

# Babel, a multilingual package for use with L<sup>A</sup>T<sub>E</sub>X's standard document classes\*

Johannes Braams  
Kooiensewater 62  
2715 AJ Zoetermeer  
The Netherlands  
J.L.Braams@cistron.nl

Printed July 11, 1995

## Abstract

The standard distribution of L<sup>A</sup>T<sub>E</sub>X 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 L<sup>A</sup>T<sub>E</sub>X 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 T<sub>E</sub>X version 3 to provide an environment in which multilingual documents can be written.

<b>Contents</b>	<b>5</b>	<b>Compatibility with <code>german.sty</code></b>	<b>7</b>
<b>1 The user interface</b>	<b>2</b>	<b>6 Identification</b>	<b>7</b>
<b>2 Changes for L<sup>A</sup>T<sub>E</sub>X 2<math>\epsilon</math></b>	<b>3</b>	<b>7 The Package File</b>	<b>8</b>
<b>3 Changes in Babel version 3.5</b>	<b>4</b>	<b>8 The Kernel of Babel</b>	<b>10</b>
<b>4 The interface between the core of <code>babel</code> and the language definition files</b>	<b>4</b>	8.1 Multiple languages . . . . .	11
4.1 Support for active characters . . . . .	5	8.2 Support for active characters . . . . .	21
4.2 Support for saving macro definitions . . . . .	6	8.3 Support for active characters . . . . .	22
4.3 Support for extending macros . . . . .	6	8.4 Support for saving macro definitions . . . . .	27
4.4 Macros common to a number of languages . . .	6	8.5 Support for extending macros . . . . .	28
		8.6 Macros common to a number of languages . . .	28
		8.7 Making glyphs available . . .	29

---

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

8.8	Quotation marks . . . . .	29	23	The Spanish language	86
8.9	Letters . . . . .	30	24	The Catalan language	92
8.10	Shorthands for quotation marks . . . . .	32	25	The Galician language	99
8.11	Umlauts and trema's . . .	32	26	The Romanian language	104
8.12	The redefinition of the style commands . . . . .	33	27	The Danish language	107
	8.12.1 Redefinition of macros . . . . .	35	28	The Norwegian language	110
8.13	Cross referencing macros .	38	29	The Swedish language	113
9	Local Language Configuration	40	30	The Finnish language	116
10	Driver files for the documented source code	41	31	The Hungarian language	120
11	Conclusion	45	32	The Estonian language	123
12	Acknowledgements	45	32.1	Implementation . . . . .	123
13	The Esperanto language	46	33	The Croatian language	128
14	The Dutch language	50	34	The Czech language	131
15	The English language	54	35	The Polish language	134
16	The German language	57	36	The Slovak language	140
17	The Breton language	63	37	The Slovenian language	143
18	The Irish language	68	38	The Lower Sorbian language	146
19	The scottish language	71	39	The Upper Sorbian language	149
20	The French language	74	40	The Turkish language	153
21	The Italian language	79	41	The Bahasa language	157
22	The Portuguese language	82			

## 1 The user interface

The user interface of this package is quite simple. It basially consists of only two commands. These commands can be used to select another language or to find out what the current language is.

`language` The environment `language` 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.

`\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.

`\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.

`\language` The name of the current language is stored in the control sequence `\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.

`\usesshorthands` A command with one argument, the character to make active in order to define 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.

`\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.

## 2 Changes for L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>

With the advent of L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> the interface to `babel` in the preamble of the document has changed. With L<sup>A</sup>T<sub>E</sub>X 2.09 one used to call up the `babel` system with a line such as:

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

which would tell L<sup>A</sup>T<sub>E</sub>X that the document would be written in two languages, `dutch` and `english` and that `english` would be the first language in use.

The L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> 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 3.5

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:

- `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 probably 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`.

### 4 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 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 L<sup>A</sup>T<sub>E</sub>X as well as by plain TeX. This 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 `\langle lang \rangle hyphenmins`, `\captions⟨lang⟩`, `\date⟨lang⟩`, `\extras⟨lang⟩` and `\noextras⟨lang⟩`. 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 language definition 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.

