 encoding is being used and behave appropriately. If any other encoding is being used, the accent macros will be redefined to allow hyphenation on the accented words.

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

```
⟨*code⟩
\LdfInit{spanish}\captionsspanish
```

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

```
\ifx\l@spanish\@undefined
  \@nopatterns{Spanish}
  \adddialect\l@spanish0
\fi
```

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

`\captionsspanish` The macro `\captionsspanish` defines all strings[30] used in the four standard documentclasses provided with LaTeX.

```
\addto\captionsspanish{%
  \def\prefacename{Prefacio}%
  \def\refname{Referencias}%
  \def\abstractname{Resumen}%
  \def\bibname{Bibliograf\'{\i}a}%
  \def\chaptername{Cap\'{\i}tulo}%
  \def\appendixname{Ap\'endice}%
  \def\contentsname{\'Indice General}%
  \def\listfigurename{\'Indice de Figuras}%
  \def\listtablename{\'Indice de Tablas}%
  \def\indexname{\'Indice de Materias}%
  \def\figurename{Figura}%
  \def\tablename{Tabla}%
  \def\partname{Parte}%
  \def\enclname{Adjunto}%
```

tory `src/TeX/spanish`. The list of Frequently Asked Questions with Answers about TeX for Spanish is kept there as well. That list is meant to be a summary of the discussions held in the Spanish-TeX mail list. Warning: It is in Spanish.

[30] The accent on the uppercase 'I' is intentional, following the recommendation of the *Real Academia de la Lengua* in *Esbozo de una Nueva Gramática de la Lengua Española, Comisión de Gramática, Espasa-Calpe, 1973.*

```
\def\ccname{Copia a}%
\def\headtoname{A}%
\def\pagename{P\'agina}%
\def\seename{v\'ease}%
\def\alsoname{v\'ease tambi\'en}%
\def\proofname{Demostraci\'on}%
}%
```

`\datespanish` The macro `\datespanish` redefines the command `\today` to produce Spanish[31] dates.

```
\def\datespanish{%
\def\today{\number\day~de\space\ifcase\month\or
  enero\or febrero\or marzo\or abril\or mayo\or junio\or
  julio\or agosto\or septiembre\or octubre\or noviembre\or diciembre\fi
  \space de~\number\year}}
```

`\extrasspanish` `\noextrasspanish` The macro `\extrasspanish` will perform all the extra definitions needed for the Spanish language. The macro `\noextrasspanish` is used to cancel the actions of `\extrasspanish`. For Spanish, some characters are made active or are redefined. In particular, the `"` character, the `'` character and the `~` character receive new meanings. Therefore these characters have to be treated as 'special' characters.

```
\addto\extrasspanish{\languageshorthands{spanish}}
\initiate@active@char{"}
\initiate@active@char{~}
\addto\extrasspanish{%
  \bbl@activate{"}%
  \bbl@activate{~}}

\@ifpackagewith{babel}{activeacute}{%
  \initiate@active@char{'}}{}
\@ifpackagewith{babel}{activeacute}{%
  \addto\extrasspanish{\bbl@activate{'}}}{}
%\addto\noextrasspanish{
%  \bbl@deactivate{"}\bbl@deactivate{~}\bbl@deactivate{'}}
```

Apart from the active characters some other macros get a new definition. Therefore we store the current one to be able to restore them later.

```
\addto\extrasspanish{%
  \babel@save\"%
  \babel@save\~%
  \def\"{\protect\@umlaut}%
  \def\~{\protect\@tilde}}
\@ifpackagewith{babel}{activeacute}{%
  \babel@save\'
  \addto\extrasspanish{\def\'{\protect\@acute}}
  }{}
```

`\spanishhyphenmins` Spanish hyphenation uses `\lefthyphenmin` and `\righthyphenmin` both set to 2.

```
\def\spanishhyphenmins{\tw@\tw@}
```

[31]Months are written lowercased. This has been cause of some controversy. This file follows *Diccionario de Uso de la Lengua Española, María Moliner, 1990,* that is in agreement with the most common practice.

`\dieresis` `\textacute` `\texttilde` The original definition of `\"` is stored as `\dieresis`, because the we do not know what is its definition, since it depends on the encoding we are using or on special macros that the user might have loaded. The expansion of the macro might use the TEX `\accent` primitive using some particular accent that the font provides or might check if a combined accent exists in the font. These two cases happen with respectively OT1 and T1 encodings. For this reason we save the definition of `\"` and use that in the definition of other macros. We do likewise for `\'` and `\~`. The present coding of this option file is incorrect in that it can break when the encoding changes. We do not use `\acute` or `\tilde` as the macro names because they are already defined as `\mathaccent`.

```
\let\dieresis\"
\let\texttilde\~
\@ifpackagewith{babel}{activeacute}{\let\textacute\'}{}
```

`\@umlaut` `\@acute` `\@tilde` We check the encoding and if not using T1, we make the accents expand but enabling hyphenation beyond the accent. If this is the case, not all break positions will be found in words that contain accents, but this is a limitation in TEX. An unsolved problem here is that the encoding can change at any time. The definitions below are made in such a way that a change between two 256-char encodings are supported, but changes between a 128-char and a 256-char encoding are not properly supported. We check if T1 is in use. If not, we will give a warning and proceed redefining the accent macros so that TEX at least finds the breaks that are not too close to the accent. The warning will only be printed to the log file.

```
\ifx\DeclareFontShape\@undefined
  \wlog{Warning: You are using an old LaTeX}
  \wlog{Some word breaks will not be found.}
  \def\@umlaut#1{\allowhyphens\dieresis{#1}\allowhyphens}
  \def\@tilde#1{\allowhyphens\texttilde{#1}\allowhyphens}
  \@ifpackagewith{babel}{activeacute}{%
    \def\@acute#1{\allowhyphens\textacute{#1}\allowhyphens}}{}
\else
  \edef\next{T1}
  \ifx\f@encoding\next
    \let\@umlaut\dieresis
    \let\@tilde\texttilde
    \@ifpackagewith{babel}{activeacute}{%
      \let\@acute\textacute}{}
  \else
    \wlog{Warning: You are using encoding \f@encoding\space
      instead of T1.}
    \wlog{Some word breaks will not be found.}
    \def\@umlaut#1{\allowhyphens\dieresis{#1}\allowhyphens}
    \def\@tilde#1{\allowhyphens\texttilde{#1}\allowhyphens}
    \@ifpackagewith{babel}{activeacute}{%
      \def\@acute#1{\allowhyphens\textacute{#1}\allowhyphens}}{}
  \fi
\fi
```

Now we can define our shorthands: the umlauts,

```
\declare@shorthand{spanish}{"u}{\@umlaut u}
\declare@shorthand{spanish}{"U}{\@umlaut U}
```

french quotes,

```
\declare@shorthand{spanish}{"<}{%
  \textormath{\guillemotleft{}}{\mbox{\guillemotleft}}}
\declare@shorthand{spanish}{">}{%
  \textormath{\guillemotright{}}{\mbox{\guillemotright}}}
```

ordinals[32],

```
\declare@shorthand{spanish}{"o}{%
  \leavevmode\raise1ex\hbox{\underbar{\scriptsize o}}}
\declare@shorthand{spanish}{"a}{%
  \leavevmode\raise1ex\hbox{\underbar{\scriptsize a}}}
```

acute accents,

```
\@ifpackagewith{babel}{activeacute}{%
  \declare@shorthand{spanish}{'a}{\textormath{\@acute a}{^{\prime} a}}
  \declare@shorthand{spanish}{'e}{\textormath{\@acute e}{^{\prime} e}}
  \declare@shorthand{spanish}{'i}{\textormath{\@acute\i{}}{^{\prime}i}}
  \declare@shorthand{spanish}{'o}{\textormath{\@acute o}{^{\prime} o}}
  \declare@shorthand{spanish}{'u}{\textormath{\@acute u}{^{\prime} u}}
  \declare@shorthand{spanish}{'A}{\textormath{\@acute A}{^{\prime} A}}
  \declare@shorthand{spanish}{'E}{\textormath{\@acute E}{^{\prime} E}}
  \declare@shorthand{spanish}{'I}{\textormath{\@acute I}{^{\prime} I}}
  \declare@shorthand{spanish}{'O}{\textormath{\@acute O}{^{\prime} O}}
  \declare@shorthand{spanish}{'U}{\textormath{\@acute U}{^{\prime} U}}
```

the acute accent,

```
  \declare@shorthand{spanish}{''}{%
    \textormath{\textquotedblright}{\sp\bgroup\prim@s'}}
```

tildes,

```
  \declare@shorthand{spanish}{'n}{\textormath{\~n}{^{\prime} n}}
  \declare@shorthand{spanish}{'N}{\textormath{\~N}{^{\prime} N}}
  }{}
\declare@shorthand{spanish}{~n}{\textormath{\~n}{\@tilde n}}
\declare@shorthand{spanish}{~N}{\textormath{\~N}{\@tilde N}}
```

and some additional commands:

```
\declare@shorthand{spanish}{"-}{\allowhyphens\-\allowhyphens}
\declare@shorthand{spanish}{"|}{%
  \textormath{\penalty\@M\discretionary{-}{}{\kern.03em}%
              \allowhyphens}{}}
\declare@shorthand{spanish}{""}{\hskip\z@skip}
```

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{spanish}
⟨/code⟩
```

[32]The code for the ordinals was taken from the answer provided by Raymond Chen (`raymond@math.berkeley.edu`) to a question by Joseph Gil (`yogi@cs.ubc.ca`) in `comp.text.tex`.

28 The Catalan language

The file `catalan.dtx`[33] defines all the language-specific macro's for the Catalan language.

For this language only the double quote character (`"`) is made active by default. In table 8 an overview is given of the new macros defined and the new meanings of `"`. Additionally to that, the user can explicitly activate the acute accent or apostrophe (`'`) and/or the grave accent (`` ` ``) characters by using the activeacute and activegrave options. In that case, the definitions shown in table 9 also become available[34].

`\lgem`	geminated-l digraph (similar to l·l). `\Lgem` produces the uppercase version.
`\up`	Macro to help typing raised ordinals, like 1$^{\text{er}}$. Takes one argument.
`\-`	like the old `\-`, but allowing hyphenation in the rest of the word.
`"i`	i with diaeresis, allowing hyphenation in the rest of the word. Valid for the following vowels: i, u (both lowercase and uppercase).
`"c`	c-cedilla (ç). Valid for both uppercase and lowercase c.
`"l`	geminated-l digraph (similar to l·l). Valid for both uppercase and lowercase l.
`"<`	French left double quotes (similar to <<).
`">`	French right double quotes (similar to >>).
`"-`	explicit hyphen sign, allowing hyphenation in the rest of the word.
`"|`	disable ligature at this position.

Table 8: Extra definitions made by file `catalan.ldf` (activated by default)

`'e`	acute accented a, allowing hyphenation in the rest of the word. Valid for the following vowels: e, i, o, u (both lowercase and uppercase).
`` `a ``	grave accented a, allowing hyphenation in the rest of the word. Valid for the following vowels: a, e, o (both lowercase and uppercase).

Table 9: Extra definitions made by file `catalan.ldf` (activated only when using the options activeacute and activegrave)

These active accents characters behave according to their original definitions if not followed by one of the characters indicated in that table.

[33]The file described in this section has version number v2.2h and was last revised on 1997/01/08.

[34]Please note that if the acute accent character is active, it is necessary to take special care of coding apostrophes in a way which cannot be confounded with accents. Therefore, it is necessary to type `l'{}estri` instead of `l'estri`.

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

```
⟨*code⟩
\LdfInit{catalan}\captionscatalan
```

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

```
\ifx\l@catalan\@undefined
  \@nopatterns{Catalan}
  \adddialect\l@catalan0
\fi
```

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

`\captionscatalan` The macro `\captionscatalan` defines all strings used in the four standard documentclasses provided with LaTeX.

```
\addto\captionscatalan{%
  \def\prefacename{Pr\`oleg}%
  \def\refname{Refer\`encies}%
  \def\abstractname{Resum}%
  \def\bibname{Bibliografia}%
  \def\chaptername{Cap\'{\i}tol}%
  \def\appendixname{Ap\`endix}%
  \def\contentsname{\'Index}%
  \def\listfigurename{\'Index de figures}%
  \def\listtablename{\'Index de taules}%
  \def\indexname{\'Index alfab\`etic}%
  \def\figurename{Figura}%
  \def\tablename{Taula}%
  \def\partname{Part}%
  \def\enclname{Adjunt}%
  \def\ccname{C\`opies a}%
  \def\headtoname{A}%
  \def\pagename{P\`agina}%
  \def\seename{Vegeu}%
  \def\alsoname{Vegeu tamb\'e}%
  \def\proofname{Demostraci\'o}%
}
```

`\datecatalan` The macro `\datecatalan` redefines the command `\today` to produce Catalan dates. Months are written in lowercase[35].

```
\def\datecatalan{%
  \def\today{\number\day~\ifcase\month\or
    de gener\or de febrer\or de mar\c{c}\or d'abril\or de maig\or
    de juny\or de juliol\or d'agost\or de setembre\or d'octubre\or
    de novembre\or de desembre\fi
    \space de~\number\year}}
```

[35]This seems to be the common practice. See for example: E. Coromina, *El 9 Nou: Manual de redacció i estil*, Ed. Eumo, Vic, 1993

`\extrascatalan` `\noextrascatalan` The macro `\extrascatalan` will perform all the extra definitions needed for the Catalan language. The macro `\noextrascatalan` is used to cancel the actions of `\extrascatalan`.

To improve hyphenation we give the grave character (') a non-zero lower case code; when we do that TeX will find more breakpoints in words that contain this character in its rôle as apostrophe.

```
\addto\extrascatalan{%
  \lccode`'=`'}
\addto\noextrascatalan{%
  \lccode`'=0}
```

For Catalan, some characters are made active or are redefined. In particular, the " character receives a new meaning; this can also happen for the ' character and the ` character when the options activegrave and/or activeacute are specified.

```
\addto\extrascatalan{\languageshorthands{catalan}}
\initiate@active@char{"}
\addto\extrascatalan{\bbl@activate{"}}
\@ifpackagewith{babel}{activegrave}{%
  \initiate@active@char{`}}{}
\@ifpackagewith{babel}{activegrave}{%
  \addto\extrascatalan{\bbl@activate{`}}}{}
\@ifpackagewith{babel}{activeacute}{%
  \initiate@active@char{'}}{}
\@ifpackagewith{babel}{activeacute}{%
  \addto\extrascatalan{\bbl@activate{'}}}{}
%\addto\noextrascatalan{%
%  \bbl@deactivate{"}
%  \bbl@deactivate{`}\bbl@deactivate{'}}
```

Apart from the active characters some other macros get a new definition. Therefore we store the current ones to be able to restore them later. When their current meanings are saved, we can safely redefine them.

We provide new definitions for the accent macros when one or boith of the options activegrave or activeacute were specified.

```
\addto\extrascatalan{%
  \babel@save\"%
  \def\"{\protect\@umlaut}}%
\@ifpackagewith{babel}{activegrave}{%
  \babel@save\`%
  \addto\extrascatalan{\def\`{\protect\@grave}}
  }{}
\@ifpackagewith{babel}{activeacute}{%
  \babel@save\'%
  \addto\extrascatalan{\def\'{\protect\@acute}}
  }{}
```

All the code above is necessary because we need a few extra active characters. These characters are then used as indicated in tables 8 and 9.

`\dieresis` `\textacute` `\textgrave` The original definition of `\"` is stored as `\dieresis`, because the definition of `\"` might not be the default plain TeX one. If the user uses PostScript fonts with the Adobe font encoding the " character is not in the same position as in Knuth's font encoding. In this case `\"` will not be defined as `\accent"7F 1`, but as `\accent'310 #1`. Something similar happens when using fonts that follow the Cork encoding. For this reason we save the definition of `\"` and use that in the definition of other macros. We do likewise for `` \` ``, and `\'`.

```
\let\dieresis\"
\@ifpackagewith{babel}{activegrave}{\let\textgrave\`}{}
\@ifpackagewith{babel}{activeacute}{\let\textacute\'}{}
```

`\@umlaut` `\@acute` `\@grave` We check the encoding and if not using T1, we make the accents expand but enabling hyphenation beyond the accent. If this is the case, not all break positions will be found in words that contain accents, but this is a limitation in TEX. An unsolved problem here is that the encoding can change at any time. The definitions below are made in such a way that a change between two 256-char encodings are supported, but changes between a 128-char and a 256-char encoding are not properly supported. We check if T1 is in use. If not, we will give a warning and proceed redefining the accent macros so that TEX at least finds the breaks that are not too close to the accent. The warning will only be printed to the log file.

```
\ifx\DeclareFontShape\@undefined
  \wlog{Warning: You are using an old LaTeX}
  \wlog{Some word breaks will not be found.}
  \def\@umlaut#1{\allowhyphens\dieresis{#1}\allowhyphens}
  \@ifpackagewith{babel}{activeacute}{%
    \def\@acute#1{\allowhyphens\textacute{#1}\allowhyphens}}{}
  \@ifpackagewith{babel}{activegrave}{%
    \def\@grave#1{\allowhyphens\textgrave{#1}\allowhyphens}}{}
\else
  \edef\next{T1}
  \ifx\f@encoding\next
    \let\@umlaut\dieresis
    \@ifpackagewith{babel}{activeacute}{%
      \let\@acute\textacute}{}
    \@ifpackagewith{babel}{activegrave}{%
      \let\@grave\textgrave}{}
  \else
    \wlog{Warning: You are using encoding \f@encoding\space
      instead of T1.}
    \wlog{Some word breaks will not be found.}
    \def\@umlaut#1{\allowhyphens\dieresis{#1}\allowhyphens}
    \@ifpackagewith{babel}{activeacute}{%
      \def\@acute#1{\allowhyphens\textacute{#1}\allowhyphens}}{}
    \@ifpackagewith{babel}{activegrave}{%
      \def\@grave#1{\allowhyphens\textgrave{#1}\allowhyphens}}{}
  \fi
\fi
```

If the user setup has extended fonts, the Ferguson macros are required to be defined. We check for their existance and, if defined, expand to whatever they are defined to. For instance, `\'a` would check for the existance of a `\@ac@a` macro. It is assumed to expand to the code of the accented letter. If it is not defined, we assume that no extended codes are available and expand to the original definition but enabling hyphenation beyond the accent. This is as best as we can do. It is better if you have extended fonts or ML-TEX because the hyphenation algorithm can work on the whole word. The following macros are directly derived from

ML-TeX.[36]

Now we can define our shorthands: the diaeresis and “ela geminada” support,

```
\declare@shorthand{catalan}{"i}{\textormath{\@umlaut\i}{\ddot\imath}}
\declare@shorthand{catalan}{"l}{\lgem{}}
\declare@shorthand{catalan}{"u}{\textormath{\@umlaut u}{\ddot u}}
\declare@shorthand{catalan}{"I}{\textormath{\@umlaut I}{\ddot I}}
\declare@shorthand{catalan}{"L}{\Lgem{}}
\declare@shorthand{catalan}{"U}{\textormath{\@umlaut U}{\ddot U}}
```

cedille,

```
\declare@shorthand{catalan}{"c}{\textormath{\c c}{^{\prime} c}}
\declare@shorthand{catalan}{"C}{\textormath{\c C}{^{\prime} C}}
```

‘french’ quote characters,

```
\declare@shorthand{catalan}{"<}{%
  \textormath{\guillemotleft{}}{\mbox{\guillemotleft}}}
\declare@shorthand{catalan}{">}{%
  \textormath{\guillemotright{}}{\mbox{\guillemotright}}}
```

grave accents,

```
\@ifpackagewith{babel}{activegrave}{%
  \declare@shorthand{catalan}{`a}{\textormath{\@grave a}{\grave a}}
  \declare@shorthand{catalan}{`e}{\textormath{\@grave e}{\grave e}}
  \declare@shorthand{catalan}{`o}{\textormath{\@grave o}{\grave o}}
  \declare@shorthand{catalan}{`A}{\textormath{\@grave A}{\grave A}}
  \declare@shorthand{catalan}{`E}{\textormath{\@grave E}{\grave E}}
  \declare@shorthand{catalan}{`O}{\textormath{\@grave O}{\grave O}}
  \declare@shorthand{catalan}{``}{\textquotedblleft}
  }{}
```

acute accents,

```
\@ifpackagewith{babel}{activeacute}{%
  \declare@shorthand{catalan}{'a}{\textormath{\@acute a}{^{\prime} a}}
  \declare@shorthand{catalan}{'e}{\textormath{\@acute e}{^{\prime} e}}
  \declare@shorthand{catalan}{'i}{\textormath{\@acute\i{}}{^{\prime} i}}
  \declare@shorthand{catalan}{'o}{\textormath{\@acute o}{^{\prime} o}}
  \declare@shorthand{catalan}{'u}{\textormath{\@acute u}{^{\prime} u}}
  \declare@shorthand{catalan}{'A}{\textormath{\@acute A}{^{\prime} A}}
  \declare@shorthand{catalan}{'E}{\textormath{\@acute E}{^{\prime} E}}
  \declare@shorthand{catalan}{'I}{\textormath{\@acute I}{^{\prime} I}}
  \declare@shorthand{catalan}{'O}{\textormath{\@acute O}{^{\prime} O}}
  \declare@shorthand{catalan}{'U}{\textormath{\@acute U}{^{\prime} U}}
  \declare@shorthand{catalan}{'|}{%
    \textormath{\csname normal@char\string'\endcsname}{^{\prime}}}
```

the acute accent,

```
  \declare@shorthand{catalan}{''}{%
    \textormath{\textquotedblright}{\sp\bgroup\prim@s'}}
  }{}
```

[36]A problem is perceived here with these macros when used in a multilingual environment where extended hyphenation patterns are available for some but not all languages. Assume that no extended patterns exist at some site for French and that `french.sty` would adopt this scheme too. In that case, `'e` in French would produce the combined accented letter, but hyphenation around it would be suppressed. Both language options would need an independent method to know whether they have extended patterns available. The precise impact of this problem and the possible solutions are under study.

and finally, some support definitions

```
\declare@shorthand{catalan}{"-}{\allowhyphens-\allowhyphens}
\declare@shorthand{catalan}{"|}{%
  \textormath{\penalty\@M\discretionary{-}{}{\kern.03em}%
              \allowhyphens}{}}
```

`\-` All that is left now is the redefinition of `\-`. The new version of `\-` should indicate an extra hyphenation position, while allowing other hyphenation positions to be generated automatically. The standard behaviour of TEX in this respect is unfortunate for Catalan but not as much as for Dutch or German, where long compound words are quite normal and all one needs is a means to indicate an extra hyphenation position on top of the ones that TEX can generate from the hyphenation patterns. However, the average length of words in Catalan makes this desirable and so it is kept here.