<code>\langhyphenmin</code>	The macro <code>\langle lang \rangle hyphenmins</code> is used to store the values of the <code>\lefthyphenmin</code> and <code>\righthyphenmin</code> .
<code>\captionslang</code>	The macro <code>\captions⟨lang⟩</code> defines the macros that hold the texts to replace the original hard-wired texts.
<code>\datelang</code>	The macro <code>\date⟨lang⟩</code> defines <code>\today</code> and
<code>\extraslang</code>	The macro <code>\extras⟨lang⟩</code> contains all the extra definitions needed for a specific language.
<code>\noextraslang</code>	Because we want to offer the user the possibility to switch between languages and we do not know in what state <code>T<sub>E</sub>X</code> might be after the execution of <code>\extras⟨lang⟩</code> , a macro that brings <code>T<sub>E</sub>X</code> into a predefined state is needed. It will be no surprise that the name of this macro is <code>\noextras⟨lang⟩</code> .
<code>\main@language</code>	To postpone the activation of the definitions needed for a language until the beginning of a document, all language definition files should use <code>\main@language</code> instead of <code>\selectlanguage</code> . This will just store the name of the language and the proper language will be activated at the start of the document.
<code>\loadlocalcfg</code>	At the end of the processing of a language definition file <code>L<sup>A</sup>T<sub>E</sub>X</code> can be instructed to load a local configuration file. This file can for instance be used to add strings to <code>\captions⟨lang⟩</code> in order to support local document classes. The user will be informed of the fact that this configuration file is loaded.

## 4.1 Support for active characters

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

<code>\initiate@active@char</code>	The internal macro <code>\initiate@active@char</code> is used in language definition files to instruct <code>L<sup>A</sup>T<sub>E</sub>X</code> to give a character the category code ‘active’. When a character has been made active it will remain that way until the end of the document. Its definition may vary.
<code>\bbl@activate</code> <code>\bbl@deactivate</code>	The command <code>\bbl@activate</code> is used to change the way an active character expands. <code>\bbl@activate</code> ‘switches on’ the active behaviour of the character. <code>\bbl@deactivate</code> lets the active character expand to its former (mostly) non-active self.
<code>\declare@shorthand</code>	The macro <code>\declare@shorthand</code> 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`

“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`. L<sup>A</sup>TeX 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.

## 4.2 Support for saving macro definitions

Language definition files may want to *redefine* 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<sup>1</sup>.

`\babel@save`  
`\babel@savevariable`

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

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.

## 4.3 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`.

## 4.4 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.

---

<sup>1</sup>This mechanism was introduced by Bernd Raichle.

## 5 Compatibility with `german.sty`

As has been discussed before, 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`<sup>2</sup>.

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

## 6 Identification

The file `babel.sty`<sup>3</sup> is meant for  $\text{\LaTeX 2}_\epsilon$ , therefore we make sure that the format file used is the right one.

```
6.1 <+package>\NeedsTeXFormat{LaTeX2e}
```

The identification code for each file is something that was introduced in  $\text{\LaTeX 2}_\epsilon$ . When the command `\ProvidesFile` does not exist, a dummy definition is provided.

```
6.2 <!*package>
```

```
6.3 \ifx\ProvidesFile\undefined
```

```
6.4   \def\ProvidesFile#1[#2 #3 #4]{\wlog{#4 #3 <#2>}}\fi
```

```
6.5 </!package>
```

Identify each file that is produced from this source file.

```
6.6 <+package>\ProvidesPackage{babel}
```

```
6.7 <+core>\ProvidesFile{babel.def}
```

```
6.8 <+kernel & patterns>\ProvidesFile{hyphen.cfg}
```

```
6.9 <+kernel &!patterns>\ProvidesFile{switch.def}
```

```
6.10 <+driver &!user>\ProvidesFile{babel.drv}
```

```
6.11 <+driver & user>\ProvidesFile{user.drv}
```

```
6.12                                     [1995/07/11 v3.5e
```

```
6.13 <+package>           The Babel package]
```

```
6.14 <+core>              Babel common definitions]
```

```
6.15 <+kernel>            Babel language switching mechanism]
```

```
6.16 <+driver>]
```

---

<sup>2</sup>The ‘b’ is added to the name to distinguish the file from Partls’ file.

<sup>3</sup>The file described in this section is called `babel.dtx`, has version number ? and was last revised on ?.

## 7 The Package File

In order to make use of the new features of L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>, a new file is introduced to the babel system, `babel.sty`. This file is loaded by the `\usepackage` command and defines all the language options known in the babelsystem.

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

‘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 or the patterns to be loaded as ‘english’ ‘american’ or ‘USenglish’.

```
7.1 (*package)
7.2 \DeclareOption{american}{%
7.3   \ifx\l@USenglish\undefined
7.4   \else
7.5     \let\l@american\l@USenglish
7.6   \fi
7.7   \input{english.ldf}%
7.8   \main@language{american}}

Austrian is really a dialect of German.

7.9 \DeclareOption{austrian}{%
7.10  \input{germanb.ldf}%
7.11  \main@language{austrian}}
7.12 \DeclareOption{bahasa}{\input{bahasa.ldf}}
7.13 \DeclareOption{brazil}{\input{portuges.ldf}\main@language{brazil}}

7.14 \DeclareOption{brazilian}{%
7.15  \input{portuges.ldf}%
7.16  \let\captionsbrazilian\captionsbrazil
7.17  \let\datebrazilian\datebrazil
7.18  \let\extrasbrazilian\extrasbrazil
7.19  \let\noextrasbrazilian\noextrasbrazil
7.20  \let\brazilianhyphenmins\brazilhyphenmins
7.21  \main@language{brazilian}}
7.22 \DeclareOption{breton}{\input{breton.ldf}}

7.23 \DeclareOption{british}{%
7.24  \ifx\l@british\undefined
7.25    \ifx\l@UKenglish\undefined
7.26    \else
7.27      \let\l@english\l@UKenglish
7.28    \fi
7.29  \else
7.30    \let\l@english\l@british
7.31  \fi
7.32  \input{english.ldf}%
7.33  \ifx\l@british\undefined
7.34    \let\l@british\l@english
7.35  \fi
7.36  \let\captionsbritish\captionsenglish
7.37  \let\datebritish\dateenglish
7.38  \let\extrasbritish\extrasenglish
7.39  \let\britishhyphenmins\englishhyphenmins
7.40  \main@language{british}
7.41 }
```

```

7.42 \DeclareOption{catalan}{\input{catalan.ldf}}
7.43 \DeclareOption{croatian}{\input{croatian.ldf}}
7.44 \DeclareOption{czech}{\input{czech.ldf}}
7.45 \DeclareOption{danish}{\input{danish.ldf}}
7.46 \DeclareOption{dutch}{\input{dutch.ldf}}

```

We allow for the british english patterns to be loaded as either ‘english’ or ‘UKenglish’

```

7.47 \DeclareOption{english}{%
7.48   \ifx\l@UKenglish\undefined
7.49   \else
7.50     \let\l@english\l@UKenglish
7.51   \fi
7.52   \input{english.ldf}%
7.53 }
7.54 \DeclareOption{esperanto}{\input{esperant.ldf}}

```

```

7.55 \DeclareOption{estonian}{\input{estonian.ldf}}
7.56 \DeclareOption{finnish}{\input{finnish.ldf}}

```

The babel support of French is stored in `francais.ldf`; therefore the  $\text{\LaTeX}2.09$  option used to be `francais`. The hyphenation patterns may be loaded as either ‘french’ or as ‘francais’.

```

7.57 \DeclareOption{francais}{%
7.58   \ifx\l@francais\undefined
7.59   \let\l@francais\l@french
7.60   \fi
7.61   \input{francais.ldf}%
7.62 }