```
\addto\extrascatalan{%
  \babel@save{\-}%
  \def\-{\allowhyphens\discretionary{-}{}{}\allowhyphens}}
```

`\lgem` `\Lgem` Here we define a macro for typing the catalan “ela geminada” (geminated l). The macros `\lgem` and `\Lgem` have been chosen for its lowercase and uppercase representation, respectively[37].

The code used in the actual macro used is a combination of the one proposed by Feruglio and Fuster[38] and the proposal[39] from Valiente presented at the TEX Users Group Annual Meeting in 1995. This last proposal has not been fully implemented due to its limitation to CM fonts.

```
\newdimen\leftllkern \newdimen\rightllkern \newdimen\raiselldim
\def\lgem{%
  \ifmmode
    \csname normal@char\string"\endcsname l%
  \else
    \leftllkern=0pt\rightllkern=0pt\raiselldim=0pt%
    \setbox0\hbox{l}\setbox1\hbox{l\/}\setbox2\hbox{.}%
    \advance\raiselldim by \the\fontdimen5\the\font
    \advance\raiselldim by -\ht2%
    \leftllkern=-.25\wd0%
    \advance\leftllkern by \wd1%
    \advance\leftllkern by -\wd0%
    \rightllkern=-.25\wd0%
    \advance\rightllkern by -\wd1%
    \advance\rightllkern by \wd0%
    \allowhyphens\discretionary{l-}{l}%
    {\hbox{l}\kern\leftllkern\raise\raiselldim\hbox{.}%
      \kern\rightllkern\hbox{l}}\allowhyphens
  \fi
  }
\def\Lgem{%
  \ifmmode
    \csname normal@char\string"\endcsname L%
```

[37]The macro names `\ll` and `\LL` were not taken because of the fact that `\ll` is already used in mathematical mode.

[38]G. Valiente and R. Fuster, Typesetting Catalan Texts with TEX, *TUGboat* **14**(3), 1993.

[39]G. Valiente, Modern Catalan Typographical Conventions, *TUGboat* **16**(3), 1995.

```
\else
  \leftllkern=0pt\rightllkern=0pt\raiselldim=0pt%
  \setbox0\hbox{L}\setbox1\hbox{L\/}\setbox2\hbox{.}%
  \advance\raiselldim by .5\ht0%
  \advance\raiselldim by -.5\ht2%
  \leftllkern=-.125\wd0%
  \advance\leftllkern by \wd1%
  \advance\leftllkern by -\wd0%
  \rightllkern=-\wd0%
  \divide\rightllkern by 6%
  \advance\rightllkern by -\wd1%
  \advance\rightllkern by \wd0%
  \allowhyphens\discretionary{L-}{L}%
  {\hbox{L}\kern\leftllkern\raise\raiselldim\hbox{.}%
    \kern\rightllkern\hbox{L}}\allowhyphens
\fi
}
```

`\l.l` `\L.L` It seems to be the most natural way of entering the "ela geminda" to use the sequences `\l.l` and `\L.L`. These are not really macro's by themselves but the macros `\l` and `\L` with delimited arguments. Therefor we define two macros that check if the next character is a period. If not the "polish l" will be typeset, otherwise a "ela geminada" will be typeset and the next two tokens will be 'eaten'.

```
\let\lslash\l
\let\Lslash\L
\DeclareRobustCommand\l{\@ifnextchar.\bbl@l\lslash}
\DeclareRobustCommand\L{\@ifnextchar.\bbl@L\Lslash}
\def\bbl@l#1#2{\lgem}
\def\bbl@L#1#2{\Lgem}
```

`\up` A macro for typesetting things like 1^{er} as proposed by Raymon Seroul[40].

```
\DeclareRobustCommand*{\up}[1]{\textsuperscript{#1}}
```

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{catalan}
⟨/code⟩
```

[40]This macro has been borrowed from francais.dtx

29 The Galician language

The file `galician.dtx`[41] defines all the language definition macros for the Galician language.

For this language the characters `'` `~` and `"` are made active. In table 10 an overview is given of their purpose. These active accents character behave according

`"|`	disable ligature at this position.
`"-`	an explicit hyphen sign, allowing hyphenation in the rest of the word.
`\-`	like the old `\-`, but allowing hyphenation in the rest of the word.
`'a`	an accent that allows hyphenation. Valid for all vowels uppercase and lowercase.
`'n`	a n with a tilde. This is included to improve compatibility with FTC. Works for uppercase too.
`"u`	a u with dieresis allowing hyphenation.
`"a`	feminine ordinal as in $1^{\underline{a}}$.
`"o`	masculine ordinal as in $1^{\underline{o}}$.
`~n`	a n with tilde. Works for uppercase too.

Table 10: The extra definitions made by `galician.ldf`

to their original definitions if not followed by one of the characters indicated in that table.

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

```
29.1 ⟨*code⟩
29.2 \LdfInit{galician}\captionsgalician
```

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

```
29.3 \ifx\l@galician\@undefined
29.4   \@nopatterns{Galician}
29.5   \adddialect\l@galician0\fi
```

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

`\captionsgalician` The macro `\captionsgalician` defines all strings used in the four standard documentclasses provided with LaTeX.

```
29.6 \addto\captionsgalician{%
29.7   \def\prefacename{Prefacio}%
29.8   \def\refname{Referencias}%
29.9   \def\abstractname{Resumo}%
29.10  \def\bibname{Bibliograf\'{\i}a}%
29.11  \def\chaptername{Cap\'{\i}tulo}%
29.12  \def\appendixname{Ap\'endice}%
```

[41]The file described in this section has version number v1.2g and was last revised on 1997/01/15.

```
  \def\contentsname{\'Indice Xeral}%
  \def\listfigurename{\'Indice de Figuras}%
  \def\listtablename{\'Indice de T\'aboas}%
  \def\indexname{\'Indice de Materias}%
  \def\figurename{Figura}%
  \def\tablename{T\'aboa}%
  \def\partname{Parte}%
  \def\enclname{Adxunto}%
  \def\ccname{Copia a}%
  \def\headtoname{A}%
  \def\pagename{P\'axina}%
  \def\seename{v\'exase}%
  \def\alsoname{v\'exase tam\'en}%
  \def\proofname{Proof}%  <-- Needs Translation!
}
```

`\dategalician` The macro `\dategalician` redefines the command `\today` to produce Galician dates.

```
\def\dategalician{%
  \def\today{\number\day~de\space\ifcase\month\or
    xaneiro\or febreiro\or marzo\or abril\or maio\or xu\~no\or
    xullo\or agosto\or setembro\or outubro\or novembro\or decembro\fi
    \space de~\number\year}}
```

`\extrasgalician` `\noextrasgalician` The macro `\extrasgalician` will perform all the extra definitions needed for the Galician language. The macro `\noextrasgalician` is used to cancel the actions of `\extrasgalician`.

For Galician, some characters are made active or are redefined. In particular, the `"` character and the `~` character receive new meanings this can also happen for the `'` character when the option activeacute is specified.

```
\addto\extrasgalician{\languageshorthands{galician}}
\initiate@active@char{"}
\initiate@active@char{~}
\addto\extrasgalician{%
  \bbl@activate{"}\bbl@activate{~}}
\@ifpackagewith{babel}{activeacute}{%
  \initiate@active@char{'}}{}
\@ifpackagewith{babel}{activeacute}{%
  \addto\extrasgalician{\bbl@activate{'}}}{}
%\addto\noextrasgalician{%
%  \bbl@deactivate{"}\bbl@deactivate{~}\bbl@deactivate{'}}
```

Apart from the active characters some other macros get a new definition. Therefore we store the current one to be able to restore them later.

```
\addto\extrasgalician{%
  \babel@save\"\babel@save\~%
  \def\"{\protect\@umlaut}%
  \def\~{\protect\@tilde}}
\@ifpackagewith{babel}{activeacute}{%
  \babel@save\'%
  \addto\extrasgalician{\def\'{\protect\@acute}}
  }{}
```

All the code above is necessary because we need a few extra active characters. These characters are then used as indicated in table 10.

This option includes some support for working with extended, 8-bit fonts, if available. This assumes that the user has some macros predefined. For instance, if the user has a `\@ac@a` macro defined, the sequence `\'a` or `'a` will both expand to whatever `\@ac@a` is defined to expand, presumably á. The names of these macros are the same as those in Ferguson's ML-TEX compatibility package on purpose. Using this method, and provided that adequate hyphenation patterns exist, it is possible to get better hyphenation for Galician than before. If the user has a terminal able to produce these codes directly, it is possible to do so. If the need arises to send the document to someone who does not have such support, it is possible to mechanically translate the document so that the receiver can make use of it.

To be able to define the function of the new accents, we first define a couple of 'support' macros.

`\dieresis` `\textacute` `\texttilde` The original definition of `\"` is stored as `\dieresis`, because the definition of `\"` might not be the default plain TEX one. If the user uses PostScript fonts with the Adobe font encoding the `"` character is not in the same position as in Knuth's font encoding. In this case `\"` will not be defined as `\accent"7F #1`, but as `\accent'310 #1`. Something similar happens when using fonts that follow the Cork encoding. For this reason we save the definition of `\"` and use that in the definition of other macros. We do likewise for `\'` and `\~`.

```
29.52 \let\dieresis\"
29.53 \let\texttilde\~
29.54 \@ifpackagewith{babel}{activeacute}{\let\textacute\'}{}
```

`\@umlaut` `\@acute` `\@tilde` If the user setup has extended fonts, the Ferguson macros are required to be defined. We check for their existance and, if defined, expand to whatever they are defined to. For instance, `\'a` would check for the existance of a `\@ac@a` macro. It is assumed to expand to the code of the accented letter. If it is not defined, we assume that no extended codes are available and expand to the original definition but enabling hyphenation beyond the accent. This is as best as we can do. It is better if you have extended fonts or ML-TEX because the hyphenation algorithm can work on the whole word. The following macros are directly derived from ML-TEX.[42]

```
29.55 \def\@umlaut#1{\allowhyphens\dieresis{#1}\allowhyphens}
29.56 \def\@tilde#1{\allowhyphens\texttilde{#1}\allowhyphens}
29.57 \@ifpackagewith{babel}{activeacute}{%
29.58   \def\@acute#1{\allowhyphens\textacute{#1}\allowhyphens}}{}
```

Now we can define our shorthands: the umlauts,

```
29.59 \declare@shorthand{galician}{"-}{\allowhyphens-\allowhyphens}
29.60 \declare@shorthand{galician}{"|}{\discretionary{-}{}{\kern.03em}}
```

[42]A problem is perceived here with these macros when used in a multilingual environment where extended hyphenation patterns are available for some but not all languages. Assume that no extended patterns exist at some site for French and that `french.sty` would adopt this scheme too. In that case, `'e` in French would produce the combined accented letter, but hyphenation around it would be suppressed. Both language options would need an independent method to know whether they have extended patterns available. The precise impact of this problem and the possible solutions are under study.

```
\declare@shorthand{galician}{"u}{\@umlaut{u}}
\declare@shorthand{galician}{"U}{\@umlaut{U}}
```

ordinals[43],

```
\declare@shorthand{galician}{"o}{%
  \leavevmode\raise1ex\hbox{\underbar{\scriptsize o}}}
\declare@shorthand{galician}{"a}{%
  \leavevmode\raise1ex\hbox{\underbar{\scriptsize a}}}
```

acute accents,

```
\@ifpackagewith{babel}{activeacute}{%
  \declare@shorthand{galician}{'a}{\textormath{\@acute a}{^{\prime} a}}
  \declare@shorthand{galician}{'e}{\textormath{\@acute e}{^{\prime} e}}
  \declare@shorthand{galician}{'i}{\textormath{\@acute\i{}}{^{\prime}i}}
  \declare@shorthand{galician}{'o}{\textormath{\@acute o}{^{\prime} o}}
  \declare@shorthand{galician}{'u}{\textormath{\@acute u}{^{\prime} u}}
  \declare@shorthand{galician}{'A}{\textormath{\@acute A}{^{\prime} A}}
  \declare@shorthand{galician}{'E}{\textormath{\@acute E}{^{\prime} E}}
  \declare@shorthand{galician}{'I}{\textormath{\@acute I}{^{\prime} I}}
  \declare@shorthand{galician}{'O}{\textormath{\@acute O}{^{\prime} O}}
  \declare@shorthand{galician}{'U}{\textormath{\@acute U}{^{\prime} U}}
```

tildes,

```
  \declare@shorthand{galician}{'n}{\textormath{\~n}{^{\prime} n}}
  \declare@shorthand{galician}{'N}{\textormath{\~N}{^{\prime} N}}
```

the acute accent,

```
  \declare@shorthand{galician}{''}{%
    \textormath{\textquotedblright}{\sp\bgroup\prim@s'}}
  }{}
\declare@shorthand{galician}{~n}{\textormath{\~n}{\@tilde n}}
\declare@shorthand{galician}{~N}{\textormath{\~N}{\@tilde N}}
```

`\-` All that is left now is the redefinition of `\-`. The new version of `\-` should indicate an extra hyphenation position, while allowing other hyphenation positions to be generated automatically. The standard behaviour of TEX in this respect is unfortunate for Galician but not as much as for Dutch or German, where long compound words are quite normal and all one needs is a means to indicate an extra hyphenation position on top of the ones that TEX can generate from the hyphenation patterns. However, the average length of words in Galician makes this desirable and so it is kept here.

```
\addto\extrasgalician{%
  \babel@save{\-}%
  \def\-{\allowhyphens\discretionary{-}{}{}\allowhyphens}}
```

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{galician}
⟨/code⟩
```

[43]The code for the ordinals was taken from the answer provided by Raymond Chen (raymond@math.berkeley.edu) to a question by Joseph Gil (`yogi@cs.ubc.ca`) in `comp.text.tex`.

30 The Romanian language

The file `romanian.dtx`[44] defines all the language-specific macros for the Romanian language.

For this language currently no special definitions are needed or available.

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

```
⟨*code⟩
\LdfInit{romanian}\captionsromanian
```

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

```
\ifx\l@romanian\@undefined
    \@nopatterns{Romanian}
    \adddialect\l@romanian0\fi
```

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

`\captionsromanian` The macro `\captionsromanian` defines all strings used in the four standard documentclasses provided with LaTeX.

```
\addto\captionsromanian{%
  \def\prefacename{Prefa\c{t}\u{a}}%
  \def\refname{Bibliografie}%
  \def\abstractname{Rezumat}%
  \def\bibname{Bibliografie}%
  \def\chaptername{Capitolul}%
  \def\appendixname{Anexa}%
  \def\contentsname{Cuprins}%
  \def\listfigurename{List\u{a} de figuri}%
  \def\listtablename{List\u{a} de tabele}%
  \def\indexname{Glosar}%
  \def\figurename{Figura}%    % sau Plan\c{s}a
  \def\tablename{Tabela}%
  \def\partname{Partea}%
  \def\enclname{Anex\u{a}}%   % sau Anexe
  \def\ccname{Copie}%
  \def\headtoname{Pentru}%
  \def\pagename{Pagina}%
  \def\seename{Vezi}%
  \def\alsoname{Vezi de asemenea}%
  \def\proofname{Demonstra\c{t}ie} %
  }%
```

`\dateromanian` The macro `\dateromanian` redefines the command `\today` to produce Romanian dates.

```
\def\dateromanian{%
\def\today{\number\day~\ifcase\month\or
  ianuarie\or februarie\or martie\or aprilie\or mai\or
```

[44]The file described in this section has version number v1.2h and was last revised on 1996/12/23. A contribution was made by Umstatter Horst (`hhu@cernvm.cern.ch`).

```
  iunie\or iulie\or august\or septembrie\or octombrie\or
  noiembrie\or decembrie\fi
  \space \number\year}}
```

`\extrasromanian` `\noextrasromanian` The macro `\extrasromanian` will perform all the extra definitions needed for the Romanian language. The macro `\noextrasromanian` is used to cancel the actions of `\extrasromanian` For the moment these macros are empty but they are defined for compatibility with the other language definition files.

```
\addto\extrasromanian{}
\addto\noextrasromanian{}
```

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{romanian}
⟨/code⟩
```

31 The Danish language

The file `danish.dtx`[45] defines all the language definition macros for the Danish language.

For this language the character `"` is made active. In table 11 an overview is given of its purpose.

`"|`	disable ligature at this position.
`"-`	an explicit hyphen sign, allowing hyphenation in the rest of the word.
`""`	like `"-`, but producing no hyphen sign (for words that should break at some sign such as "entrada/salida."
``"` ``	lowered double left quotes (looks like „)
`"'`	normal double right quotes
`"<`	for French left double quotes (similar to <<).
`">`	for French right double quotes (similar to >>).

Table 11: The extra definitions made by `danish.ldf`

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

```
⟨*code⟩
\LdfInit{danish}\captionsdanish
```

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

```
\ifx\l@danish\@undefined
    \@nopatterns{Danish}
    \adddialect\l@danish0\fi
```

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

`\captionsdanish` The macro `\captionsdanish` defines all strings used in the four standard documentclasses provided with LaTeX.

```
\addto\captionsdanish{%
  \def\prefacename{Forord}%
  \def\refname{Litteratur}%
  \def\abstractname{Resum\'e}%
  \def\bibname{Litteratur}%
  \def\chaptername{Kapitel}%
  \def\appendixname{Bilag}%
  \def\contentsname{Indhold}%
  \def\listfigurename{Figurer}%
  \def\listtablename{Tabeller}%
  \def\indexname{Indeks}%
  \def\figurename{Figur}%
  \def\tablename{Tabel}%
  \def\partname{Del}%
```

[45]The file described in this section has version number v1.3j and was last revised on 1996/12/23. A contribution was made by Henning Larsen (`larsen@cernvm.cern.ch`)

```
  \def\enclname{Vedlagt}%
  \def\ccname{Kopi til}%   or    Kopi sendt til
  \def\headtoname{Til}% in letter
  \def\pagename{Side}%
  \def\seename{Se}%
  \def\alsoname{Se ogs{\aa}}%
  \def\proofname{Bevis}%
  }%
```

`\datedanish` The macro `\datedanish` redefines the command `\today` to produce Danish dates.

```
\def\datedanish{%
\def\today{\number\day.~\ifcase\month\or
  januar\or februar\or marts\or april\or maj\or juni\or
  juli\or august\or september\or oktober\or november\or december\fi
  \space\number\year}}
```

`\extrasdanish` `\noextrasdanish` The macro `\extrasdanish` will perform all the extra definitions needed for the Danish language. The macro `\noextrasdanish` is used to cancel the actions of `\extrasdanish`.