```

With  $\text{\LaTeX}2_{\epsilon}$  we can now also use the option `french` and still call the file `francais.ldf`.

```

7.63 \DeclareOption{french}{%
7.64   \ifx\l@french\undefined
7.65   \let\l@french\l@francais
7.66   \fi
7.67   \IfFileExists{french.ldf}{%
7.68     \input{french.ldf}}{%
7.69     \input{francais.ldf}}%
7.70   \let\captionfrench\captionfrancais
7.71   \let\datefrench\datefrancais
7.72   \let\extrafrench\extrasfrancais
7.73   \let\noextrafrench\noextrasfrancais
7.74   \let\frenchhyphenmins\francaishyphenmins
7.75 }
7.76 \DeclareOption{galician}{\input{galician.ldf}}
7.77 \DeclareOption{german}{\input{germanb.ldf}}
7.78 \DeclareOption{germanb}{\input{germanb.ldf}}

```

hungarian is just a synonym for magyar

```

7.79 \DeclareOption{hungarian}{\input{magyar.ldf}%
7.80   \let\captionhungarian\captionmagyar
7.81   \let\datehungarian\datemagyar
7.82   \let\extrahungarian\extrasmagyar
7.83   \let\noextrahungarian\noextrasmagyar

```

```

7.84 \let\hugarianhyphenmins\magyarhyphenmins
7.85 }
7.86 \DeclareOption{irish}{\input{irish.ldf}}
7.87 \DeclareOption{italian}{\input{italian.ldf}}
7.88 \DeclareOption{lowersorbian}{\input{lsorbian.ldf}}
7.89 \DeclareOption{magyar}{\input{magyar.ldf}}
7.90 \DeclareOption{norsk}{\input{norsk.ldf}}

```

For Norwegian two spelling variants are provided.

```

7.91 \DeclareOption{nynorsk}{%
7.92   \input{norsk.ldf}%
7.93   \main@language{nynorsk}}
7.94 \DeclareOption{polish}{\input{polish.ldf}}
7.95 \DeclareOption{portuges}{\input{portuges.ldf}}
7.96 \DeclareOption{portuguese}{\input{portuges.ldf}%
7.97   \let\captionportuguese\captionportuges
7.98   \let\dateportuguese\dateportuges
7.99   \let\extraportuguese\extraportuges
7.100  \let\noextraportuguese\noextraportuges
7.101  \let\portuguesehyphenmins\portugeshyphenmins
7.102 }
7.103 \DeclareOption{romanian}{\input{romanian.ldf}}
7.104 %\DeclareOption{russian}{\input{russian.ldf}}
7.105 \DeclareOption{scottish}{\input{scottish.ldf}}
7.106 \DeclareOption{spanish}{\input{spanish.ldf}}
7.107 \DeclareOption{slovak}{\input{slovak.ldf}}
7.108 \DeclareOption{slovene}{\input{slovene.ldf}}
7.109 \DeclareOption{swedish}{\input{swedish.ldf}}
7.110 \DeclareOption{turkish}{\input{turkish.ldf}}
7.111 \DeclareOption{uppersorbian}{\input{usorbian.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.

```

7.112 \DeclareOption{activeacute}{}
7.113 \DeclareOption{activegrave}{}

```

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

```

7.114 \ProcessOptions*
7.115 \endpackage

```

## 8 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 T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X, some of it is for the L<sup>A</sup>T<sub>E</sub>X 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.

```
8.1 \ifx\AtBeginDocument\undefined
8.2 \input plain.def\relax
8.3 \fi
```

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

```
8.4 (*core)
8.5 \ifx\undefined\iflanguage
8.6 \input switch.def\relax
8.7 \fi
```

To communicate to the language definition files that the core of the `babel` system has been loaded, the following control sequence is just `\let` equal to `\relax`.

```
8.8 \let\babel@core@loaded\relax
8.9 </core>
```

## 8.1 Multiple languages

With T<sub>E</sub>X 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 T<sub>E</sub>X 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 T<sub>E</sub>X environment has to record for which languages he has hyphenation patterns *and* in which files these are stored<sup>4</sup>. 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 L<sup>A</sup>T<sub>E</sub>X 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
```

---

<sup>4</sup>This is because different operating systems sometimes use *very* different filenames conventions.

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.

```
8.10 (*kernel)
8.11 (*!patterns)
8.12 \expandafter\ifx\csname iflanguage\endcsname\relax \else
8.13 \expandafter\endinput
8.14 \fi
8.15 \!/patterns)
```

`\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.

```
8.16 \ifx\language\undefined
8.17 \csname newcount\endcsname\language
8.18 \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,

```
8.19 \ifx\newlanguage\undefined
8.20 \csname newcount\endcsname\last@language
    plain TeX version 3.0 uses \count 19 for this purpose.
8.21 \else
8.22 \countdef\last@language=19
8.23 \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.

```
8.24 \ifx\newlanguage\undefined
8.25 \def\addlanguage#1{%
8.26     \global\advance\last@language \@ne
8.27     \ifnum\last@language<\@cclvi
8.28     \else
8.29         \errmessage{No room for a new \string\language!}%
8.30     \fi
8.31     \global\chardef#1\last@language
8.32     \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`.

```
8.33 \else
8.34 \def\addlanguage{\alloc@9\language\chardef\@cclvi}
8.35 \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.

```
8.36 \def\adddialect#1#2{%
8.37   \global\chardef#1#2\relax
8.38   \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.

```
8.39 \def\iflanguage#1#2#3{%
8.40   \expandafter\ifx\csname l@#1\endcsname\relax
8.41     \@nolanerr{#1}%
8.42   \else
8.43     \ifnum\csname l@#1\endcsname=\language #2%
8.44     \else#3\fi
8.45   \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.

```
8.46 \def\selectlanguage#1{%
8.47   \edef\languagename{%
8.48     \ifnum\escapechar=\expandafter'\string#1\empty
8.49     \else \string#1\empty\fi}
8.50   \expandafter\protect\csname selectlanguage \expandafter\endcsname
8.51   \expandafter{\languagename}}
```

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

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

Remark: If the `\selectlanguage` command is written to a file, a possible control sequence argument gets totally expanded to the string without the leading escape character. In the normal case we have to deal with the fact that the argument of `\selectlanguage` is totally unexpanded at the moment. There is only one disadvantage in the current implementation: `\originalTeX` contains this unexpanded argument and therefore needs more memory for its macro definition.

```
8.53 \expandafter\def\csname selectlanguage \endcsname#1{%
8.54   \select@language{#1}%
```

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

```
8.55 \if@filesw
8.56   \protected@write\auxout{}\string\select@language{#1}%
8.57   \addtocontents{toc}{\string\select@language{#1}}%
8.58   \addtocontents{lof}{\string\select@language{#1}}%
8.59   \addtocontents{lot}{\string\select@language{#1}}%
8.60 \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.

```
8.61 \def\select@language#1{%
8.62   \expandafter\ifx\csname l@#1\endcsname\relax
8.63     \@nolanerr{#1}%
8.64   \else
8.65     \language=\csname l@#1\endcsname\relax
8.66     \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 *redefine* `\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.

```
8.67   \expandafter\def\expandafter\originalTeX
8.68     \expandafter{\csname noextras#1\endcsname
8.69               \let\originalTeX\empty}%
8.70   \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.

```
8.71   \csname captions#1\endcsname
8.72   \csname date#1\endcsname
8.73   \csname extras#1\endcsname\relax
```

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

```

8.74 \babel@savevariable\lefthyphenmin
8.75 \babel@savevariable\righthyphenmin
8.76 \expandafter\ifx\csname #1hyphenmins\endcsname\relax
8.77 \lefthyphenmin\tw@\righthyphenmin\thr@@
8.78 \else
8.79 \expandafter\expandafter\expandafter\set@hyphenmins
8.80 \csname #1hyphenmins\endcsname
8.81 \fi
8.82 \fi}

```

`otherlanguage` The `otherlanguage` environment can be used as an alternative to using the `\selectlanguage` declarative command. When you are typesetting a document with 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 `\language`; 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.