Danish typesetting requires `\frencspacing` to be in effect.

```
\addto\extrasdanish{\bbl@frenchspacing}
\addto\noextrasdanish{\bbl@nonfrenchspacing}
```

For Danish the `"` character is made active. This is done once, later on its definition may vary. Other languages in the same document may also use the `"` character for shorthands; we specify that the danish group of shorthands should be used.

```
\initiate@active@char{"}
\addto\extrasdanish{\languageshorthands{danish}}
\addto\extrasdanish{\bbl@activate{"}}
%\addto\noextrasdanish{\bbl@deactivate{"}}
```

First we define access to the low opening double quote and guillemets for quotations,

```
\declare@shorthand{danish}{"`}{%
  \textormath{\quotedblbase{}}{\mbox{\quotedblbase}}}
\declare@shorthand{danish}{"'}{%
  \textormath{\textquotedblleft{}}{\mbox{\textquotedblleft}}}
\declare@shorthand{danish}{"<}{%
  \textormath{\guillemotleft{}}{\mbox{\guillemotleft}}}
\declare@shorthand{danish}{">}{%
  \textormath{\guillemotright{}}{\mbox{\guillemotright}}}
```

then we define to be able to specify hyphenation breakpoints that behave a little different from `\-`.

```
\declare@shorthand{danish}{"-}{\allowhyphens-\allowhyphens}
\declare@shorthand{danish}{""}{\hskip\z@skip}
\declare@shorthand{danish}{"~}{\textormath{\leavevmode\hbox{-}}{-}}
\declare@shorthand{danish}{"=}{\penalty\@M-\hskip\z@skip}
```

And we want to have a shorthand for disabling a ligature.

```
\declare@shorthand{danish}{"|}{%
  \textormath{\discretionary{-}{}{\kern.03em}}{}}
```

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.

```
31.53 \ldf@finish{danish}
31.54 ⟨/code⟩
```

32 The Norwegian language

The file `norsk.dtx`[46] defines all the language definition macros for the Norwegian language as well as for a new spelling variant 'nynorsk' for this language.

For this language currently no special definitions are needed or available.

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

```
⟨*code⟩
\LdfInit\CurrentOption{captions\CurrentOption}
```

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

```
\ifx\l@norsk\@undefined
    \@nopatterns{Norsk}
    \adddialect\l@norsk0\fi
```

`\norskhyphenmins` The Norwegian hyphenation patterns can be used with `\lefthyphenmin` set to 1 and `\righthyphenmin` set to 2. This is true for both 'versions' of the language.

```
\@namedef{\CurrentOption hyphenmins}{\@ne\tw@}
```

Now we have to decide which version of the captions should be made available. This can be done by checking the contents of `\CurrentOption`.

```
\def\bbl@tempa{norsk}
\ifx\CurrentOption\bbl@tempa
```

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

`\captionsnorsk` The macro `\captionsnorsk` defines all strings used in the four standard documentclasses provided with LaTeX.

```
  \def\captionsnorsk{%
    \def\prefacename{Forord}%
    \def\refname{Referanser}%
    \def\abstractname{Sammendrag}%
    \def\bibname{Bibliografi}%      or Litteraturoversikt
%                                   or Litteratur or Referanser
    \def\chaptername{Kapittel}%
    \def\appendixname{Tillegg}%    or Appendiks
    \def\contentsname{Innhold}%
    \def\listfigurename{Figurer}%  or Figurliste
    \def\listtablename{Tabeller}%  or Tabelliste
    \def\indexname{Register}%
    \def\figurename{Figur}%
    \def\tablename{Tabell}%
    \def\partname{Del}%
    \def\enclname{Vedlegg}%
    \def\ccname{Kopi sendt}%
    \def\headtoname{Til}% in letter
```

[46]The file described in this section has version number v1.2h and was last revised on 1996/12/23. Contributions were made by Haavard Helstrup (`HAAVARD@CERNVM`) and Alv Kjetil Holme (`HOLMEA@CERNVM`); the 'nynorsk' variant has been supplied by Per Steinar Iversen `iversen@vxcern.cern.ch`) and Terje Engeset Petterst (`TERJEEP@VSFYS1.FI.UIB.NO`).

```
   \def\pagename{Side}%
   \def\seename{Se}%
   \def\alsoname{Se ogs\aa{}}%
   \def\proofname{Bevis}%
   }
\else
```

For the 'nynorsk' version of these definitions we just add a "dialect".

```
  \adddialect\l@nynorsk\l@norsk
```

`\captionsnynorsk` The macro `\captionsnynorsk` defines all strings used in the four standard documentclasses provided with LaTeX, but using a different spelling than in the command `\captionsnorsk`.

```
 \def\captionsnynorsk{%
   \def\prefacename{Forord}%
   \def\refname{Referansar}%
   \def\abstractname{Samandrag}%
   \def\bibname{Litteratur}%     or Litteraturoversyn
%                          %    or Referansar
   \def\chaptername{Kapittel}%
   \def\appendixname{Tillegg}%   or Appendiks
   \def\contentsname{Innhald}%
   \def\listfigurename{Figurar}% or Figurliste
   \def\listtablename{Tabellar}% or Tabelliste
   \def\indexname{Register}%
   \def\figurename{Figur}%
   \def\tablename{Tabell}%
   \def\partname{Del}%
   \def\enclname{Vedlegg}%
   \def\ccname{Kopi sendt}%
   \def\headtoname{Til}% in letter
   \def\pagename{Side}%
   \def\seename{Sj\aa{}}%
   \def\alsoname{Sj\aa{} ogs\aa{}}%
   \def\proofname{Bevis}%
   }
\fi
```

`\datenorsk` The macro `\datenorsk` redefines the command `\today` to produce Norwegian dates.

```
\@namedef{date\CurrentOption}{%
 \def\today{\number\day.~\ifcase\month\or
   januar\or februar\or mars\or april\or mai\or juni\or
   juli\or august\or september\or oktober\or november\or desember
   \fi
   \space\number\year}}
```

`\extrasnorsk`
`\extrasnynorsk` The macro `\extrasnorsk` will perform all the extra definitions needed for the Norwegian language. The macro `\noextrasnorsk` is used to cancel the actions of `\extrasnorsk`.

Norwegian typesetting requires `\frencspacing` to be in effect.

```
\@namedef{extras\CurrentOption}{\bbl@frenchspacing}
\@namedef{noextras\CurrentOption}{\bbl@nonfrenchspacing}
```

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.

```
32.66 \ldf@finish\CurrentOption
32.67 ⟨/code⟩
```

33 The Swedish language

The file `swedish.dtx`[47] defines all the language-specific macros for the Swedish language.

For this language the character `"` is made active. In table 12 an overview is given of its purpose. The vertical placement of the "umlaut" in some letters can be controlled this way.

`"a`	`\"a`, also implemented for A, o and O.
`"w`	gives å, also works for uppercase letters.
`"ff`	for `ff` to be hyphenated as `ff-f`, this is also implemented for b, d, f, g, l, m, n, p, r, s, and t.
`"|`	disable ligature at this position.
`"-`	an explicit hyphen sign, allowing hyphenation in the rest of the word.
`""`	like `"-`, but producing no hyphen sign (for compound words with hyphen, e.g. `x-""y`).
`"~`	for a compound word mark without a breakpoint.
`"=`	for a compound word mark with a breakpoint, allowing hyphenation in the composing words.

Table 12: The extra definitions made by `swedish.sty`

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

```
33.1 ⟨*code⟩
33.2 \LdfInit{swedish}\captionsswedish
```

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

```
33.3 \ifx\l@swedish\@undefined
33.4     \@nopatterns{Swedish}
33.5     \adddialect\l@swedish0\fi
```

The next step consists of defining commands to switch to the Swedish language. The reason for this is that a user might want to switch back and forth between languages.

`\captionsswedish` The macro `\captionsswedish` defines all strings used in the four standard documentclasses provided with LaTeX.

```
33.6 \addto\captionsswedish{%
33.7   \def\prefacename{F\"orord}%
33.8   \def\refname{Referenser}%
33.9   \def\abstractname{Sammanfattning}%
33.10  \def\bibname{Litteraturf\"orteckning}%
33.11  \def\chaptername{Kapitel}%
```

[47]The file described in this section has version number v2.1 and was last revised on 1996/12/23. Contributions were made by Sten Hellman (`HELLMAN@CERNVM.CERN.CH`) and Erik Öshols (`erik@ine.kfk.de`).

```
  \def\appendixname{Bilaga}%
  \def\contentsname{Inneh\csname aa\endcsname ll}%
  \def\listfigurename{Figurer}%
  \def\listtablename{Tabeller}%
  \def\indexname{Sakregister}%
  \def\figurename{Figur}%
  \def\tablename{Tabell}%
  \def\partname{Del}%
  \def\enclname{Bil}%
  \def\ccname{Kopia f\"or k\"annedom}%
  \def\headtoname{Till}% in letter
  \def\pagename{Sida}%
  \def\seename{se}%

  \def\alsoname{se \"aven}%

  \def\proofname{Bevis}%
  }%
```

`\dateswedish` The macro `\dateswedish` redefines the command `\today` to produce Swedish dates.

```
\def\dateswedish{%
  \def\today{%
    \number\day~\ifcase\month\or
    januari\or februari\or mars\or april\or maj\or juni\or
    juli\or augusti\or september\or oktober\or november\or
    december\fi
    \space\number\year}}
```

`\swedishhyphenmins` The swedish hyphenation patterns can be used with `\lefthyphenmin` set to 2 and `\righthyphenmin` set to 2.

```
\def\swedishhyphenmins{\tw@\tw@}
```

`\extrasswedish` `\noextrasswedish` The macro `\extrasswedish` performs all the extra definitions needed for the Swedish language. The macro `\noextrasswedish` is used to cancel the actions of `\extrasswedish`.

For Swedish texts `\frenchspacing` should be in effect. We make sure this is the case and reset it if necessary.

```
\addto\extrasswedish{\bbl@frenchspacing}
\addto\noextrasswedish{\bbl@nonfrenchspacing}
```

For Swedish the `"` character is made active. This is done once, later on its definition may vary.

```
\initiate@active@char{"}
\addto\extrasswedish{\languageshorthands{swedish}}
\addto\extrasswedish{\bbl@activate{"}}
%\addto\noextrasswedish{\bbl@deactivate{"}}
```

The “umlaut” accent macro `\"` is changed to lower the umlaut dots. The redefinition is done with the help of `\umlautlow`.

```
\addto\extrasswedish{\babel@save\"\umlautlow}
\addto\noextrasswedish{\umlauthigh}
```

The code above is necessary because we need an extra active character. This character is then used as indicated in table 12.

To be able to define the function of ", we first define a couple of 'support' macros.

\dq We save the original double quote character in \dq to keep it available, the math accent \" can now be typed as ".

```
\begingroup \catcode`\"12
\def\x{\endgroup
  \def\@SS{\mathchar"7019 }
  \def\dq{"}}
\x
```

Now we can define the doublequote macros: the umlauts and å,

```
\declare@shorthand{swedish}{"w}{\textormath{{\aa}}{\ddot w}}
\declare@shorthand{swedish}{"a}{\textormath{\"{a}}{\ddot a}}
\declare@shorthand{swedish}{"o}{\textormath{\"{o}}{\ddot o}}
\declare@shorthand{swedish}{"W}{\textormath{{\AA}}{\ddot W}}
\declare@shorthand{swedish}{"A}{\textormath{\"{A}}{\ddot A}}
\declare@shorthand{swedish}{"O}{\textormath{\"{O}}{\ddot O}}
```

discretionary commands

```
\declare@shorthand{swedish}{"b}{\textormath{\bbl@disc b{bb}}{b}}
\declare@shorthand{swedish}{"B}{\textormath{\bbl@disc B{BB}}{B}}
\declare@shorthand{swedish}{"d}{\textormath{\bbl@disc d{dd}}{d}}
\declare@shorthand{swedish}{"D}{\textormath{\bbl@disc D{DD}}{D}}
\declare@shorthand{swedish}{"f}{\textormath{\bbl@disc f{ff}}{f}}
\declare@shorthand{swedish}{"F}{\textormath{\bbl@disc F{FF}}{F}}
\declare@shorthand{swedish}{"g}{\textormath{\bbl@disc g{gg}}{g}}
\declare@shorthand{swedish}{"G}{\textormath{\bbl@disc G{GG}}{G}}
\declare@shorthand{swedish}{"l}{\textormath{\bbl@disc l{ll}}{l}}
\declare@shorthand{swedish}{"L}{\textormath{\bbl@disc L{LL}}{L}}
\declare@shorthand{swedish}{"m}{\textormath{\bbl@disc m{mm}}{m}}
\declare@shorthand{swedish}{"M}{\textormath{\bbl@disc M{MM}}{M}}
\declare@shorthand{swedish}{"n}{\textormath{\bbl@disc n{nn}}{n}}
\declare@shorthand{swedish}{"N}{\textormath{\bbl@disc N{NN}}{N}}
\declare@shorthand{swedish}{"p}{\textormath{\bbl@disc p{pp}}{p}}
\declare@shorthand{swedish}{"P}{\textormath{\bbl@disc P{PP}}{P}}
\declare@shorthand{swedish}{"r}{\textormath{\bbl@disc r{rr}}{r}}
\declare@shorthand{swedish}{"R}{\textormath{\bbl@disc R{RR}}{R}}
\declare@shorthand{swedish}{"s}{\textormath{\bbl@disc s{ss}}{s}}
\declare@shorthand{swedish}{"S}{\textormath{\bbl@disc S{SS}}{S}}
\declare@shorthand{swedish}{"t}{\textormath{\bbl@disc t{tt}}{t}}
\declare@shorthand{swedish}{"T}{\textormath{\bbl@disc T{TT}}{T}}
```

and some additional commands:

```
\declare@shorthand{swedish}{"-}{\penalty\@M\-\allowhyphens}
\declare@shorthand{swedish}{"|}{%
  \textormath{\penalty\@M\discretionary{-}{}{\kern.03em}%
              \allowhyphens}{}}
\declare@shorthand{swedish}{""}{\hskip\z@skip}
\declare@shorthand{swedish}{"~}{\textormath{\leavevmode\hbox{-}}{-}}
\declare@shorthand{swedish}{"=}{\penalty\@M-\hskip\z@skip}
```

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{swedish}
⟨/code⟩
```

34 The Finnish language

The file `finnish.dtx`[48] defines all the language definition macros for the Finnish language.

For this language the character `"` is made active. In table 13 an overview is given of its purpose.

`"|`	disable ligature at this position.
`"-`	an explicit hyphen sign, allowing hyphenation in the rest of the word.
`"=`	an explicit hyphen sign for expressions such as "pakastekaapit ja -arkut".
`""`	like `"-`, but producing no hyphen sign (for words that should break at some sign such as "entrada/salida."
``"` ``	lowered double left quotes (looks like „)
`"'`	normal double right quotes
`"<`	for French left double quotes (similar to <<).
`">`	for French right double quotes (similar to >>).
`\-`	like the old `\-`, but allowing hyphenation in the rest of the word.

Table 13: The extra definitions made by `finnish.ldf`

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

```
⟨*code⟩
\LdfInit{finnish}\captionsfinnish
```

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

```
\ifx\l@finnish\@undefined
    \@nopatterns{Finnish}
    \adddialect\l@finnish0\fi
```

The next step consists of defining commands to switch to the Finnish language. The reason for this is that a user might want to switch back and forth between languages.

`\captionsfinnish` The macro `\captionsfinnish` defines all strings used in the four standard documentclasses provided with LaTeX.

```
\addto\captionsfinnish{%
  \def\prefacename{Esipuhe}%
  \def\refname{Viitteet}%
  \def\abstractname{Tiivistelm\"a}
  \def\bibname{Kirjallisuutta}%
  \def\chaptername{Luku}%
  \def\appendixname{Liite}%
```

[48]The file described in this section has version number v1.3j and was last revised on 1997/01/23. A contribution was made by Mikko KANERVA (`KANERVA@CERNVM`) and Keranen Reino (`KERANEN@CERNVM`).

```
  \def\contentsname{Sis\"alt\"o}%   /* Could be "Sis\"allys" as well */
  \def\listfigurename{Kuvat}%
  \def\listtablename{Taulukot}%
  \def\indexname{Hakemisto}%
  \def\figurename{Kuva}%
  \def\tablename{Taulukko}%
  \def\partname{Osa}%
  \def\enclname{Liitteet}%
  \def\ccname{Jakelu}%
  \def\headtoname{Vastaanottaja}%
  \def\pagename{Sivu}%
  \def\seename{katso}%
  \def\alsoname{katso my\"os}%
  \def\proofname{Todistus}%
  }%
```

`\datefinnish` The macro `\datefinnish` redefines the command `\today` to produce Finnish dates.

```
\def\datefinnish{%
\def\today{\number\day.~\ifcase\month\or
  tammikuuta\or helmikuuta\or maaliskuuta\or huhtikuuta\or
  toukokuuta\or kes\"akuuta\or hein\"akuuta\or elokuuta\or
  syyskuuta\or lokakuuta\or marraskuuta\or joulukuuta\fi
  \space\number\year}}
```

`\extrasfinnish` `\noextrasfinnish` Finnish has many long words (some of them compound, some not). For this reason hyphenation is very often the only solution in line breaking. For this reason the values of `\hyphenpenalty`, `\exhyphenpenalty` and `\doublehyphendemerits` should be decreased. (In one of the manuals of style Matti Rintala noticed a paragraph with ten lines, eight of which ended in a hyphen!)

Matti Rintala noticed that with these changes TEX handles Finnish very well, although sometimes the values of `\tolerance` and `\emergencystretch` must be increased. However, I don't think changing these values in `finnish.ldf` is appropriate, as the looseness of the font (and the line width) affect the correct choice of these parameters.

```
\addto\extrasfinnish{%
  \babel@savevariable\hyphenpenalty\hyphenpenalty=30%
  \babel@savevariable\exhyphenpenalty\exhyphenpenalty=30%
  \babel@savevariable\doublehyphendemerits\doublehyphendemerits=5000%
  \babel@savevariable\finalhyphendemerits\finalhyphendemerits=5000%
  }
\addto\noextrasfinnish{}
```

Another thing `\extrasfinnish` needs to do is to make sure that `\frenchspacing` is in effect. If this is not the case the execution of `\noextrasfinnish` will switch it of again.

```
\addto\extrasfinnish{\bbl@frenchspacing}
\addto\noextrasfinnish{\bbl@nonfrenchspacing}
```

For Finnish the `"` character is made active. This is done once, later on its definition may vary. Other languages in the same document may also use the `"` character for shorthands; we specify that the finnish group of shorthands should be used.

```
\initiate@active@char{"}
\addto\extrasfinnish{\languageshorthands{finnish}}
\addto\extrasfinnish{\bbl@activate{"}}
%\addto\noextrasfinnish{\bbl@deactivate{"}}
```

The ‘umlaut’ character should be positioned lower on *all* vowels in Finnish texts.

```
\addto\extrasfinnish{\umlautlow\umlautelow}
\addto\noextrasfinnish{\umlauthigh}
```

First we define access to the low opening double quote and guillemets for quotations,

```
\declare@shorthand{finnish}{"`}{%
  \textormath{\quotedblbase{}}{\mbox{\quotedblbase}}}
\declare@shorthand{finnish}{"'}{%
  \textormath{\textquotedblright{}}{\mbox{\textquotedblright}}}
\declare@shorthand{finnish}{"<}{%
  \textormath{\guillemotleft{}}{\mbox{\guillemotleft}}}
\declare@shorthand{finnish}{">}{%
  \textormath{\guillemotright{}}{\mbox{\guillemotright}}}
```

then we define two shorthands to be able to specify hyphenation breakpoints that behavew a little different from `\-`.

```
\declare@shorthand{finnish}{"-}{\allowhyphens-\allowhyphens}
\declare@shorthand{finnish}{""}{\hskip\z@skip}
\declare@shorthand{finnish}{"=}{\hbox{-}\allowhyphens}
```

And we want to have a shorthand for disabling a ligature.

```
\declare@shorthand{finnish}{"|}{%
  \textormath{\discretionary{-}{}{\kern.03em}}{}}
```

`\-` All that is left now is the redefinition of `\-`. The new version of `\-` should indicate an extra hyphenation position, while allowing other hyphenation positions to be generated automatically. The standard behaviour of TeX in this respect is very unfortunate for languages such as Dutch, Finnish and German, where long compound words are quite normal and all one needs is a means to indicate an extra hyphenation position on top of the ones that TeX can generate from the hyphenation patterns.

```
\addto\extrasfinnish{\babel@save\-}
\addto\extrasfinnish{\def\-{\allowhyphens
                                \discretionary{-}{}{}\allowhyphens}}
```

`\finishhyphenmins` The finnish hyphenation patterns can be used with `\lefthyphenmin` set to 2 and `\righthyphenmin` set to 2.

```
\def\finnishhyphenmins{\tw@\tw@}
```

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{finnish}
⟨/code⟩
```

35 The Hungarian language

The file option `magyar.dtx`[49] defines all the language definition macros for the Hungarian language.

`\ontoday` For this language currently the only special definition that is added is the `\ontoday` command which works like `\today` but produces a slightly different date format used in expressions suh as 'on february 10th'.