```

8.83 \long\def\otherlanguage#1{%
8.84 \def\language{#1}%
8.85 \csname selectlanguage \endcsname{#1}%
8.86 \ignorespaces
8.87 }

```

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.

```

8.88 \long\def\endotherlanguage{%
8.89 \originalTeX
8.90 \global\@ignoretrue\ignorespaces
8.91 }

```

`\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\langle lang \rangle` command doesn't make any `\global` changes. The coding is very similar to part of `\selectlanguage`.

```

8.92 \def\foreignlanguage#1#2{%
8.93 \begingroup
8.94 \def\language{#1}%
8.95 \expandafter\ifx\csname l@#1\endcsname\relax
8.96 \@nolanerr{#1}%
8.97 \else
8.98 \language=\csname l@#1\endcsname\relax
8.99 \csname extras#1\endcsname

8.100 \expandafter\ifx\csname #1hyphenmins\endcsname\relax
8.101 \lefthyphenmin\tw@\righthyphenmin\thr@@

```

```

8.102     \else
8.103         \expandafter\expandafter\expandafter\set@hyphenmins
8.104             \csname #1hyphenmins\endcsname
8.105     \fi
8.106     \fi
8.107     #2
8.108     \csname noextras#1\endcsname
8.109     \endgroup
8.110 }

```

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

```

8.111 \def\set@hyphenmins#1#2{\lefthyphenmin#1\righthyphenmin#2}
8.112 </kernel>

```

`\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.

```

8.113 (*core)
8.114 \def\main@language#1{\def\bbl@main@language{#1}}

```

The default is to use English as the main language.

```

8.115 \main@language{english}

```

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

```

8.116 \AtBeginDocument{%
8.117     \expandafter\selectlanguage\expandafter{\bbl@main@language}}
8.118 </core>

```

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`.

```

8.119 (*kernel)
8.120 \ifx\undefined\originalTeX\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.

```

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

```

`\@nolanerr` 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  $\LaTeX_{2\epsilon}$ , so we can safely use its error handling interface. Otherwise we'll have to 'keep it simple'.

```

8.122 \ifx\PackageError\undefined
8.123     \def\@nolanerr#1{%
8.124         \errhelp{Your command will be ignored, type <return> to proceed}%
8.125         \errmessage{You haven't defined the language #1\space yet}}
8.126     \def\@nopatterns#1{%
8.127         \message{No hyphenation patterns were loaded for}
8.128         \message{the language '#1'}}

```

```

8.129 \message{I will use the patterns loaded for \string\language=0
8.130         instead}}
8.131 \def\@activated#1{%
8.132   \wlog{Package babel Info: Making #1 an active character}}
8.133 \else
8.134 \newcommand*\@nolanerr}[1]{%
8.135   \PackageError{babel}%
8.136     {You haven't defined the language #1\space yet}%
8.137     {Your command will be ignored, type <return> to proceed}}
8.138 \newcommand*\@nopatterns}[1]{%
8.139   \PackageWarningNoLine{babel}%
8.140     {No hyphenation patterns were loaded for\MessageBreak
8.141     the language '#1'\MessageBreak
8.142     I will use the patterns loaded for \string\language=0
8.143     instead}}
8.144 \newcommand*\@activated}[1]{%
8.145   \PackageInfo{babel}{%
8.146     Making #1 an active character}}
8.147 \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`.

```
8.148 (*patterns)
```

`\patterns@loaded` It has been suggested to add a remark to `LaTeX`'s `\everyjob` message, stating which hyphenation patterns have been loaded. This can be done by first collecting (in a token register) the names when processing the file `language.dat` and afterwards adding a string to the `\everyjob` message. The token register is initially empty.

```
8.149 \newtoks\patterns@loaded \global\patterns@loaded={}
```

`\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 whether the line starts with `=`.

`\bbl@eq@` To be able to do that we need an `=`, stored in a macro.

```
8.150 \def\bbl@eq@{=}
```

When the first token of a line is an `=`, the macro `\process@synonym` is called; otherwise the macro `\process@language` will continue.

```

8.151 \def\process@line#1#2/{%
8.152   \def\bbl@tmp{#1}
8.153   \ifx\bbl@tmp\bbl@eq@
8.154     \process@synonym#2/
8.155   \else
8.156     \process@language#1#2/%
8.157   \fi
8.158 }

```

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

```

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

```

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

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

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

```
8.165 \expandafter\global
8.166 \expandafter\chardef\csname l@#1\endcsname\last@language
8.167 \wlog{\string\l@#1=\string\language\the\last@language}
8.168 \fi
8.169 }
```

`\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, therefore 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’.

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

Then the ‘name’ of the language that will be loaded now is added to the token register `\patterns@loaded`. and finally the pattern file is read.

```
8.173 \global\patterns@loaded\expandafter{\the\patterns@loaded#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 `\langhyphenmins` macro. When no assignments were made we provide a default setting.

```
8.174 \lefthyphenmin\m@ne
8.175 \input #2\relax
8.176 \ifnum\lefthyphenmin=\m@ne
8.177 \lefthyphenmin\tw@
8.178 \righthyphenmin\thr@@
8.179 \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.

```
8.180 \def\bbl@tmp{#3}
8.181 \ifx\bbl@tmp@empty
8.182 \else
8.183 \ifx\bbl@tmp\space
8.184 \else
8.185 \input #3\relax
8.186 \fi
8.187 \fi
```

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

```

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

```

The configuration file can now be opened for reading.

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

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

```

8.191 \ifeof1
8.192 \message{I couldn't find the file language.dat, \space
8.193         I will try the file hyphen.tex}
8.194 \input hyphen.tex\relax
8.195 \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`.

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

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

```
8.197 \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.

```

8.198 \endlinechar@m@ne
8.199 \read1 to \bbl@line
8.200 \endlinechar'\^M

```

Empty lines are skipped.

```

8.201 \ifx\bbl@line\empty
8.202 \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.

```

8.203 \edef\bbl@line{\bbl@line\space/}
8.204 \expandafter\process@line\bbl@line
8.205 \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`.

```

8.206 \iftrue \csname fi\endcsname
8.207 \csname if\ifeof1 false\else true\fi\endcsname
8.208 \repeat

```

Reactivate the default patterns,

```

8.209 \language=0
8.210 \fi

```

and close the configuration file.

```
8.211 \closein1
```

Also remove some macros from memory

```
8.212 \let\process@language\undefined
8.213 \let\process@synonym\undefined
8.214 \let\process@line\undefined
8.215 \let\bbl@tmp\undefined
8.216 \let\bbl@eq@\undefined
8.217 \let\bbl@line\undefined
```

We want to add a message to the message L<sup>A</sup>T<sub>E</sub>X 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 L<sup>A</sup>T<sub>E</sub>X fills the register. There are some problems with this approach though.

- When someone wants to use several hyphenation patterns with S<sub>L</sub>T<sub>E</sub>X the above scheme won't work. The reason is that S<sub>L</sub>T<sub>E</sub>X overwrites the contents of the `\everyjob` register with its own message.
- Plain T<sub>E</sub>X 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 original `\dump` is saved in `\orig@dump` and a new definition is supplied.

```
8.218 \let\orig@dump=\dump
8.219 \def\dump{%
```

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.

```
8.220 \everyjob\expandafter{\the\everyjob%
8.221 \immediate\write16{Hyphenation patterns for \the\patterns@loaded
8.222 loaded.}}%
```

Then everything is restored to the old situation and the format is dumped.

```
8.223 \let\dump\orig@dump\let\orig@dump\undefined\dump}
```

Here the code for iniT<sub>E</sub>X ends.