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

```
35.1 ⟨*code⟩
35.2 \LdfInit{magyar}{caption\CurrentOption}
```

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

```
35.3 \ifx\l@magyar\@undefined
35.4   \ifx\l@hungarian\@undefined
35.5     \@nopatterns{Magyar}
35.6     \adddialect\l@magyar0
35.7   \fi
35.8 \fi
35.9 \let\l@hungarian\l@magyar
```

An additional note about formatting Hungarian texts: One should invert the order of the number and text in things like chapter headings, page references etc. So one should write 'I. rész' instead of 'Part I', or '3. oldal' for 'page 3'.

For chapter headings this could be accomplished by a redefinition of the macros `\@makechapterhead` and `\@makeschapterhead`, for other instances this a lot harder to accomplish. Therefore I think complete document classes should be written to accomadate the needed formatting.

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

`\captionsmagyar` The macro `\captionsmagyar` defines all strings used in the four standard documentclasses provided with LaTeX.

```
35.10 \@namedef{captions\CurrentOption}{%
35.11   \def\prefacename{El\H osz\'o}%
```

For the list of references at the end of an article we have a choice between two words, 'Referenciák' (a Hungarian version of the English word) and 'Hivatkozások'. The latter seems to be in more widespread use.

```
35.12   \def\refname{Hivatkoz\'asok}%
```

If you have a document with a summary instead of an abstract you might want to replace the word 'Kivonat' with 'Összefoglaló'.

```
35.13   \def\abstractname{Kivonat}%
```

[49]The file described in this section has version number v1.3h and was last revised on 1996/12/23. A contribution was made by Attila Koppanyi (`attila@cernvm.cern.ch`). Later updates and suggestions by Árpád Bíró (`JZP1104@HUSZEG11.bitnet`), Istvan Hamecz (`hami@ursus.bke.hu`) and Horvath Dezso (`horvath@pisa.infn.it`).

The Hungarian version of ‘Bibliography’ is ‘Bibliográfia’, but a more natural word to use is ‘Irodalomjegyzék’.

```
  \def\bibname{Irodalomjegyz\'ek}%
  \def\chaptername{Fejezet}%
  \def\appendixname{F\"uggel\'ek}%
  \def\contentsname{Tartalomjegyz\'ek}%
  \def\listfigurename{\'Abr\'ak jegyz\'eke}%
  \def\listtablename{T\'abl\'azatok jegyz\'eke}%
  \def\indexname{T\'argymutat\'o}%
  \def\figurename{\'abra}%
  \def\tablename{T\'abl\'azat}%
  \def\partname{R\'esz}%
  \def\enclname{Mell\'eklet}%
  \def\ccname{K\"orlev\'el--c\'\i mzettek}%
  \def\headtoname{C\'\i mzett}%
  \def\pagename{oldal}%
  \def\seename{L\'asd}%
  \def\alsoname{L\'asd m\'eg}%
```

Besides the Hungarian word for Proof, ‘Bizonyítás’ we can also name Corollary (Következmény), Theorem (Tétel) and Lemma (Lemma).

```
  \def\proofname{Bizony\'\i t\'as}%
  }%
```

`\datemagyar` The macro `\datemagyar` redefines the command `\today` to produce Hungarian dates.

```
\@namedef{date\CurrentOption}{%
  \def\today{\number\year.~\ifcase\month\or
  janu\'ar\or febru\'ar\or m\'arcius\or
  \'aprilis\or m\'ajus\or j\'unius\or
  j\'ulius\or augusztus\or szeptember\or
  okt\'ober\or november\or december\fi
    \space\ifcase\day\or
    1.\or  2.\or  3.\or  4.\or  5.\or
    6.\or  7.\or  8.\or  9.\or 10.\or
   11.\or 12.\or 13.\or 14.\or 15.\or
   16.\or 17.\or 18.\or 19.\or 20.\or
   21.\or 22.\or 23.\or 24.\or 25.\or
   26.\or 27.\or 28.\or 29.\or 30.\or
   31.\fi}}
```

`\ondatemagyar` The macro `\ondatemagyar` produces Hungarian dates which have the meaning ‘*on this day*’. It does not redefine the command `\today`.

```
\@namedef{ondate\CurrentOption}{%
  \number\year.~\ifcase\month\or
  janu\'ar\or febru\'ar\or m\'arcius\or
  \'aprilis\or m\'ajus\or j\'unius\or
  j\'ulius\or augusztus\or szeptember\or
  okt\'ober\or november\or december\fi
    \space\ifcase\day\or
    1-j\'en\or  2-\'an\or  3-\'an\or  4-\'en\or  5-\'en\or
    6-\'an\or  7-\'en\or  8-\'an\or  9-\'en\or 10-\'en\or
   11-\'en\or 12-\'en\or 13-\'an\or 14-\'en\or 15-\'en\or
   16-\'an\or 17-\'en\or 18-\'an\or 19-\'en\or 20-\'an\or
  21-\'en\or 22-\'en\or 23-\'an\or 24-\'en\or 25-\'en\or
  26-\'an\or 27-\'en\or 28-\'an\or 29-\'en\or 30-\'an\or
  31-\'en\fi}
```

`\extrasmagyar` `\noextrasmagyar` The macro `\extrasmagyar` will perform all the extra definitions needed for the Hungarian language. The macro `\noextrasmagyar` is used to cancel the actions of `\extrasmagyar`. For the moment these macros are nearly empty; only the user command `\ontoday` to access `\ondatemagyar` is defined.

```
\@namedef{extras\CurrentOption}{%
  \expandafter\let\expandafter\ontoday
    \csname ondate\CurrentOption\endcsname}
\@namedef{noextras\CurrentOption}{\let\ontoday\@undefined}
```

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\CurrentOption
⟨/code⟩
```

36 The Estonian language

The file `estonian.dtx`[50] defines the language definition macro's for the Estonian language.

This file was written as part of the TWGML project, and borrows heavily from the babel German and Spanish language files `germanb.ldf` and `spanish.ldf`.

Estonian has the same umlauts as German (ä, ö, ü), but in addition to this, we have also õ, and two recent characters š and ž, so we need at least two active characters. We shall use `"` and `~` to type Estonian accents on ASCII keyboards (in the 7-bit character world). Their use is given in table 14. These active accent

`~o`	`\~o`, (and uppercase);
`"a`	`\"a`, (and uppercase);
`"o`	`\"o`, (and uppercase);
`"u`	`\"u`, (and uppercase);
`~s`	`\v s`, (and uppercase);
`~z`	`\v z`, (and uppercase);
`"|`	disable ligature at this position;
`"-`	an explicit hyphen sign, allowing hyphenation in the rest of the word;
`\-`	like the old `\-`, but allowing hyphenation in the rest of the word;
`"‘`	for Estonian low left double quotes (same as German);
`"’`	for Estonian right double quotes;
`"<`	for French left double quotes (also rather popular)
`">`	for French right double quotes.

Table 14: The extra definitions made by `estonian.ldf`

characters behave according to their original definitions if not followed by one of the characters indicated in that table; the original quote character can be typed using the macro `\dq`.

We support also the T1 output encoding (and Cork-encoded text input). You can choose the T1 encoding by the command `\usepackage[T1]{fontenc}`. This package must be loaded before babel. As the standard Estonian hyphenation file `eehyph.tex` is in the Cork encoding, choosing this encoding will give you better hyphenation.

As mentioned in the Spanish style file, it may happen that some packages fail (usually in a `\message`). In this case you should change the order of the `\usepackage` declarations or the order of the style options in `\documentclass`.

36.1 Implementation

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

36.1 ⟨∗code⟩

[50]The file described in this section has version number v1.0e and was last revised on 1996/12/23. The original author is Enn Saar, (`saar@aai.ee`).

```
\LdfInit{estonian}\captionsestonian
```

If Estonian is not included in the format file (does not have hyphenation patterns), we shall use English hyphenation.

```
\ifx\l@estonian\@undefined
  \@nopatterns{Estonian}
  \adddialect\l@estonian0
\fi
```

Now come the commands to switch to (and from) Estonian.

`\captionsestonian` The macro `\captionsestonian` defines all strings used in the four standard documentclasses provided with LaTeX.

```
\addto\captionsestonian{%
  \def\prefacename{Sissejuhatus}%
  \def\refname{Viited}%
  \def\bibname{Kirjandus}%
  \def\appendixname{Lisa}%
  \def\contentsname{Sisukord}%
  \def\listfigurename{Joonised}%
  \def\listtablename{Tabelid}%
  \def\indexname{Indeks}%
  \def\figurename{Joonis}%
  \def\tablename{Tabel}%
  \def\partname{Osa}%
  \def\enclname{Lisa(d)}%
  \def\ccname{Koopia(d)}%
  \def\headtoname{}%
  \def\pagename{Lk.}%
  \def\seename{vt.}%
  \def\alsoname{vt. ka}%
  \def\proofname{Korrektuur}%
  }
```

These captions contain accented characters.

```
\begingroup \catcode`\"\active
\def\x{\endgroup
\addto\captionsestonian{%
  \def\abstractname{Kokkuv~ote}%
  \def\chaptername{Peat"ukk}}}
\x
```

`\dateestonian` The macro `\dateestonian` redefines the command `\today` to produce Estonian dates.

```
\begingroup \catcode`\"\active
\def\x{\endgroup
   \def\month@estonian{\ifcase\month\or
     jaanuar\or veebruar\or m"arts\or aprill\or mai\or juuni\or
     juuli\or august\or september\or oktoober\or november\or
         detsember\fi}}
\x
\def\dateestonian{\def\today{\number\day.\space\month@estonian
  \space\number\year.\space a.}}
```

`\extrasestonian` `\noextrasestonian` The macro `\extrasestonian` will perform all the extra definitions needed for Estonian. The macro `\noextrasestonian` is used to cancel the actions of `\extrasestonian`. For Estonian, `"` is made active and has to be treated as 'special' (`~` is active already).

```
\initiate@active@char{"}
\initiate@active@char{~}
\addto\extrasestonian{\languageshorthands{estonian}}
\addto\extrasestonian{\bbl@activate{"}\bbl@activate{~}}
```

Store the original macros, and redefine accents.

```
\addto\extrasestonian{\babel@save\"\umlautlow\babel@save\~\tildelow}
```

Estonian does not use extra spaces after sentences.

```
\addto\extrasestonian{\bbl@frenchspacing}
\addto\noextrasestonian{\bbl@nonfrenchspacing}
```

`\estonianhyphenmins` For Estonian, `\lefthyphenmin` and `\righthyphenmin` are both 2.

```
\def\estonianhyphenmins{\tw@\tw@}
```

`\tildelow` `\gentilde` `\newtilde` `\newcheck` The standard TEX accents are too high for Estonian typography, we have to lower them (following the babel German style). For a detailed explanation see the file `glyphs.dtx`.

```
\def\tildelow{\def\~{\protect\gentilde}}
\def\gentilde#1{\if#1o\newtilde{#1}\else\if#1O\newtilde{#1}%
    \else\newcheck{#1}%
    \fi\fi}
\def\newtilde#1{\leavevmode\allowhyphens
  {\U@D 1ex%
  {\setbox\z@\hbox{\char126}\dimen@ -.45ex\advance\dimen@\ht\z@
  \ifdim 1ex<\dimen@ \fontdimen5\font\dimen@ \fi}%
  \accent126\fontdimen5\font\U@D #1}\allowhyphens}
\def\newcheck#1{\leavevmode\allowhyphens
  {\U@D 1ex%
  {\setbox\z@\hbox{\char20}\dimen@ -.45ex\advance\dimen@\ht\z@
  \ifdim 1ex<\dimen@ \fontdimen5\font\dimen@ \fi}%
  \accent20\fontdimen5\font\U@D #1}\allowhyphens}
```

We save the double quote character in `\dq`, and tilde in `\til`, and store the original definitions of `\"` and `~` as `\dieresis` and `\texttilde`.

```
\begingroup \catcode`\"12
\edef\x{\endgroup
  \def\noexpand\dq{"}
  \def\noexpand\til{~}}
\x
\let\dieresis\"
\let\texttilde\~
```

This part follows closely `spanish.ldf`. We check the encoding and if it is T1, we have to tell TEX about our redefined accents.

```
\edef\next{T1}
\ifx\f@encoding\next
  \let\@umlaut\dieresis
  \let\@tilde\texttilde
  \DeclareTextComposite{\~}{T1}{s}{178}
  \DeclareTextComposite{\~}{T1}{S}{146}
  \DeclareTextComposite{\~}{T1}{z}{186}
  \DeclareTextComposite{\~}{T1}{Z}{154}
  \DeclareTextComposite{\"}{T1}{'}{17}
  \DeclareTextComposite{\"}{T1}{`}{18}
  \DeclareTextComposite{\"}{T1}{<}{19}
  \DeclareTextComposite{\"}{T1}{>}{20}
```

If the encoding differs from T1, we expand the accents, enabling hyphenation beyond the accent. In this case TEX will not find all possible breaks, and we have to warn people.

```
\else
  \wlog{Warning: Hyphenation would work better for the T1 encoding.}
  \let\@umlaut\newumlaut
  \let\@tilde\gentilde
\fi
```

Now we define the shorthands.

```
\declare@shorthand{estonian}{"a}{\textormath{\"{a}}{\ddot a}}
\declare@shorthand{estonian}{"A}{\textormath{\"{A}}{\ddot A}}
\declare@shorthand{estonian}{"o}{\textormath{\"{o}}{\ddot o}}
\declare@shorthand{estonian}{"O}{\textormath{\"{O}}{\ddot O}}
\declare@shorthand{estonian}{"u}{\textormath{\"{u}}{\ddot u}}
\declare@shorthand{estonian}{"U}{\textormath{\"{U}}{\ddot U}}
```

german and french quotes,

```
\declare@shorthand{estonian}{"`}{%
  \textormath{\quotedblbase{}}{\mbox{\quotedblbase}}}
\declare@shorthand{estonian}{"'}{%
  \textormath{\textquotedblleft{}}{\mbox{\textquotedblleft}}}
\declare@shorthand{estonian}{"<}{%
  \textormath{\guillemotleft{}}{\mbox{\guillemotleft}}}
\declare@shorthand{estonian}{">}{%
  \textormath{\guillemotright{}}{\mbox{\guillemotright}}}
```

```
\declare@shorthand{estonian}{~o}{\textormath{\@tilde o}{\tilde o}}
\declare@shorthand{estonian}{~O}{\textormath{\@tilde O}{\tilde O}}
\declare@shorthand{estonian}{~s}{\textormath{\@tilde s}{\check s}}
\declare@shorthand{estonian}{~S}{\textormath{\@tilde S}{\check S}}
\declare@shorthand{estonian}{~z}{\textormath{\@tilde z}{\check z}}
\declare@shorthand{estonian}{~Z}{\textormath{\@tilde Z}{\check Z}}
```

and some additional commands:

```
\declare@shorthand{estonian}{"-}{\allowhyphens\-\allowhyphens}
\declare@shorthand{estonian}{"|}{%
  \textormath{\penalty\@M\discretionary{-}{}{\kern.03em}%
              \allowhyphens}{}}
\declare@shorthand{estonian}{""}{\dq}
\declare@shorthand{estonian}{~~}{\til}
```

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{estonian}
⟨/code⟩
```

37 The Croatian language

The file `croatian.dtx`[51] defines all the language definition macros for the Croatian language.

For this language currently no special definitions are needed or available.

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

```
⟨*code⟩
\LdfInit{croatian}\captionscroatian
```

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

```
\ifx\l@croatian\@undefined
    \@nopatterns{Croatian}
    \adddialect\l@croatian0\fi
```

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

`\captionscroatian` The macro `\captionscroatian` defines all strings used in the four standard documentclasses provided with LaTeX.

```
\addto\captionscroatian{%
  \def\prefacename{Predgovor}%
  \def\refname{Literatura}%
  \def\abstractname{Sa\v{z}etak}%
  \def\bibname{Bibliografija}%
  \def\chaptername{Glava}%
  \def\appendixname{Dodatak}%
  \def\contentsname{Sadr\v{z}aj}%
  \def\listfigurename{Slike}%
  \def\listtablename{Tablice}%
  \def\indexname{Indeks}%
  \def\figurename{Slika}%
  \def\tablename{Tablica}%
  \def\partname{Dio}%
  \def\enclname{Prilozi}%
  \def\ccname{Kopije}%
  \def\headtoname{Prima}%
  \def\pagename{Strana}%
  \def\seename{Vidi}%
  \def\alsoname{Vidi tako\dj er}%
  \def\proofname{Dokaz}%
  }%
```

`\datecroatian` The macro `\datecroatian` redefines the command `\today` to produce Croatian dates.

```
\def\datecroatian{%
  \def\today{\number\day .~\ifcase\month\or
    sijev{c}nja\or velja\v{c}e\or o\v{z}ujka\or travnja\or svibnja\or
```

[51]The file described in this section has version number v1.3g and was last revised on 1996/12/23. A contribution was made by Alan Paić (`paica@cernvm.cern.ch`).

```
    lipnja\or srpnja\or kolovoza\or rujna\or listopada\od studenog\or
    prosinca\fi \space \number\year}}
```

`\extrascroatian` `\noextrascroatian` The macro `\extrascroatian` will perform all the extra definitions needed for the Croatian language. The macro `\noextrascroatian` is used to cancel the actions of `\extrascroatian`. For the moment these macros are empty but they are defined for compatibility with the other language definition files.

```
\addto\extrascroatian{}
\addto\noextrascroatian{}
```

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{croatian}
⟨/code⟩
```

38 The Czech language

The file `czech.dtx`[52] defines all the language definition macros for the Czech language.

For this language `\frenchspacing` is set and two macros `\q` and `\w` for easy access to two accents are defined.

The command `\q` is used with the letters (`t`, `d`, `l`, and `L`) and adds a `'` to them to simulate a 'hook' that should be there. The result looks like ť. The command `\w` is used to put the ring-accent which appears in ångstrøm over the letters `u` and `U`.

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

```
⟨*code⟩
\LdfInit{czech}\captionsczech
```

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

```
\ifx\l@czech\@undefined
    \@nopatterns{Czech}
    \adddialect\l@czech0\fi
```

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

`\captionsczech` The macro `\captionsczech` defines all strings used in the four standard documentclasses provided with LaTeX.

```
\addto\captionsczech{%
  \def\prefacename{P\v redmluva}%
  \def\refname{Reference}%
  \def\abstractname{Abstrakt}%
  \def\bibname{Literatura}%
  \def\chaptername{Kapitola}%
  \def\appendixname{Dodatek}%
  \def\contentsname{Obsah}%
  \def\listfigurename{Seznam obr\'azk\r{u}}%
  \def\listtablename{Seznam tabulek}%
  \def\indexname{Index}%
  \def\figurename{Obr\'azek}%
  \def\tablename{Tabulka}%
  \def\partname{\v{C}\'ast}%
  \def\enclname{P\v{r}\'{\i}loha}%
  \def\ccname{Na v\v{e}dom\'{\i}:}%
  \def\headtoname{Komu}%
  \def\pagename{Strana}%
  \def\seename{viz}%
  \def\alsoname{viz tak\'e}%
  \def\proofname{D\r{u}kaz}%
  }%
```

[52] The file described in this section has version number v1.3h and was last revised on 1996/12/23. Contributions were made by Milos Lokajicek (`LOKAJICK@CERNVM`).

\dateczech The macro \dateczech redefines the command \today to produce Czech dates.

```
\def\dateczech{%
\def\today{\number\day.~\ifcase\month\or
  ledna\or \'unora\or b\v{r}ezna\or dubna\or kv\v{e}tna\or \v{c}ervna\or
  \v{c}ervence\or srpna\or z\'a\v{r}\'{\i}\or \v{r}\'{\i}jna\or
  listopadu\or prosince\fi
  \space \number\year}}
```

\extrasczech \noextrasczech The macro \extrasczech will perform all the extra definitions needed for the Czech language. The macro \noextrasczech is used to cancel the actions of \extrasczech. This means saving the meaning of two one-letter control sequences before defining them.

```
\addto\extrasczech{\babel@save\q\let\q\v}
\addto\extrasczech{\babel@save\w\let\w\r}
```

For Czech texts \frenchspacing should be in effect. We make sure this is the case and reset it if necessary.

```
\addto\extrasczech{\bbl@frenchspacing}
\addto\noextrasczech{\bbl@nonfrenchspacing}
```

\v LaTeX's normal \v accent places a caron over the letter that follows it (ǒ). This is not what we want for the letters d, t, l and L; for those the accent should change shape. This is acheived by the following.

```
\AtBeginDocument{%
  \DeclareTextCompositeCommand{\v}{OT1}{t}{%
    t\kern-.23em\raise.24ex\hbox{'}}
  \DeclareTextCompositeCommand{\v}{OT1}{d}{%
    d\kern-.13em\raise.24ex\hbox{'}}
  \DeclareTextCompositeCommand{\v}{OT1}{l}{\lcaron{}}
  \DeclareTextCompositeCommand{\v}{OT1}{L}{\Lcaron{}}}
```

\lcaron \Lcaron Fot the letters l and L we want to disinguish between normal fonts and monospaced fonts.

```
\def\lcaron{%
  \setbox0\hbox{M}\setbox\tw@\hbox{i}%
  \ifdim\wd0>\wd\tw@\relax
    l\kern-.13em\raise.24ex\hbox{'}\kern-.11em%
  \else
    l\raise.45ex\hbox to\z@{\kern-.35em '\hss}%
  \fi}
\def\Lcaron{%
  \setbox0\hbox{M}\setbox\tw@\hbox{i}%
  \ifdim\wd0>\wd\tw@\relax
    L\raise.24ex\hbox to\z@{\kern-.28em'\hss}%
  \else
    L\raise.45ex\hbox to\z@{\kern-.40em '\hss}%
  \fi}
```

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{czech}
⟨/code⟩
```

39 The Polish language

The file `polish.dtx`[53] defines all the language-specific macros for the Polish language.

For this language the character `"` is made active. In table 15 an overview is given of its purpose.

`"a`	or `\aob`, for tailed-a (like ą)
`"A`	or `\Aob`, for tailed-A (like Ą)
`"e`	or `\eob`, for tailed-e (like ę)
`"E`	or `\Eob`, for tailed-E (like Ę)
`"c`	or `\'c`, for accented c (like ć), same with uppercase letters and n,o,s
`"l`	or `\lpb{}`, for l with stroke (like ł)
`"L`	or `\Lpb{}`, for L with stroke (like Ł)
`"r`	or `\zkb{}`, for pointed z (like ż), cf. pronounciation
`"R`	or `\Zkb{}`, for pointed Z (like Ż)
`"z`	or `\'z`, for accented z
`"Z`	or `\'Z`, for accented Z
`"|`	disable ligature at this position.
`"-`	an explicit hyphen sign, allowing hyphenation in the rest of the word.
`""`	like `"-`, but producing no hyphen sign (for compund words with hyphen, e.g. `x-""y`).
`"‘`	for German left double quotes (looks like ,,).
`"’`	for German right double quotes.
`"<`	for French left double quotes (similar to <<).
`">`	for French right double quotes (similar to >>).

Table 15: The extra definitions made by `polish.sty`

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

```
39.1 ⟨*code⟩
39.2 \LdfInit{polish}\captionspolish
```

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

```
39.3 \ifx\l@polish\@undefined
39.4   \@nopatterns{Polish}
39.5   \adddialect\l@polish0\fi
```

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

`\captionspolish` The macro `\captionspolish` defines all strings used in the four standard documentclasses provided with LaTeX.

[53]The file described in this section has version number v1.2d and was last revised on 1996/12/23.

```
\addto\captionspolish{%
  \def\prefacename{Przedmowa}%
  \def\refname{Bibliografia}%
  \def\abstractname{Streszczenie}%
  \def\bibname{Literatura}%
  \def\chaptername{Rozdzia\l}%
  \def\appendixname{Dodatek}%
  \def\contentsname{Spis rzeczy}%
  \def\listfigurename{Spis rysunk\'ow}%
  \def\listtablename{Spis tablic}%
  \def\indexname{Indeks}%
  \def\figurename{Rysunek}%
  \def\tablename{Tablica}%
  \def\partname{Cz\eob{}\'s\'c}%
  \def\enclname{Za\l\aob{}cznik}%
  \def\ccname{Kopie:}%
  \def\headtoname{Do}%
  \def\pagename{Strona}%
  \def\seename{Por\'ownaj}%
  \def\alsoname{Por\'ownaj tak\.ze}%
  \def\proofname{Proof}%   <-- needs translation
}
```

`\datepolish` The macro `\datepolish` redefines the command `\today` to produce Polish dates.

```
\def\datepolish{%
  \def\today{\number\day~\ifcase\month\or
  stycznia\or lutego\or marca\or kwietnia\or maja\or czerwca\or lipca\or
  sierpnia\or wrze\'snia\or pa\'zdziernika\or listopada\or grudnia\fi
  \space\number\year}
}
```

`\extraspolish` `\noextraspolish` The macro `\extraspolish` will perform all the extra definitions needed for the Polish language. The macro `\noextraspolish` is used to cancel the actions of `\extraspolish`.

For Polish the `"` character is made active. This is done once, later on its definition may vary. Other languages in the same document may also use the `"` character for shorthands; we specify that the polish group of shorthands should be used.

```
\initiate@active@char{"}
\addto\extraspolish{\languageshorthands{polish}}
\addto\extraspolish{\bbl@activate{"}}
%\addto\noextraspolish{\bbl@deactivate{"}}
```

The code above is necessary because we need an extra active character. This character is then used as indicated in table 15.

If you have problems at the end of a word with a linebreak, use the other version without hyphenation tricks. Some TeX wizard may produce a better solution with forcasting another token to decide whether the character after the double quote is the last in a word. Do it and let us know.

In Polish texts some letters get special diacritical marks. Leszek Holenderski designed the following code to position the diacritics correctly for every font in every size. These macros need a few extra dimension variables.

```
\newdimen\pl@left
\newdimen\pl@down
\newdimen\pl@right
\newdimen\pl@temp
```

`\sob` The macro `\sob` is used to put the 'ogonek' in the right place.

```
\def\sob#1#2#3#4#5{%parameters: letter and fractions hl,ho,vl,vo
  \setbox0\hbox{#1}\setbox1\hbox{$_\mathchar'454$}\setbox2\hbox{p}%
  \pl@right=#2\wd0 \advance\pl@right by-#3\wd1
  \pl@down=#5\ht1 \advance\pl@down by-#4\ht0
  \pl@left=\pl@right \advance\pl@left by\wd1
  \pl@temp=-\pl@down \advance\pl@temp by\dp2 \dp1=\pl@temp
  \leavevmode
  \kern\pl@right\lower\pl@down\box1\kern-\pl@left #1}
```

`\aob` `\Aob` `\eob` `\Eob` The ogonek is placed with the letters 'a', 'A', 'e', and 'E'.

```
\DeclareTextCommand{\aob}{OT1}{\sob a{.66}{.20}{0}{.90}}
\DeclareTextCommand{\Aob}{OT1}{\sob A{.80}{.50}{0}{.90}}
\DeclareTextCommand{\eob}{OT1}{\sob e{.50}{.35}{0}{.93}}
\DeclareTextCommand{\Eob}{OT1}{\sob E{.60}{.35}{0}{.90}}
```

For the 'new' `T1` encoding we can provide simpler definitions.

```
\DeclareTextCommand{\aob}{T1}{\k a}
\DeclareTextCommand{\Aob}{T1}{\k A}
\DeclareTextCommand{\eob}{T1}{\k e}
\DeclareTextCommand{\Eob}{T1}{\k E}
```

Construct the characters by default from the OT1 encoding.

```
\ProvideTextCommandDefault{\aob}{\UseTextSymbol{OT1}{\aob}}
\ProvideTextCommandDefault{\Aob}{\UseTextSymbol{OT1}{\Aob}}
\ProvideTextCommandDefault{\eob}{\UseTextSymbol{OT1}{\eob}}
\ProvideTextCommandDefault{\Eob}{\UseTextSymbol{OT1}{\Eob}}
```

`\spb` The macro `\spb` is used to put the 'poprzeczka' in the right place.

```
\def\spb#1#2#3#4#5{%
  \setbox0\hbox{#1}\setbox1\hbox{\char'023}%
  \pl@right=#2\wd0 \advance\pl@right by-#3\wd1
  \pl@down=#5\ht1 \advance\pl@down by-#4\ht0
  \pl@left=\pl@right \advance\pl@left by\wd1
  \ht1=\pl@down \dp1=-\pl@down
  \leavevmode
  \kern\pl@right\lower\pl@down\box1\kern-\pl@left #1}
```

`\skb` The macro `\skb` is used to put the 'kropka' in the right place.

```
\def\skb#1#2#3#4#5{%
  \setbox0\hbox{#1}\setbox1\hbox{\char'056}%
  \pl@right=#2\wd0 \advance\pl@right by-#3\wd1
  \pl@down=#5\ht1 \advance\pl@down by-#4\ht0
  \pl@left=\pl@right \advance\pl@left by\wd1
  \leavevmode
  \kern\pl@right\lower\pl@down\box1\kern-\pl@left #1}
```

`\textpl` For the ‘poprzeczka’ and the ‘kropka’ in text fonts we don’t need any special coding, but we can (almost) use what is already available.

```
\def\textpl{%
  \def\lpb{\plll}%
  \def\Lpb{\pLLL}%
  \def\zkb{\.z}%
  \def\Zkb{\.Z}}
```

Initially we assume that typesetting is done with text fonts.

```
\textpl
```

```
\let\lll=\l \let\LLL=\L
\def\plll{\lll}
\def\pLLL{\LLL}
```

`\telepl` But for the ‘teletype’ font in ‘OT1’ encoding we have to take some special actions, involving the macros defined above.

```
\def\telepl{%
  \def\lpb{\spb l{.45}{.5}{.4}{.8}}%
  \def\Lpb{\spb L{.23}{.5}{.4}{.8}}%
  \def\zkb{\skb z{.5}{.5}{1.2}{0}}%
  \def\Zkb{\skb Z{.5}{.5}{1.1}{0}}}
```

To activate these codes the font changing commands as they are defined in LaTeX are modified. The same is done for plain TeX’s font changing commands.

When `\selectfont` is undefined the current format is spposed to be either plain (based) or LaTeX 2.09.

```
\ifx\selectfont\@undefined
  \ifx\prm\@undefined \addto\rm{\textpl}\else \addto\prm{\textpl}\fi
  \ifx\pit\@undefined \addto\it{\textpl}\else \addto\pit{\textpl}\fi
  \ifx\pbf\@undefined \addto\bf{\textpl}\else \addto\pbf{\textpl}\fi
  \ifx\psl\@undefined \addto\sl{\textpl}\else \addto\psl{\textpl}\fi
  \ifx\psf\@undefined                   \else \addto\psf{\textpl}\fi
  \ifx\psc\@undefined                   \else \addto\psc{\textpl}\fi
  \ifx\ptt\@undefined \addto\tt{\telepl}\else \addto\ptt{\telepl}\fi
\else
```

When `\selectfont` exists we assume LaTeX 2_ε.

```
  \expandafter\addto\csname selectfont \endcsname{%
    \csname\f@encoding @pl\endcsname}
\fi
```

Currently we support the OT1 and T1 encodings. For T1 we don’t have to make a difference between typewriter fonts and other fonts, they all have the same glyphs.

```
\expandafter\let\csname T1@pl\endcsname\textpl
```

For OT1 we need to check the current font family, stored in `\f@family`. Unfortunately we need a hack as `\ttdefault` is defined as a `\long` macro, while `\f@family` is not.

```
\expandafter\def\csname OT1@pl\endcsname{%
  \long\edef\curr@family{\f@family}%
  \ifx\curr@family\ttdefault
    \telepl
  \else
    \textpl
  \fi}
```

`\dq` We save the original double quote character in `\dq` to keep it available, the math accent `\"` can now be typed as `"`.

```
\begingroup \catcode`\"12
\def\x{\endgroup
  \def\dq{"}}
\x
```

Now we can define the doublequote macros for diacritics,

```
\declare@shorthand{polish}{"a}{\textormath{\aob}{\ddot a}}
\declare@shorthand{polish}{"A}{\textormath{\Aob}{\ddot A}}
\declare@shorthand{polish}{"c}{\textormath{\'c}{\acute c}}
\declare@shorthand{polish}{"C}{\textormath{\'C}{\acute C}}
\declare@shorthand{polish}{"e}{\textormath{\eob}{\ddot e}}
\declare@shorthand{polish}{"E}{\textormath{\Eob}{\ddot E}}
\declare@shorthand{polish}{"l}{\textormath{\lpb}{\ddot l}}
\declare@shorthand{polish}{"L}{\textormath{\Lpb}{\ddot L}}
\declare@shorthand{polish}{"n}{\textormath{\'n}{\acute n}}
\declare@shorthand{polish}{"N}{\textormath{\'N}{\acute N}}
\declare@shorthand{polish}{"o}{\textormath{\'o}{\acute o}}
\declare@shorthand{polish}{"O}{\textormath{\'O}{\acute O}}
\declare@shorthand{polish}{"r}{\textormath{\zkb}{\ddot r}}
\declare@shorthand{polish}{"R}{\textormath{\Zkb}{\ddot R}}
\declare@shorthand{polish}{"s}{\textormath{\'s}{\acute s}}
\declare@shorthand{polish}{"S}{\textormath{\'S}{\acute S}}
\declare@shorthand{polish}{"z}{\textormath{\'z}{\acute z}}
\declare@shorthand{polish}{"Z}{\textormath{\'Z}{\acute Z}}
```

Then we define access to two forms of quotation marks, similar to the german and french quotation marks.

```
\declare@shorthand{polish}{"`}{%
  \textormath{\quotedblbase{}}{\mbox{\quotedblbase}}}
\declare@shorthand{polish}{"'}{%
  \textormath{\textquotedblleft{}}{\mbox{\textquotedblleft}}}
\declare@shorthand{polish}{"<}{%
  \textormath{\guillemotleft{}}{\mbox{\guillemotleft}}}
\declare@shorthand{polish}{">}{%
  \textormath{\guillemotright{}}{\mbox{\guillemotright}}}
```

then we define two shorthands to be able to specify hyphenation breakpoints that behavew a little different from `\-`.

```
\declare@shorthand{polish}{"-}{\allowhyphens-\allowhyphens}
\declare@shorthand{polish}{""}{\hskip\z@skip}
```

And we want to have a shorthand for disabling a ligature.

```
\declare@shorthand{polish}{"|}{%
  \textormath{\discretionary{-}{}{\kern.03em}}{}}
```

`\mdqon` `\mdqoff` All that's left to do now is to define a couple of commands for reasons of compatibility with `polish.tex`.

```
\def\mdqon{\bbl@activate{"}}
\def\mdqoff{\bbl@deactivate{"}}
```

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{polish}
⟨/code⟩
```

40 The Slovak language

The file `slovak.dtx`[54] defines all the language-specific macros for the Slovak language.

For this language the macro `\q` is defined. It is used with the letters (`t`, `d`, `l`, and `L`) and adds a `'` to them to simulate a 'hook' that should be there. The result looks like ť.

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

```
⟨*code⟩
\LdfInit{slovak}\captionsslovak
```

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

```
\ifx\l@slovak\@undefined
    \@nopatterns{Slovak}
    \adddialect\l@slovak0\fi
```

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

`\captionsslovak` The macro `\captionsslovak` defines all strings used in the four standard documentclasses provided with LaTeX.

```
\addto\captionsslovak{%
  \def\prefacename{\'Uvod}%
  \def\refname{Referencia}%
  \def\abstractname{Abstrakt}%
  \def\bibname{Literat\'ura}%
  \def\chaptername{Kapitola}%
  \def\appendixname{Dodatok}%
  \def\contentsname{Obsah}%
  \def\listfigurename{Zoznam obr\'azkov}%
  \def\listtablename{Zoznam tabuliek}%
  \def\indexname{Index}%
  \def\figurename{Obr\'azok}%
  \def\tablename{Tabu\q lka}%%% special letter l with hook
  \def\partname{\v{C}as\q t}%%% special letter t with hook
  \def\enclname{Pr\'{\i}loha}%
  \def\ccname{CC}%
  \def\headtoname{Komu}%
  \def\pagename{Strana}%
  \def\seename{vi\q d}%%%  Special letter d with hook
  \def\alsoname{vi\q d tie\v z}%%%  Special letter d with hook
  \def\proofname{Proof}%  <-- needs translation
  }
```

`\dateslovak` The macro `\dateslovak` redefines the command `\today` to produce Slovak dates.

```
\def\dateslovak{%
\def\today{\number\day.~\ifcase\month\or
janu\'ara\or febru\'ara\or marca\or apr\'{\i}la\or m\'aja\or j\'una\or
```

[54]The file described in this section has version number v1.2i and was last revised on 1996/12/23. It was written by Jana Chlebikova (`chlebik@euromath.dk`).

```
  j\'ula\or augusat\or septembra\or okt\'obra\or
  novembra\or decembra\fi
    \space \number\year}}
```

\extrasslovak \noextrasslovak The macro `\extrasslovak` will perform all the extra definitions needed for the Slovak language. The macro `\noextrasslovak` is used to cancel the actions of `\extrasslovak`. This currently means saving the meaning of one one-letter control sequence before defining it.

```
\addto\extrasslovak{\babel@save\q\let\q\v}
```

The slovak hyphenation patterns should be used with `\lefthyphenmin` set to 2 and `\righthyphenmin` set to 2.

```
\def\slovakhyphenmins{\tw@\tw@}
```

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{slovak}
⟨/code⟩
```

41 The Slovenian language

The file `slovene.dtx`[55] defines all the language-specific macros for the Slovenian language.

For this language the character `"` is made active. In table 16 an overview is given of its purpose. One of the reasons for this is that in the Slovene language some special characters are used.

`"c`	`\"c`, also implemented for the lowercase and uppercase s and z.
`"-`	an explicit hyphen sign, allowing hyphenation in the rest of the word.
`""`	like `"-`, but producing no hyphen sign (for compund words with hyphen, e.g. `x-""y`).
`"‘`	for Slovene left double quotes (looks like ,,).
`"’`	for Slovene right double quotes.
`"<`	for French left double quotes (similar to <<).
`">`	for French right double quotes (similar to >>).

Table 16: The extra definitions made by `slovene.ldf`

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

```
⟨*code⟩
\LdfInit{slovene}\captionsslovene
```

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

```
\ifx\l@slovene\@undefined
    \@nopatterns{Slovene}
    \adddialect\l@slovene0\fi
```

The next step consists of defining commands to switch to the Slovenian language. The reason for this is that a user might want to switch back and forth between languages.

`\captionsslovene` The macro `\captionsslovene` defines all strings used in the four standard documentclasses provided with LaTeX.

```
\addto\captionsslovene{%
  \def\prefacename{Predgovor}%
  \def\refname{Literatura}%
  \def\abstractname{Povzetek}%
  \def\bibname{Literatura}%
  \def\chaptername{Poglavje}%
  \def\appendixname{Dodatek}%
  \def\contentsname{Kazalo}%
```

[55]The file described in this section has version number v1.2i and was last revised on 1996/12/23. Contributions were made by Danilo Zavrtanik, University of Ljubljana (YU) and Leon Žlajpah (`leon.zlajpah@ijs.si`).

```
\def\listfigurename{Slike}%
\def\listtablename{Tabele}%
\def\indexname{Stvarno kazalo}% used to be Indeks
\def\figurename{Slika}%
\def\tablename{Tabela}%
\def\partname{Del}%
\def\enclname{Priloge}%
\def\ccname{Kopije}%
\def\headtoname{Prejme}%
\def\pagename{Stran}%
\def\seename{glej}%
\def\alsoname{glej tudi}%
\def\proofname{Dokaz}%
}%
```

`\dateslovene` The macro `\dateslovene` redefines the command `\today` to produce Slovenian dates.

```
\def\dateslovene{%
\def\today{\number\day.~\ifcase\month\or
  januar\or februar\or marec\or april\or maj\or junij\or
  julij\or avgust\or september\or oktober\or november\or december\fi
  \space \number\year}}
```

`\extrasslovene` `\noextrasslovene` The macro `\extrasslovene` performs all the extra definitions needed for the Slovenian language. The macro `\noextrasslovene` is used to cancel the actions of `\extrasslovene`.

For Slovene the `"` character is made active. This is done once, later on its definition may vary. Other languages in the same document may also use the `"` character for shorthands; we specify that the slovenian group of shorthands should be used.

```
\initiate@active@char{"}
\addto\extrasslovene{\languageshorthands{slovene}}
\addto\extrasslovene{\bbl@activate{"}}
%\addto\noextrasslovene{\bbl@deactivate{"}}
```

First we define shorthands to facilitate the occurence of letters such as č.

```
\declare@shorthand{slovene}{"c}{\textormath{\v c}{\check c}}
\declare@shorthand{slovene}{"s}{\textormath{\v s}{\check s}}
\declare@shorthand{slovene}{"z}{\textormath{\v z}{\check z}}
\declare@shorthand{slovene}{"C}{\textormath{\v C}{\check C}}
\declare@shorthand{slovene}{"S}{\textormath{\v S}{\check S}}
\declare@shorthand{slovene}{"Z}{\textormath{\v Z}{\check Z}}
```

Then we define access to two forms of quotation marks, similar to the german and french quotation marks.

```
\declare@shorthand{slovene}{"`}{%
  \textormath{\quotedblbase{}}{\mbox{\quotedblbase}}}
\declare@shorthand{slovene}{"'}{%
  \textormath{\textquotedblleft{}}{\mbox{\textquotedblleft}}}
\declare@shorthand{slovene}{"<}{%
  \textormath{\guillemotleft{}}{\mbox{\guillemotleft}}}
\declare@shorthand{slovene}{">}{%
  \textormath{\guillemotright{}}{\mbox{\guillemotright}}}
```

then we define two shorthands to be able to specify hyphenation breakpoints that behavew a little different from `\-`.

```
\declare@shorthand{slovene}{"-}{\allowhyphens-\allowhyphens}
\declare@shorthand{slovene}{""}{\hskip\z@skip}
```

And we want to have a shorthand for disabling a ligature.

```
\declare@shorthand{slovene}{"|}{%
  \textormath{\discretionary{-}{}{\kern.03em}}{}}
```

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{slovene}
⟨/code⟩
```

42 The Russian language

The file `russianb.dtx`[56] defines all the language-specific macros for the Russian language. It needs the file `cyrcod` for success documentation with Russian encodings (see below).

For this language the character `"` is made active. In table 17 an overview is given of its purpose.

`"|`	disable ligature at this position.
`"-`	an explicit hyphen sign, allowing hyphenation in the rest of the word.
`""`	like `"-`, but producing no hyphen sign (for compund words with hyphen, e.g. `x-""y` or some other signs as "disable/enable").
`"~`	for a compound word mark without a breakpoint.
`"=`	for a compound word mark with a breakpoint, allowing hyphenation in the composing words.
`"‘`	for German left double quotes (looks like ,,).
`"’`	for German right double quotes (looks like “).
” `"<`	for French left double quotes (looks like <<).
`">`	for French right double quotes (looks like >>).

Table 17: The extra definitions made by `russianb`

The quotes in table 17 can also be typeset by using the commands in table 18.

`\glqq`	for German left double quotes (looks like ,,).
`\grqq`	for German right double quotes (looks like “).
” `\flqq`	for French left double quotes (looks like <<).
`\frqq`	for French right double quotes (looks like >>).
`\dq`	the original quotes character (").

Table 18: More commands which produce quotes, defined by babel

The quotation marks traditionally used in Russian language were borrowed from other languages (e.g. French and German) so they keep their original names.

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

```
42.1 ⟨*code⟩
42.2 \LdfInit{russian}{captionsrussian}
```

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

[56]The file described in this section has version number ? and was last revised on ?. This file was initially derived from the original version of `german.sty`, which has some definitions for Russian. Later the definitions from `russian.sty` version 1.0b (for LaTeX 2.09), `russian.sty` version v2.5c (for LaTeX2e) and `francais.sty` version 4.5c and `germanb.sty` version 2.5c were added.

```
\ifx\l@russian\@undefined
  \@nopatterns{Russian}%
  \adddialect\l@russian0
\fi
```

Tyesetting Russian texts implies that a special set of fonts needs to be used. The current support for Russian uses a set of fonts that more or less 'match' with the computer modern fonts. Therefore we define the Local WashingtoN encoding (LWN, see the file `cyrillic.fdd`). We make sure that this encoding is known to LaTeX.

```
\input{LWNenc.def}
```

`\latinencoding` We need to know the encoding for text that is supposed to be typeset in latin text. We assume that it will be the encoding which is active at the end of the babel package. If the `fontenc` package is loaded later, then... too bad!

```
\AtEndOfPackage{\edef\latinencoding{\cf@encoding}}
```

Now we define two commands that offer the possibility to switch between cyrillic and roman encodings.

`\cyrillictext` `\latintext` The command `\cyrillictext` will switch from latin font encoding to the cyrillic font encoding, the command `\latintext` switches back. This assumes that the 'normal' font encoding is a latin one. These commands are *declarations*, for shorter peaces of text the commands `\textlatin` and `\textcyrillic` can be used.

```
\DeclareRobustCommand{\cyrillictext}{%
  \fontencoding{LWN}\selectfont
  \def\encodingdefault{LWN}}
\DeclareRobustCommand{\latintext}{%
  \fontencoding{\latinencoding}\selectfont
  \def\encodingdefault{\latinencoding}}
\let\lat\latintext
\let\cyr\cyrillictext
```

`\textcyrillic` `\textlatin` These commands take an argument which is then typeset using the requested font encoding.

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

`\captionsrussian` The macro `\captionsrussian` defines all strings used in the four standard document classes provided with LaTeX. There are the two commands: `\cyr` and `\lat` which switch on the right (Cyrillic or Latin) encoding.

```
\addto\captionsrussian{%
  \def\prefacename{%
    {\cyr \CYRP\CYRr\CYRe\CYRd\CYRi\CYRs\CYRl\CYRo\CYRv\CYRi\CYRe}}%
   %{\cyr \CYRV\CYRv\CYRe\CYRd\CYRe\CYRn\CYRi\CYRe}}%
  \def\refname{%
    {\cyr \CYRS\CYRp\CYRi\CYRs\CYRo\CYRk\space
      \CYRl\CYRi\CYRt\CYRe\CYRr\CYRa\CYRt\CYRu\CYRr\CYRy}}%
  \def\abstractname{%
    {\cyr \CYRA\CYRn\CYRn\CYRo\CYRt\CYRa\CYRc\CYRi\CYRya}}%
  \def\bibname{%
    {\cyr\CYRB\CYRi\CYRb\CYRl\CYRi\CYRo\CYRg\CYRr\CYRa\CYRf\CYRi\CYRya}}%
  \def\chaptername{%
    {\cyr \CYRG\CYRl\CYRa\CYRv\CYRa}}%
  \def\appendixname{%
    {\cyr \CYRP\CYRr\CYRi\CYRl\CYRo\CYRzh\CYRe\CYRn\CYRi\CYRe}}%
```

There are two names for the Table of Contents that are in use in Russian publications.

```
  \def\contentsname{%
```

For books this one is appropriate:

```
    {\cyr \CYRO\CYRg\CYRl\CYRa\CYRv\CYRl\CYRe\CYRn\CYRi\CYRe}}%
```

but for proceedings the following is preferred:

```
   %{\cyr \CYRS\CYRo\CYRd\CYRe\CYRr\CYRzh\CYRa\CYRn\CYRi\CYRe}}%
  \def\listfigurename{%
    {\cyr \CYRS\CYRp\CYRi\CYRs\CYRo\CYRk\space
      \CYRi\CYRl\CYRl\CYRyu\CYRs\CYRt\CYRr\CYRa\CYRc\CYRi\CYRishrt}}%
```

The List of Tables is not used so we provide an empty definition by default.

```
  \def\listtablename{%
    %\CYRS\CYRp\CYRi\CYRs\CYRo\CYRk\space
    %\CYRt\CYRa\CYRb\CYRl\CYRi\CYRc}%
    }
  \def\indexname{%
    {\cyr \CYRP\CYRr\CYRe\CYRd\CYRm\CYRe\CYRt\CYRn\CYRy\CYRishrt\space
      \CYRu\CYRk\CYRa\CYRz\CYRa\CYRt\CYRe\CYRl\CYRssgn}}%
  \def\authorname{%
    {\cyr \CYRI\CYRm\CYRe\CYRn\CYRn\CYRo\CYRishrt\space
      \CYRu\CYRk\CYRa\CYRz\CYRa\CYRt\CYRe\CYRl\CYRssgn}}%
  \def\figurename{{\cyr \CYRR\CYRi\CYRs.}}%
  \def\tablename{{\cyr \CYRT\CYRa\CYRb\CYRl\CYRi\CYRc\CYRa}}%
  \def\partname{{\cyr \CYRCH\CYRa\CYRs\CYRt\CYRssgn}}%
  \def\enclname{{\cyr \CYRv\CYRk\CYRl.}}%
  \def\ccname{{\cyr \CYRi\CYRs\CYRh.}}%
  \def\headtoname{{\cyr \CYRv\CYRh.}}%
  \def\pagename{{\cyr \CYRs.}}%
  \def\seename{{\cyr \CYRs\CYRm.}}%
  \def\alsoname{{\cyr \CYRs\CYRm.\ \CYRt\CYRa\CYRk\CYRzh\CYRe}}}
```

`\daterussian` The macro `\daterussian` redefines the command `\today` to produce Russian dates.

```
\def\month@russian{\ifcase\month\or
  \CYRya\CYRn\CYRv\CYRa\CYRr\CYRya\or
  \CYRf\CYRe\CYRv\CYRr\CYRa\CYRl\CYRya\or
  \CYRm\CYRa\CYRr\CYRt\CYRa\or
  \CYRa\CYRp\CYRr\CYRe\CYRl\CYRya\or
  \CYRm\CYRa\CYRya\or
  \CYRi\CYRyu\CYRn\CYRya\or
  \CYRi\CYRyu\CYRl\CYRya\or
  \CYRa\CYRv\CYRg\CYRu\CYRs\CYRt\CYRa\or
  \CYRs\CYRe\CYRn\CYRt\CYRya\CYRb\CYRr\CYRya\or
  \CYRo\CYRk\CYRt\CYRya\CYRb\CYRr\CYRya\or
  \CYRn\CYRo\CYRya\CYRb\CYRr\CYRya\or
  \CYRd\CYRe\CYRk\CYRa\CYRb\CYRr\CYRya\fi}
\def\daterussian{%
  \def\today{\number\day~\month@russian\space\number\year~\CYRg.}}
```

`\extrasrussian` The macro `\extrasrussian` will perform all the extra definitions needed for the Russian language. The macro `\noextrasrussian` is used to cancel the actions of `\extrasrussian`.

The first action we define is to switch to the `LWN` encoding whenever we enter 'russian'.

```
\addto\extrasrussian{\cyrillictext}
```

When the file `LWNenc.def` was processed by LaTeX it stores the current font encoding in `\latinencoding`, assuming that LaTeX uses `T1` or `OT1` as default. Therefore we switch back to `\latinencoding` whenever the russian language is no longer 'active'.

```
\addto\noextrasrussian{\latintext}
```

`\verbatim@font` In order to get verbatim text in the latin alphabet we need to change the definition of an internal LaTeX command somewhat:

```
\def\verbatim@font{%
  \normalfont
  \fontencoding\latinencoding\ttfamily}
```

In order to be able to use cyrillic letters in mathematics we need to have the package `cyrmath` available.

```
\AtEndOfPackage{\RequirePackage{cyrmath}}
```

The category code of the characters `:`, `;`, `!` and `?` is made `\active` to insert a little white space.

For Russian (as well as for German) the `"` character also is made active.

```
\initiate@active@char{:}
\initiate@active@char{;}
\initiate@active@char{!}
\initiate@active@char{?}
\initiate@active@char{"}
```

The code above is necessary because we need extra active characters. The character `"` is used as indicated in table 17.

We specify that the russian group of shorthands should be used.

```
\addto\extrasrussian{\languageshorthands{russian}}
```

These characters are 'turned on' once, later their definition may vary.

```
\addto\extrasrussian{%
  \bbl@activate{:}\bbl@activate{;}%
  \bbl@activate{!}\bbl@activate{?}%
  \bbl@activate{"}}
%\addto\noextrasrussian{%
%  \bbl@deactivate{:}\bbl@deactivate{;}%
%  \bbl@deactivate{!}\bbl@deactivate{?}%
%  \bbl@deactivate{"}}
```

`\russian@sh@;@` `\russian@sh@:@` `\russian@sh@!@` `\russian@sh@?@` We have to reduce the amount of white space before `;`, `:` and `!`. This should only happen in horizontal mode, hence the test with `\ifhmode`.

```
\declare@shorthand{russian}{;}{%
  \ifhmode
```

In horizontal mode we check for the presence of a 'space', 'unskip' if it exists and place a `0.1em` kerning.

```
    \ifdim\lastskip>\z@
      \unskip\penalty\@M\thinspace
    \else
      \thinspace
    \fi
  \fi
```

Now we can insert a `;` character.

```
  \string;}
```

Because these definitions are very similar only one is displayed in a way that the definition can be easily checked.

```
\declare@shorthand{russian}{:}{%
  \ifhmode
    \ifdim\lastskip>\z@
      \unskip\penalty\@M\thinspace
    \else
      \thinspace
    \fi
  \fi
  \string:}
```

```
\declare@shorthand{russian}{!}{%
  \ifhmode
    \ifdim\lastskip>\z@
      \unskip\penalty\@M\thinspace
    \else
      \thinspace
    \fi
  \fi
  \string!}
```

```
\declare@shorthand{russian}{?}{%
  \ifhmode
    \ifdim\lastskip>\z@
      \unskip\penalty\@M\thinspace
    \else
      \thinspace
    \fi
  \fi
  \string?}
```

`\system@sh@:@` `\system@sh@!@` `\system@sh@?@` `\system@sh@;@`

When the active characters appear in an environment where their Russian behaviour is not wanted they should give an 'expected' result. Therefore we define shorthands at system level as well.

```
\declare@shorthand{system}{:}{\string:}
\declare@shorthand{system}{!}{\string!}
\declare@shorthand{system}{?}{\string?}
\declare@shorthand{system}{;}{\string;}
```

To be able to define the function of `"`, we first define a couple of 'support' macros.

`\dq` We save the original double quote character in `\dq` to keep it available, the math accent `\"` can now be typed as `"`.

```
\begingroup \catcode`\"12
\def\x{\endgroup
  \def\@SS{\mathchar"7019 }
  \def\dq{"}}
\x
```

Now we can define the doublequote macros: german and french quotes. The french quotes are maded in Russian font so they are described in `lhrcod.sty`

```
\declare@shorthand{russian}{"`}{%
  \textormath{\quotedblbase{}}{\mbox{\quotedblbase}}}
\declare@shorthand{russian}{"'}{%
  \textormath{\kern-.07em\textquotedblleft{}}{\mbox{\textquotedblleft}}}
\declare@shorthand{russian}{"<}{%
  \textormath{\flqq}{\mbox{\flqq}}}
\declare@shorthand{russian}{">}{%
  \textormath{\frqq}{\mbox{\frqq}}}
```

and some additional commands:

```
\declare@shorthand{russian}{""}{\hskip\z@skip}
\declare@shorthand{russian}{"~}{\textormath{\leavevmode\hbox{-}}{-}}
\declare@shorthand{russian}{"=}{\penalty\@M-\hskip\z@skip}
\declare@shorthand{russian}{"|}{%
  \textormath{\penalty\@M\discretionary{-}{}{\kern.03em}%
              \allowhyphens}{}}
```

The next two macros for `"-` and `"---` have some difference. We must check whether the second token is a hyphen character:

```
\declare@shorthand{russian}{"-}{%
```

If the next token is `-`, we typeset an emdash, else—hyphen sign:

```
  \def\russian@sh@tmp{%
    \if\russian@sh@next-\expandafter\russian@sh@emdash
    \else\expandafter\russian@sh@hyphen\fi
  }%
```

TeX looks for the next sign after first `-`, the meaning of this sign it writes to `\russian@sh@next` and call `\russian@sh@tmp`

```
  \futurelet\russian@sh@next\russian@sh@tmp}
```

There are the definitions of hyphen and emdash: hyphen definition:

```
\def\russian@sh@hyphen{%
  \penalty\@M\-\allowhyphens}
```

emdash definition, there are the two parameters: we must “eat” two last hyphen signs of our emdash:

```
\def\russian@sh@emdash#1#2{%
  \ifdim\lastskip>\z@
    \unskip
  \fi
  \penalty\@M
  \hskip.2\fontdimen6\font
  \hbox to.8\fontdimen6\font{--\hss--}%
  \hskip.2\fontdimen6\font
  \ignorespaces}
```

The russian hyphenation patterns can be used with `\lefthyphenmin` and `\righthyphenmin` set to 2.

```
\def\russianhyphenmins{\tw@\tw@}
```

Now the thing `\extrasrussian` needs to do is to make sure that `\frenchspacing` is in effect. If this is not the case the execution of `\noextrasrussian` will switch it off again.

```
\addto\extrasrussian{\bbl@frenchspacing}
\addto\noextrasrussian{\bbl@nonfrenchspacing}
```

Now we add a new enumeration style for Russian manuscripts with Cyrillic letters and later on we define some math operatornames in accordance with Russian typesetting traditions.

`\Asbuk` We begin by defining `\Asbuk` which functions like `\Alph`, but produces (uppercase) cyrillic letters intead of latin ones.

```
\def\Asbuk#1{\expandafter\@Asbuk\csname c@#1\endcsname}
\def\@Asbuk#1{\ifcase#1\or
  \CYRA\or \CYRB\or \CYRV\or \CYRG\or \CYRD\or \CYRE\or \CYRZH\or
  \CYRZ\or \CYRI\or \CYRK\or \CYRL\or \CYRM\or \CYRN\or \CYRO\or
  \CYRP\or \CYRR\or \CYRS\or \CYRT\or \CYRU\or \CYRF\or \CYRH\or
  \CYRC\or \CYRCH\or \CYRSH\or \CYRSHCH\or \CYRErev\or \CYRYU\or
  \CYRYA\else\@ctrerr\fi}%
```

`\asbuk` The macro `\asbuk` is similar to `\alph`, it produces lowercase Russian letters.

```
\def\asbuk#1{\expandafter\@asbuk\csname c@#1\endcsname}
\def\@asbuk#1{\ifcase#1\or
  \CYRa\or \CYRb\or \CYRv\or \CYRg\or \CYRd\or \CYRe\or \CYRzh\or
  \CYRz\or \CYRi\or \CYRk\or \CYRl\or \CYRm\or \CYRn\or \CYRo\or
  \CYRp\or \CYRr\or \CYRs\or \CYRt\or \CYRu\or \CYRf\or \CYRh\or
  \CYRc\or \CYRch\or \CYRsh\or \CYRshch\or \CYRerev\or \CYRyu\or
  \CYRya\else\@ctrerr\fi}
```

`\mathrussian` Some math functions in Russian math books have other names: e.g. `sinh` in Russian is written as `sh` etc. So we define a number of new math operators.

```
\def\sh{\mathop{\operator@font sh}\nolimits} % same as \sinh
\def\ch{\mathop{\operator@font ch}\nolimits} % same as \cosh
\def\tg{\mathop{\operator@font tg}\nolimits} % same as \tan
\def\arctg{\mathop{\operator@font arctg}\nolimits} % same as \arctan
\def\arcctg{\mathop{\operator@font arcctg}\nolimits} %
\def\th{\mathop{\operator@font th}\nolimits} % same as \tanh
\def\ctg{\mathop{\operator@font ctg}\nolimits} % same as \cot
\def\cth{\mathop{\operator@font cth}\nolimits} % same as \coth
\def\cosec{\mathop{\operator@font cosec}\nolimits} % same as \csc
\def\Prob{\mathop{\hbox{\sfshape P}}\nolimits}
\def\nod{\mathop{\operator@font \CYRn.\CYRo.\CYRd.}\nolimits}
\def\nok{\mathop{\operator@font \CYRn.\CYRo.\CYRk.}\nolimits}
\def\Variance{\mathop{\hbox{\sfshape D}}\nolimits}
```

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{russian}
⟨/code⟩
```

43 The Lower Sorbian language

The file `lsorbian.dtx`[57] It defines all the language-specific macros for Lower Sorbian.

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

```
⟨*code⟩
\LdfInit{lsorbian}\captionslsorbian
```

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

```
\ifx\l@lsorbian\@undefined
  \@nopatterns{Lsorbian}
  \adddialect\l@lsorbian\l@usorbian\fi
```

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

`\captionslsorbian` The macro `\captionslsorbian` defines all strings used in the four standard documentclasses provided with LaTeX.

```
\addto\captionslsorbian{%
  \def\prefacename{Zawod}%
  \def\refname{Referency}%
  \def\abstractname{Abstrakt}%
  \def\bibname{Literatura}%
  \def\chaptername{Kapitl}%
  \def\appendixname{Dodawki}%
  \def\contentsname{Wop\'simje\'se}%
  \def\listfigurename{Zapis wobrazow}%
  \def\listtablename{Zapis tabulkow}%
  \def\indexname{Indeks}%
  \def\figurename{Wobraz}%
  \def\tablename{Tabulka}%
  \def\partname{\'Z\v el}%
  \def\enclname{P\'si\l oga}%
  \def\ccname{CC}%
  \def\headtoname{Komu}%
  \def\pagename{Strona}%
  \def\seename{gl.}%
  \def\alsoname{gl.~teke}%
  \def\proofname{Proof}%  <-- needs translation
  }%
```

`\newdatelsorbian` The macro `\newdatelsorbian` redefines the command `\today` to produce Lower Sorbian dates.

```
\def\newdatelsorbian{%
\def\today{\number\day.~\ifcase\month\or
januara\or februara\or m\v erca\or apryla\or maja\or junija\or
  julija\or awgusta\or septembra\or oktobra\or
```

[57]The file described in this section has version number v1.0d and was last revised on 1996/12/23. It was written by Eduard Werner (`edi@kaihh.hanse.de`).

```
  nowembra\or decembra\fi
    \space \number\year}}
```

`\olddatelsorbian` The macro `\olddatelsorbian` redefines the command `\today` to produce old-style Lower Sorbian dates.

```
\def\olddatelsorbian{%
  \def\today{\number\day.~\ifcase\month\or
    wjelikego ro\v zka\or
    ma\l ego ro\v zka\or
    nal\v etnika\or
    jat\v sownika\or
    ro\v zownika\or
    sma\v znika\or
    pra\v znika\or
    \v znje\'nca\or
    po\v znje\'nca\or
    winowca\or
    nazymnika\or
    godownika\fi \space \number\year}}
```

The default will be the new-style dates.

```
\let\datelsorbian\newdatelsorbian
```

`\extraslsorbian`
`\noextraslsorbian` The macro `\extraslsorbian` will perform all the extra definitions needed for the lsorbian language. The macro `\noextraslsorbian` is used to cancel the actions of `\extraslsorbian`. For the moment these macros are empty but they are defined for compatibility with the other language definition files.

```
\addto\extraslsorbian{}
\addto\noextraslsorbian{}
```

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{lsorbian}
⟨/code⟩
```

44 The Upper Sorbian language

The file `usorbian.dtx`[58] It defines all the language-specific macros for Upper Sorbian.

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

```
⟨*code⟩
\LdfInit{usorbian}\captionsusorbian
```

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

```
\ifx\l@usorbian\@undefined
    \@nopatterns{Usorbian}
    \adddialect\l@usorbian0\fi
```

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

`\captionsusorbian` The macro `\captionsusorbian` defines all strings used in the four standard documentclasses provided with LaTeX.

```
\addto\captionsusorbian{%
  \def\prefacename{Zawod}%
  \def\refname{Referency}%
  \def\abstractname{Abstrakt}%
  \def\bibname{Literatura}%
  \def\chaptername{Kapitl}%
  \def\appendixname{Dodawki}%
  \def\contentsname{Wobsah}%
  \def\listfigurename{Zapis wobrazow}%
  \def\listtablename{Zapis tabulkow}%
  \def\indexname{Indeks}%
  \def\figurename{Wobraz}%
  \def\tablename{Tabulka}%
  \def\partname{D\'z\v el}%
  \def\enclname{P\v r\l oha}%
  \def\ccname{CC}%
  \def\headtoname{Komu}%
  \def\pagename{Strona}%
  \def\seename{hl.}%
  \def\alsoname{hl.~te\v z}
  \def\proofname{Proof}%  <-- needs translation
  }%
```

`\newdateusorbian` The macro `\newdateusorbian` redefines the command `\today` to produce Upper Sorbian dates.

```
\def\newdateusorbian{%
\def\today{\number\day.~\ifcase\month\or
januara\or februara\or m\v erca\or apryla\or meje\or junija\or
  julija\or awgusta\or septembra\or oktobra\or
```

[58]The file described in this section has version number v1.0e and was last revised on 1996/12/23. It was written by Eduard Werner (`edi@kaihh.hanse.de`).

```
nowembra\or decembra\fi
  \space \number\year}}
```

\olddateusorbian The macro `\olddateusorbian` redefines the command `\today` to produce old-style Upper Sorbian dates.

```
\def\olddateusorbian{%
\def\today{\number\day.~\ifcase\month\or
 wulkeho r\'o\v zka\or ma\l eho r\'o\v zka\or nal\v etnika\or
 jutrownika\or r\'o\v zownika\or  sma\v znika\or pra\v znika\or
 \v znjenca\or po\v znjenca\or winowca\or nazymnika\or
 hodownika\fi \space \number\year}}
```

The default will be the new-style dates.

```
\let\dateusorbian\newdateusorbian
```

\extrasusorbian The macro `\extrasusorbian` will perform all the extra definitions needed for the Upper Sorbian language. It's pirated from `germanb.sty`. The macro `\noextrasusorbian` is used to cancel the actions of `\extrasusorbian`.

Because for Upper Sorbian (as well as for Dutch) the `"` character is made active. This is done once, later on its definition may vary.

```
\initiate@active@char{"}
\addto\extrasusorbian{\languageshorthands{usorbian}}
\addto\extrasusorbian{\bbl@activate{"}}
%\addto\noextrasusorbian{\bbl@deactivate{"}}
```

In order for TEX to be able to hyphenate German Upper Sorbian words which contain 'ß' we have to give the character a nonzero `\lccode` (see Appendix H, the TEXbook).

```
\addto\extrasusorbian{\babel@savevariable{\lccode`\^^Y}%
  \lccode`\^^Y`\^^Y}
```

The umlaut accent macro `\"` is changed to lower the umlaut dots. The redefinition is done with the help of `\umlautlow`.

```
\addto\extrasusorbian{\babel@save\"\umlautlow}
\addto\noextrasusorbian{\umlauthigh}
```

The Upper Sorbian hyphenation patterns can be used with `\lefthyphenmin` and `\righthyphenmin` set to 2.

```
\def\usorbianhyphenmins{\tw@\tw@}
```

\dq We save the original double quote character in `\dq` to keep it available, the math accent `\"` can now be typed as `"`. Also we store the original meaning of the command `\"` for future use.

```
\begingroup \catcode`\"12
\def\x{\endgroup
  \def\@SS{\mathchar"7019 }
  \def\dq{"}}
\x
```

Now we can define the doublequote macros: the umlauts,

```
\declare@shorthand{usorbian}{"a}{\textormath{\"{a}}{\ddot a}}
\declare@shorthand{usorbian}{"o}{\textormath{\"{o}}{\ddot o}}
\declare@shorthand{usorbian}{"u}{\textormath{\"{u}}{\ddot u}}
```

```
\declare@shorthand{usorbian}{"A}{\textormath{\"{A}}{\ddot A}}
\declare@shorthand{usorbian}{"O}{\textormath{\"{O}}{\ddot O}}
\declare@shorthand{usorbian}{"U}{\textormath{\"{U}}{\ddot U}}
```

tremas,

```
\declare@shorthand{usorbian}{"e}{\textormath{\"{e}}{\ddot e}}
\declare@shorthand{usorbian}{"E}{\textormath{\"{E}}{\ddot E}}
\declare@shorthand{usorbian}{"i}{\textormath{\"{\i}}{\ddot\imath}}
\declare@shorthand{usorbian}{"I}{\textormath{\"{I}}{\ddot I}}
```

usorbian es-zet (sharp s),

```
\declare@shorthand{usorbian}{"s}{\textormath{\ss{}}{\@SS{}}}
\declare@shorthand{usorbian}{"S}{SS}
```

german and french quotes,

```
\declare@shorthandusorbian{}{"`}{%
  \textormath{\quotedblbase{}}{\mbox{\quotedblbase}}}
\declare@shorthand{usorbian}{"'}{%
  \textormath{\textquotedblleft{}}{\mbox{\textquotedblleft}}}
\declare@shorthand{usorbian}{"<}{%
  \textormath{\guillemotleft{}}{\mbox{\guillemotleft}}}
\declare@shorthand{usorbian}{">}{%
  \textormath{\guillemotright{}}{\mbox{\guillemotright}}}
```

discretionary commands

```
\declare@shorthand{usorbian}{"c}{\textormath{\bbl@disc ck}{c}}
\declare@shorthand{usorbian}{"C}{\textormath{\bbl@disc CK}{C}}
\declare@shorthand{usorbian}{"f}{\textormath{\bbl@disc f{ff}}{f}}
\declare@shorthand{usorbian}{"F}{\textormath{\bbl@disc F{FF}}{F}}
\declare@shorthand{usorbian}{"l}{\textormath{\bbl@disc l{ll}}{l}}
\declare@shorthand{usorbian}{"L}{\textormath{\bbl@disc L{LL}}{L}}
\declare@shorthand{usorbian}{"m}{\textormath{\bbl@disc m{mm}}{m}}
\declare@shorthand{usorbian}{"M}{\textormath{\bbl@disc M{MM}}{M}}
\declare@shorthand{usorbian}{"n}{\textormath{\bbl@disc n{nn}}{n}}
\declare@shorthand{usorbian}{"N}{\textormath{\bbl@disc N{NN}}{N}}
\declare@shorthand{usorbian}{"p}{\textormath{\bbl@disc p{pp}}{p}}
\declare@shorthand{usorbian}{"P}{\textormath{\bbl@disc P{PP}}{P}}
\declare@shorthand{usorbian}{"t}{\textormath{\bbl@disc t{tt}}{t}}
\declare@shorthand{usorbian}{"T}{\textormath{\bbl@disc T{TT}}{T}}
```

and some additional commands:

```
\declare@shorthand{usorbian}{"-}{\penalty\@M\-\allowhyphens}
\declare@shorthand{usorbian}{"|}{%
  \textormath{\penalty\@M\discretionary{-}{}{\kern.03em}%
              \allowhyphens}{}}
\declare@shorthand{usorbian}{""}{\hskip\z@skip}
```

`\mdqon` `\mdqoff` `\ck` All that's left to do now is to define a couple of commands for reasons of compatibility with `german.sty`.

```
\def\mdqon{\bbl@activate{"}}
\def\mdqoff{\bbl@deactivate{"}}
\def\ck{\allowhyphens\discretionary{k-}{k}{ck}\allowhyphens}
```

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{usorbian}
⟨/code⟩
```

45 The Turkish language

The file `turkish.dtx`[59] defines all the language definition macros for the Turkish language[60].

Turkish typographic rules specify that a little 'white space' should be added before the characters '`:`', '`!`' and '`=`'. In order to insert this white space automatically these characters are made 'active'. Also `\frenhspacing` is set.

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

```
⟨*code⟩
\LdfInit{turkish}\captionsturkish
```

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

```
\ifx\l@turkish\@undefined
  \@nopatterns{Turkish}
  \adddialect\l@turkish0\fi
```

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

`\captionsturkish` The macro `\captionsturkish` defines all strings used in the four standard documentclasses provided with LaTeX.

```
\addto\captionsturkish{%
  \def\prefacename{Preface}% <-- This needs translation!!
  \def\refname{Ba\c svurulan Kitaplar}%
  \def\abstractname{Konu}%
  \def\bibname{Bibliografi}%
  \def\chaptername{Anab\"ol\"um}%
  \def\appendixname{Appendix}%
  \def\contentsname{\.I\c cindekiler}%
  \def\listfigurename{\c Sekiller Listesi}%
  \def\listtablename{Tablolar\i{}n Listesi}%
  \def\indexname{\.Index}%
  \def\figurename{\c Sekiller}%
  \def\tablename{Tablo}%
  \def\partname{B\"ol\"um}%
  \def\enclname{Ekler}%
  \def\ccname{G\"onderen}%
  \def\headtoname{Al\i{}c\i}%
  \def\pagename{Sayfa}%
  \def\subjectname{To}% <-- This needs translation!!
  \def\seename{see}% <-- This needs translation!!
  \def\alsoname{see also}% <-- This needs translation!!
  \def\proofname{Proof}% <-- This needs translation!!
}%
```

[59]The file described in this section has version number v1.2i and was last revised on 1996/12/23.
[60]Mustafa Burc, `z6001@rziris01.rrz.uni-hamburg.de` provided the code for this file. It is based on the work by Pierre Mackay

`\dateturkish` The macro `\dateturkish` redefines the command `\today` to produce Turkish dates.

```
\def\dateturkish{%
  \def\today{\number\day.~\ifcase\month\or
    Ocak\or \c Subat\or Mart\or Nisan\or May\i{}s\or Haziran\or
    Temmuz\or A\u gustos\or Eyl\"ul\or Ekim\or Kas\i{}m\or
    Aral\i{}k\fi
    \space\number\year}}
```

`\extrasturkish` `\noextrasturkish` The macro `\extrasturkish` will perform all the extra definitions needed for the Turkish language. The macro `\noextrasturkish` is used to cancel the actions of `\extrasturkish`.

Turkish typographic rules specify that a little 'white space' should be added before the characters '`:`', '`!`' and '`=`'. In order to insert this white space automatically these characters are made `\active`, so they have to be treated in a special way.

```
\initiate@active@char{:}
\initiate@active@char{!}
\initiate@active@char{=}
```

We specify that the turkish group of shorthands should be used.

```
\addto\extrasturkish{\languageshorthands{turkish}}
```

These characters are 'turned on' once, later their definition may vary.

```
\addto\extrasturkish{%
  \bbl@activate{:}\bbl@activate{!}\bbl@activate{=}}
```

For Turkish texts `\frenchspacing` should be in effect. We make sure this is the case and reset it if necessary.

```
\addto\extrasturkish{\bbl@frenchspacing}
\addto\noextrasturkish{\bbl@nonfrenchspacing}
```

`\turkish@sh@!@` `\turkish@sh@=@` `\turkish@sh@:@` The definitions for the three active characters were made using intermediate macros. These are defined now. The insertion of extra 'white space' should only happen outside math mode, hence the check `\ifmmode` in the macros.

```
\declare@shorthand{turkish}{:}{%
  \ifmmode
    \string:%
  \else\relax
    \ifhmode
      \ifdim\lastskip>\z@
        \unskip\penalty\@M\thinspace
      \fi
    \fi
    \string:%
  \fi}
\declare@shorthand{turkish}{!}{%
  \ifmmode
    \string!%
  \else\relax
    \ifhmode
      \ifdim\lastskip>\z@
        \unskip\penalty\@M\thinspace
      \fi
    \fi
    \string!%
  \fi}
\declare@shorthand{turkish}{=}{%
  \ifmmode
    \string=%
  \else\relax
    \ifhmode
      \ifdim\lastskip>\z@
        \unskip\kern\fontdimen2\font
        \kern-1.4\fontdimen3\font
      \fi
    \fi
    \string=%
  \fi}
```

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{turkish}
⟨/code⟩
```

46 The Bahasa language

The file `bahasa.dtx`[61] defines all the language definition macros for the bahasa indonesia / bahasa melayu language. Bahasa just means 'language' in bahasa indonesia / bahasa melayu. Since both national versions of the language use the same writing, although differing in pronounciation, this file can be used for both languages.

For this language currently no special definitions are needed or available.

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

```
⟨*code⟩
\LdfInit{bahasa}\captionsbahasa
```

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

```
\ifx\l@bahasa\@undefined
  \@nopatterns{Bahasa}
  \adddialect\l@bahasa0\fi
```

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

`\captionsbahasa` The macro `\captionsbahasa` defines all strings used in the four standard documentclasses provided with LaTeX.

```
\addto\captionsbahasa{%
  \def\prefacename{Pendahuluan}%
  \def\refname{Pustaka}%
  \def\abstractname{Ringkasan}% (sometime it's called 'intisari'
                              %  or 'ikhtisar')
  \def\bibname{Bibliografi}%
  \def\chaptername{Bab}%
  \def\appendixname{Lampiran}%
  \def\contentsname{Daftar Isi}%
  \def\listfigurename{Daftar Gambar}%
  \def\listtablename{Daftar Tabel}%
% Glossary: Daftar Istilah
  \def\indexname{Indeks}%
  \def\figurename{Gambar}%
  \def\tablename{Tabel}%
  \def\partname{Bagian}%
%  Subject:  Subyek
%  From:  Dari
  \def\enclname{Lampiran}%
  \def\ccname{cc}%
  \def\headtoname{Kepada}%
  \def\pagename{Halaman}%
%  Notes (Endnotes): Catatan
  \def\seename{lihat}%
  \def\alsoname{lihat juga}%
  \def\proofname{Bukti}%
```

[61]The file described in this section has version number v1.0e and was last revised on 1996/12/23.

```
  }
```

\datebahasa The macro \datebahasa redefines the command \today to produce Bahasa dates.

```
\def\datebahasa{%
  \def\today{\number\day~\ifcase\month\or
    Januari\or Februari\or Maret\or April\or Mei\or Juni\or
    Juli\or Agustus\or September\or Oktober\or Nopember\or Desember\fi
    \space \number\year}}
```

\extrasbahasa
\noextrasbahasa The macro \extrasbahasa will perform all the extra definitions needed for the Bahasa language. The macro \extrasbahasa is used to cancel the actions of \extrasbahasa. For the moment these macros are empty but they are defined for compatibility with the other language definition files.

```
\addto\extrasbahasa{}
\addto\noextrasbahasa{}
```

\bahasahyphenmins The bahasa hyphenation patterns should be used with \lefthyphenmin set to 2 and \righthyphenmin set to 2.

```
\def\bahasahyphenmins{\tw@\tw@}
```

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{bahasa}
⟨/code⟩
```

47 Support for formats based on plainTeX

The following code duplicates or emulates parts of LaTeX 2$_\varepsilon$ that are needed for babel.

```
⟨*code⟩
\ifx\adddialect\@undefined
```

When `\adddialect` is still undefined we are making a format. In that case only the first part of this file is needed.

```
  \def\@empty{}
```

We need to define `\loadlocalcfg` for plain users as the LaTeX definition uses `\InputIfFileExists`.

```
  \def\loadlocalcfg#1{%
    \openin0#1.cfg
    \ifeof0
      \closein0
    \else
      \closein0
      {\immediate\write16{*************************************}%
       \immediate\write16{* Local config file #1.cfg used}%
       \immediate\write16{*}%
       }
      \input #1.cfg\relax
    \fi
```

We have to execute `\@endofldf` in this case

```
    \@endofldf
    }
```

We want to add a message to the message LaTeX 2.09 puts in the `\everyjob` register. This could be done by the following code:

```
\let\orgeveryjob\everyjob
\def\everyjob#1{%
  \orgeveryjob{#1}%
  \orgeveryjob\expandafter{\the\orgeveryjob\immediate\write16{%
      hyphenation patterns for \the\loaded@patterns loaded.}}%
  \let\everyjob\orgeveryjob\let\orgeveryjob\@undefined}
```

The code above redefines the control sequence `\everyjob` in order to be able to add something to the current contents of the register. This is necessary because the processing of hyphenation patterns happens long before LaTeX fills the register. There are some problems with this approach though.

- When someone wants to use several hyphenation patterns with SLiTeX the above scheme won't work. The reason is that SLiTeX overwrites the contents of the `\everyjob` register with its own message.
- Plain TeX does not use the `\everyjob` register so the message would not be displayed.

To circumvent this a 'dirty trick' can be used. As this code is only processed when creating a new format file there is one command that is sure to be used,

`\dump`. Therefore the orginal `\dump` is saved in `\org@dump` and a new definition is supplied.

```
  \let\orig@dump=\dump
  \def\dump{%
```

To make sure that LaTeX 2.09 executes the `\@begindocumenthook` we would want to alter `\begin{document}`, but as this done too often already, we add the new code at the front of `\@preamblecmds`. But we can only do that after it has been defined, so we add this peice of code to `\dump`.

```
    \ifx\@ztryfc\@undefined
    \else
      \toks0=\expandafter{\@preamblecmds}
      \edef\@preamblecmds{\noexpand\@begindocumenthook\the\toks0}
      \def\@begindocumenthook{}
    \fi
```

This new definition starts by adding an instruction to write a message on the terminal and in the transcript file to inform the user of the preloaded hyphenation patterns.

```
    \everyjob\expandafter{\the\everyjob%
      \immediate\write16{\the\toks8 loaded.}}%
```

Then everything is restored to the old situation and the format is dumped.

```
    \let\dump\orig@dump\let\orig@dump\@undefined\dump}
  \expandafter\endinput
\fi
```

The rest of this file is not processed by iniTeX but during the normal document run. A number of LaTeX macro's that are needed later on.

```
\long\def\@firstofone#1{#1}
\long\def\@firstoftwo#1#2{#1}
\def\@ifstar#1{\@ifnextchar *{\@firstoftwo{#1}}}
\def\@star@or@long#1{%
  \@ifstar
  {\let\l@ngrel@x\relax#1}%
  {\let\l@ngrel@x\long#1}}
\let\l@ngrel@x\relax
\def\@car#1#2\@nil{#1}
\def\@cdr#1#2\@nil{#2}
\let\@typeset@protect\relax
\long\def\@gobble#1{}
\edef\@backslashchar{\expandafter\@gobble\string\\}
\def\strip@prefix#1>{}
\def\g@addto@macro#1#2{{%
    \toks@\expandafter{#1#2}%
    \xdef#1{\the\toks@}}}
\def\@namedef#1{\expandafter\def\csname #1\endcsname}
```

LaTeX 2_ε has the command `\@onlypreamble` which adds commands to a list of commands that are no longer needed after `\begin{document}`.

```
\ifx\@preamblecmds\@undefined
  \def\@preamblecmds{}
\fi
\def\@onlypreamble#1{%
  \expandafter\gdef\expandafter\@preamblecmds\expandafter{%
```

```
    \@preamblecmds\do#1}}
\@onlypreamble\@onlypreamble
```

Mimmick LaTeX's `\AtBeginDocument`; for this to work the user needs to add `\begindocument` to his file.

```
\def\begindocument{%
  \@begindocumenthook
  \global\let\@begindocumenthook\@undefined
  \def\do##1{\global\let ##1\@undefined}%
  \@preamblecmds
  \global\let\do\noexpand
  }
```

```
\ifx\@begindocumenthook\@undefined
  \def\@begindocumenthook{}
\fi
\@onlypreamble\@begindocumenthook
\def\AtBeginDocument{\g@addto@macro\@begindocumenthook}
```

We also have to mimick LaTeX's `\AtEndOfPackage`. Our replacement macro is much simpler; it stores its argument in `\@endofldf`.

```
\def\AtEndOfPackage#1{\g@addto@macro\@endofldf{#1}}
\@onlypreamble\AtEndOfPackage
\def\@endofldf{}
\@onlypreamble\@endofldf
```

LaTeX needs to be able to switch off writing to its auxiliary files; plain doesn't have them by default.

```
\ifx\bye\@undefined\else
\expoandafter\let\csname if@filesw\expandafter\csname iffalse\endcsname
\fi
```

Mimick LaTeX's commands to define control sequences.

```
\def\newcommand{\@star@or@long\new@command}
\def\new@command#1{%
  \@testopt{\@newcommand#1}0}
\def\@newcommand#1[#2]{%
  \@ifnextchar [{\@xargdef#1[#2]}%
                {\@argdef#1[#2]}}
\long\def\@argdef#1[#2]#3{%
  \@yargdef#1\@ne{#2}{#3}}
\long\def\@xargdef#1[#2][#3]#4{%
  \expandafter\def\expandafter#1\expandafter{%
    \expandafter\@protected@testopt\expandafter #1%
    \csname\string#1\expandafter\endcsname{#3}}%
  \expandafter\@yargdef \csname\string#1\endcsname
  \tw@{#2}{#4}}
\long\def\@yargdef#1#2#3{%
  \@tempcnta#3\relax
  \advance \@tempcnta \@ne
  \let\@hash@\relax
  \edef\reserved@a{\ifx#2\tw@ [\@hash@1]\fi}%
  \@tempcntb #2%
  \@whilenum\@tempcntb <\@tempcnta
  \do{%
    \edef\reserved@a{\reserved@a\@hash@\the\@tempcntb}%
```

```
   \advance\@tempcntb \@ne}%
  \let\@hash@##%
  \l@ngrel@x\expandafter\def\expandafter#1\reserved@a}
\let\providecommand\newcommand

\def\DeclareRobustCommand{\@star@or@long\declare@robustcommand}
\def\declare@robustcommand#1{%
   \edef\reserved@a{\string#1}%
   \def\reserved@b{#1}%
   \edef\reserved@b{\expandafter\strip@prefix\meaning\reserved@b}%
   \edef#1{%
      \ifx\reserved@a\reserved@b
          \noexpand\x@protect
          \noexpand#1%
      \fi
      \noexpand\protect
      \expandafter\noexpand\csname
         \expandafter\@gobble\string#1 \endcsname
   }%
   \expandafter\new@command\csname
      \expandafter\@gobble\string#1 \endcsname
}
\def\x@protect#1{%
   \ifx\protect\@typeset@protect\else
      \@x@protect#1%
   \fi
}
\def\@x@protect#1\fi#2#3{%
   \fi\protect#1%
}
```

LaTeX has a macro to check whether a certain package was loaded with specific options. The command has two extra arguments which are code to be executed in either the true or false case. This is used to detect whether the document needs one of the accents to be activated (activegrave and activeacute). For plain TeX we assume that the user wants them to be active by default. Therefore the only thing we do is execute the third argument (the code for the true case).

```
\def\@ifpackagewith#1#2#3#4{%
  #3}
```

For the following code we need to make sure that the commands `\newcommand` and `\providecommand` exist with some sensible definition. They are not fully equivalent to their LaTeX 2_ε versions; just enough to make things work in plain TeXenvironments.

```
\ifx\@tempcnta\@undefined
  \csname newcount\endcsname\@tempcnta\relax
\fi
\ifx\@tempcntb\@undefined
  \csname newcount\endcsname\@tempcntb\relax
\fi
```

To prevent wasting two counters in LaTeX 2.09 (because counters with the same name are allocated later by it) we reset the counter that holds the next free counter (`\count10`).

```
\ifx\bye\@undefined
 \advance\count10 by -2\relax
\fi
\ifx\@ifnextchar\@undefined
  \def\@ifnextchar#1#2#3{%
    \let\reserved@d=#1%
    \def\reserved@a{#2}\def\reserved@b{#3}%
    \futurelet\@let@token\@ifnch}
  \def\@ifnch{%
    \ifx\@let@token\@sptoken
      \let\reserved@c\@xifnch
    \else
      \ifx\@let@token\reserved@d
        \let\reserved@c\reserved@a
      \else
        \let\reserved@c\reserved@b
      \fi
    \fi
    \reserved@c}
  \def\:{\let\@sptoken= } \:  % this makes \@sptoken a space token
  \def\:{\@xifnch} \expandafter\def\: {\futurelet\@let@token\@ifnch}
\fi
\def\@testopt#1#2{%
  \@ifnextchar[{#1}{#1[#2]}}
\def\@protected@testopt#1{%%
  \ifx\protect\@typeset@protect
    \expandafter\@testopt
  \else
    \@x@protect#1%
  \fi}
\long\def\@whilenum#1\do #2{\ifnum #1\relax #2\relax\@iwhilenum{#1\relax
     #2\relax}\fi}
\long\def\@iwhilenum#1{\ifnum #1\expandafter\@iwhilenum
         \else\expandafter\@gobble\fi{#1}}
```

Code from `ltoutenc.dtx`, adapted for use in the plain TEX environment.

```
\def\DeclareTextCommand{%
   \@dec@text@cmd\providecommand
}
\def\ProvideTextCommand{%
   \@dec@text@cmd\providecommand
}
\def\DeclareTextSymbol#1#2#3{%
   \@dec@text@cmd\chardef#1{#2}#3\relax
}
\def\@dec@text@cmd#1#2#3{%
   \expandafter\def\expandafter#2%
      \expandafter{%
         \csname#3-cmd\expandafter\endcsname
         \expandafter#2%
         \csname#3\string#2\endcsname
      }%
%   \let\@ifdefinable\@rc@ifdefinable
   \expandafter#1\csname#3\string#2\endcsname
}
```

```
\def\@current@cmd#1{%
  \ifx\protect\@typeset@protect\else
      \noexpand#1\expandafter\@gobble
  \fi
}
\def\@changed@cmd#1#2{%
   \ifx\protect\@typeset@protect
      \expandafter\ifx\csname\cf@encoding\string#1\endcsname\relax
         \expandafter\ifx\csname ?\string#1\endcsname\relax
            \expandafter\def\csname ?\string#1\endcsname{%
               \@changed@x@err{#1}%
            }%
         \fi
         \global\expandafter\let
           \csname\cf@encoding \string#1\expandafter\endcsname
           \csname ?\string#1\endcsname
      \fi
      \csname\cf@encoding\string#1%
        \expandafter\endcsname
   \else
      \noexpand#1%
   \fi
}
\def\@changed@x@err#1{%
    \errhelp{Your command will be ignored, type <return> to proceed}%
    \errmessage{Command \protect#1 undefined in encoding \cf@encoding}}
\def\DeclareTextCommandDefault#1{%
   \DeclareTextCommand#1?%
}
\def\ProvideTextCommandDefault#1{%
   \ProvideTextCommand#1?%
}
\expandafter\let\csname OT1-cmd\endcsname\@current@cmd
\expandafter\let\csname?-cmd\endcsname\@changed@cmd
\def\DeclareTextAccent#1#2#3{%
  \DeclareTextCommand#1{#2}[1]{\accent#3 ##1}
}
\def\DeclareTextCompositeCommand#1#2#3#4{%
   \expandafter\let\expandafter\reserved@a\csname#2\string#1\endcsname
   \edef\reserved@b{\string##1}%
   \edef\reserved@c{%
     \expandafter\@strip@args\meaning\reserved@a:-\@strip@args}%
   \ifx\reserved@b\reserved@c
      \expandafter\expandafter\expandafter\ifx
         \expandafter\@car\reserved@a\relax\relax\@nil
         \@text@composite
      \else
         \edef\reserved@b##1{%
            \def\expandafter\noexpand
               \csname#2\string#1\endcsname####1{%
               \noexpand\@text@composite
                  \expandafter\noexpand\csname#2\string#1\endcsname
                  ####1\noexpand\@empty\noexpand\@text@composite
                  {##1}%
            }%
         }%
         \expandafter\reserved@b\expandafter{\reserved@a{##1}}%
      \fi
      \expandafter\def\csname\expandafter\string\csname
         #2\endcsname\string#1-\string#3\endcsname{#4}
   \else
     \errhelp{Your command will be ignored, type <return> to proceed}%
     \errmessage{\string\DeclareTextCompositeCommand\space used on
         inappropriate command \protect#1}
   \fi
}
\def\@text@composite#1#2#3\@text@composite{%
   \expandafter\@text@composite@x
      \csname\string#1-\string#2\endcsname
}
\def\@text@composite@x#1#2{%
   \ifx#1\relax
      #2%
   \else
      #1%
   \fi
}
%
\def\@strip@args#1:#2-#3\@strip@args{#2}
\def\DeclareTextComposite#1#2#3#4{%
   \def\reserved@a{\DeclareTextCompositeCommand#1{#2}{#3}}%
   \bgroup
      \lccode`\@=#4%
      \lowercase{%
   \egroup
      \reserved@a @%
   }%
}
%
\def\UseTextSymbol#1#2{%
%   \let\@curr@enc\cf@encoding
%   \@use@text@encoding{#1}%
   #2%
%   \@use@text@encoding\@curr@enc
}
\def\UseTextAccent#1#2#3{%
%   \let\@curr@enc\cf@encoding
%   \@use@text@encoding{#1}%
%   #2{\@use@text@encoding\@curr@enc\selectfont#3}%
%   \@use@text@encoding\@curr@enc
}
\def\@use@text@encoding#1{%
%   \edef\f@encoding{#1}%
%   \xdef\font@name{%
%      \csname\curr@fontshape/\f@size\endcsname
%   }%
%   \pickup@font
%   \font@name
%    \@@enc@update
}
\def\DeclareTextSymbolDefault#1#2{%
    \DeclareTextCommandDefault#1{\UseTextSymbol{#2}#1}%
}
\def\DeclareTextAccentDefault#1#2{%
    \DeclareTextCommandDefault#1{\UseTextAccent{#2}#1}%
}
\def\cf@encoding{OT1}
```

Currently we only use the LaTeX 2_ε method for accents for those that are known to be made active in *some* language definition file.

```
\DeclareTextAccent{\"}{OT1}{127}
\DeclareTextAccent{\'}{OT1}{19}
\DeclareTextAccent{\^}{OT1}{94}
\DeclareTextAccent{\`}{OT1}{18}
\DeclareTextAccent{\~}{OT1}{126}
```

The following two control sequences are used in `babel.def` but are not defined for PLAIN TeX.

```
\DeclareTextSymbol{\textquotedblright}{OT1}{`\"}
\DeclareTextSymbol{\textquoteright}{OT1}{`\'}
\DeclareTextSymbol{\i}{OT1}{16}
\DeclareTextSymbol{\ss}{OT1}{25}
```

For a couple of languages we need the LaTeX-control sequence `\scriptsize` to be available. Because plain TeX doesn't have such a sofisticated font mechanism as LaTeX has, we just `\let` it to `\sevenrm`.

```
\ifx\scriptsize\@undefined
  \let\scriptsize\sevenrm
\fi
⟨/code⟩
```

Index

Numbers written in italic refer to the page where the corresponding entry is described, the ones underlined to the code line of the definition, the rest to the code lines where the entry is used.

Change History

\selectlanguage: \originalTeX should only be executed once 17

babel 3.2

General: Some Changes by br 1

\adddialect: Added \relax 15

\addlanguage: Added a %, removed by 15

\babel@beginsave: Added macro 33

\babel@save: Added macro 33

\babel@savecnt: Added macro 33

\babel@savevariable: Added macro 33

\bbl@add@special: Added macro 25

\bbl@remove@special: Added macro 25

\iflanguage: Refrased \ifnum test 15

\selectlanguage: Modified to allow arguments that start with an escape character 16

babel 3.2a

General: Fixups of the code and documentation 1

\originalTeX: Set \originalTeX to \empty, because it should be expandable. 21

\readconfigfile: Free macro space for \process@language 24

\selectlanguage: Added \relax as first command to stop an expansion if \protect is empty 17

Added three \expandafters to save macro space for \originalTeX 17

Moved definition of \originalTeX before \extras⟨*lang*⟩ 17

Set \originalTeX to \empty, because it should be expandable. 17

Simplified the modification to allow the use in a \write command 16

babel 3.2b

\allowhyphens: Moved macro from language definition files 34

\save@sf@q: Moved macro from language definition files 35

\set@low@box: Moved macro from language definition files 35

babel 3.2c

\babel@save: missing backslash led to errors when executing \originalTeX . . 33

babel 3.2d

\babel@save: saving in \babel@i and restoring from \@babel@i doesn't work very well... 33

babel 3.2e

General: Added slovak 52

babel 3.3

General: \headpagename should be \pagename 44

Added catalan and galician 52

Added turkish 52

Included driver file, and prepared for dsitribution 1

babel 3.4

General: Added bahasa 52

Added language definition file for bahasa 1

Updated for LaTeX 2ε 1

\addto: Changed to use toks register 34

babel 3.4b

General: Added a small driver to be able to process just this file 1

Use the ltxdoc class instead of article 50

babel 3.4c

General: lhyphen.cfg has become lthyphen.cfg 11

babel 3.4e

\@nolanerr: Use \PackageError in LaTeX 2ε mode 21

\@nopatterns: Macro added 21

catalan-2.2a
  General: All the code to deal with active characters is now in `babel.def` . . . 116
  `\extrascatalan`: Handling of active characters completely rewritten 114
  `\noextrascatalan`: All the code for handling active characters is now moved to `babel.def` 114
catalan-2.2b
  General: Changed mathmode definition of acute shorthands to expand to a single prime followed by the next character in the input 116
    Made the activation of the grave and acute accents optional 112
  `\captionscatalan`: Added `\proofname` for AMS-LaTeX 113
  `\datecatalan`: Month names in lowercase 113
  `\Lgem`: Added support for typing the catalan "ela geminada" with the macros `\lgem` and `\Lgem` 117
  `\noextrascatalan`: Make activating the accent characters optional 114
  `\up`: Added definition of macro `\up`, which can be used to type ordinals 118
catalan-2.2c
  General: Added " as an axtra shorthand, removed 'n as a shorthand 116
    Added shorthands for guillemets 116
    cedile now produced by double quote shorthand 116
    Removed the use of the tilde for catalan 112
catalan-2.2d
  `\captionscatalan`: added translation of Proof 113
    Translations revised 113
catalan-2.2e
  General: Added " as an axtra shorthand 116
    Added vertical bar as argument to active acute 116
  `\L.L`: Added redefinition of `\l` and `\L` 118
  `\noextrascatalan`: Need to split up the `\@ifpackagewith` statements 114
    Now give the apostrophe a lowercase code 114
  `\up`: Now use `\textsuperscript` and make `\up` robust 118
catalan-2.2f
  General: Replaced `\undefined` with `\@undefined` and `\empty` with `\@empty` for consistency with LaTeX 112
  `\Lgem`: Added a check for math mode as the use of `\lgem` and `\Lgem` in math mode is not sensible. 117
catalan-2.2g
  General: Moved the definition of `\atcatcode` right to the beginning. 112
    Now use `\ldf@finish` to wrap up 118
    Now use `\LdfInit` to perform initial checks 113
catalan-2.2h
  `\noextrascatalan`: Added some comment signs to prevent unwanted spaces in the output 114
croatian-1.0a
  General: Renamed `babel.sty` in `babel.com` 146
croatian-1.0b
  General: Removed use of `\@ifundefined` 146
croatian-1.0c
  General: Removed some typos 146
croatian-1.1
  General: Added a warning when no hyphenation patterns were loaded. 146
    Brought up-to-date with babel 3.2a 146
  `\captionscroatian`: Added `\seename`, `\alsoname` and `\prefacename` 146
croatian-1.2
  `\captionscroatian`: `\headpagename` should be `\pagename` 146

Now use `\adddialect` if language undefined 63
english-3.0a
General: Removed bug found by van der Meer 63
english-3.0b
General: Removed `\global` definitions 63
`\captionsenglish`: `\pagename` should be `\headpagename` 64
Removed `\global` definitions 64
`\dateamerican`: Removed `\global` definitions 64
`\dateenglish`: Removed `\global` definitions 64
english-3.0c
General: Renamed `babel.sty` in `babel.com` 63
english-3.0d
General: removed use of `\@ifundefined` 63
english-3.1
General: Rewrote parts of the code to use the new features of babel version 3.1 63
english-3.1a
`\captionsenglish`: added `\seename` and `\alsoname` 64
english-3.1b
`\captionsenglish`: added `\prefacename` 64
english-3.2
`\captionsenglish`: `\headpagename` should be `\pagename` 64
english-3.3
General: Update or LaTeX 2_ε 63
english-3.3c
General: Now use `\@nopatterns` to produce the warning 63
Removed the use of `\filedate` and moved the identification after the loading of `babel.def` 63
english-3.3d
General: Only define american as a dialect when no separate patterns have been loaded 63
english-3.3e
`\captionsenglish`: Added `\proofname` for AMS-LaTeX 64
english-3.3g
General: Allow british as the name of the UK patterns 63
Allow USenglish as the name of the american patterns 63
Replaced `\undefined` with `\@undefined` and `\empty` with `\@empty` for consistency with LaTeX 63
`\captionsenglish`: Construct control sequence on the fly 64
`\dateenglish`: Construct control sequence on the fly 64
`\noextrasenglish`: Construct control sequences on the fly 64
english-3.3h
General: Moved the definition of `\atcatcode` right to the beginning. 63
Now use `\ldf@finish` to wrap up 65
Now use `\LdfInit` to perform initial checks 63
esperant-1.4j
General: fixed typo in table caption (funtion instead of function) 55
esperanto-1.
`\dateesperanto`: Removed the capitals from `\today` 56
esperanto-1.0a
General: Renamed `babel.sty` in `babel.com` 55
esperanto-1.0b
General: Removed use of `\makeatletter` 55
esperanto-1.1
General: Added a warning when no hyphenation patterns were loaded. 55

magyar-1.3f
  `\captionsmagyar`: translated Proof and replaced some translations 138
magyar-1.3g
  General: Replaced `\undefined` with `\@undefined` and `\empty` with `\@empty` for consistency with LaTeX 138

norsk-1.0a
  General: Renamed `babel.sty` in `babel.com` 128
norsk-1.0c
  General: Removed use of `\@ifundefined` 128
norsk-1.1
  General: Added a warning when no hyphenation patterns were loaded. 128
    Brought up-to-date with babel 3.2a 128
  `\captionsnorsk`: `\headpagename` should be `\pagename` 128
    Added `\seename`, `\alsoname` and `\prefacename` 128
  `\captionsnynorsk`: `\headpagename` should be `\pagename` 129
    Added `\seename`, `\alsoname` and `\prefacename` 129
norsk-1.1.3
  General: Added a couple of translations (from Per Norman Oma, TeX@itk.unit.no) 128
norsk-1.2
  General: Update for LaTeX 2ε 128
norsk-1.2d
  General: Now use `\@nopatterns` to produce the warning 128
    Removed the use of `\filedate` and moved identification after the loading of `babel.def` 128
norsk-1.2f
  `\captionsnorsk`: Added `\proofname` for AMS-LaTeX 128
  `\norskhyphenmins`: Added setting of hyphenmin parameters 128
norsk-1.2g
  `\captionsnorsk`: Replaced 'Proof' by its translation 128
  `\captionsnynorsk`: Replaced 'Proof' by its translation 129
norsk-1.2h
  General: Moved the definition of `\atcatcode` right to the beginning. 128
    Now use `\ldf@finish` to wrap up 130
    Now use `\LdfInit` to perform initial checks 128
    Replaced `\undefined` with `\@undefined` and `\empty` with `\@empty` for consistency with LaTeX 128

polish-1.0a
  `\textpl`: Initially execute 'textpl 153
polish-1.1c
  General: Now use `\@nopatterns` to produce the warning 150
    Removed the use of `\filedate` and moved identification after the loading of babel.def 150
polish-1.1d
  General: The dqmacro for C used `\'c` 154
polish-1.2a
  General: Don't modify `\rm` and friends for LaTeX 2ε, take `\selectfont` instead 153
polish-1.2b
  `\captionspolish`: Added `\proofname` for AMS-LaTeX 150
polish-1.2d
  General: Now use `\ldf@finish` to wrap up 154
    Now use `\LdfInit` to perform initial checks 150