```
8.224 </patterns>
8.225 </kernel>
```

## 8.2 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 L<sup>A</sup>T<sub>E</sub>X 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.

```
8.226 (*core | shorthands)
8.227 \def\bbl@add@special#1{\begingroup
8.228   \def\do{\noexpand\do\noexpand}%
8.229   \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.

```
8.230   \edef\x{\endgroup
8.231     \def\noexpand\dospecials{\dospecials\do#1}%
8.232     \expandafter\ifx\csname @sanitize\endcsname\relax \else
8.233       \def\noexpand\@sanitize{\@sanitize\@makeother#1}%
8.234     \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.

```
8.235 \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 T<sub>E</sub>X 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.

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

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

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

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

```
8.241   \edef\x{\endgroup
8.242     \def\noexpand\dospecials{\dospecials}%
8.243     \expandafter\ifx\csname @sanitize\endcsname\relax \else
8.244       \def\noexpand\@sanitize{\@sanitize}%
8.245     \fi}%
8.246 \x}
```

### 8.3 Support for active characters

`\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 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` Because the code that is used in the handling of active characters may need to  
`\bbl@afterfi` look ahead, we take extra care to ‘throw’ it over the `\else` and `\fi` parts of an `\if`-statement<sup>5</sup>.

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

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.

```
8.249 \begingroup
8.250 \catcode'\~\active
8.251 \def\x{\endgroup
8.252 \def\initiate@active@char##1{%
```

If the character is already active we don’t do anything.

```
8.253 \ifcat\noexpand##1\noexpand~\relax
8.254 \else
```

Otherwise we write a message in the transcript file,

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

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

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

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

```
8.257 \ifx##1' %
8.258 \let\pr@m@s\bbl@pr@m@s
8.259 \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’.

```
8.260 \lccode'\~='##1%
8.261 \lowercase{%
```

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

```
8.262 \catcode'\~\active
8.263 \expandafter\bbl@add@special
8.264 \csname \string##1\endcsname
```

---

<sup>5</sup>This code is based on code presented in TUGboat vol. 12, no2, June 1991 in “An expansion Power Lemma” by Sonja Maus.

Define the character to expand to

```
\active@prefix <char> \active@char<char>
```

(where `\active@char<char>` is *one* control sequence!).

```
8.265 \expandafter\gdef
8.266 \expandafter~%
8.267 \expandafter{%
8.268 \expandafter\active@prefix\expandafter##1%
8.269 \csname normal@char\string##1\endcsname}}%
```

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>`.

```
8.270 \@namedef{active@char\string##1}{%
8.271 \if@safe@actives
8.272 \bbl@afterelse\csname normal@char\string##1\endcsname
8.273 \else
8.274 \bbl@afterfi\csname user@active\string##1\endcsname
8.275 \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.

```
8.276 \@namedef{user@active\string##1}{%
8.277 \expandafter\ifx
8.278 \csname \user@group @sh@\string##1\endcsname
8.279 \relax
8.280 \bbl@afterelse\csname @sh@\string##1@sel\endcsname
8.281 {user@active@arg\string##1}{language@active\string##1}%
8.282 \else
8.283 \bbl@afterfi\csname \user@group @sh@\string##1\endcsname
8.284 \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.

```
8.285 \long\@namedef{user@active@arg\string##1}###1{%
8.286 \expandafter\ifx
8.287 \csname \user@group @sh@\string##1\string###1\endcsname
8.288 \relax
8.289 \bbl@afterelse
8.290 \csname language@active\string##1\endcsname###1%
8.291 \else
8.292 \bbl@afterfi
8.293 \csname \user@group @sh@\string##1\string###1%
8.294 \endcsname
8.295 \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.

```
8.296 \@namedef{language@active\string##1}{%
8.297 \expandafter\ifx
8.298 \csname \language@group @sh@\string##1\endcsname
8.299 \relax
```

```

8.300         \bbl@afterelse\csname @sh@\string##1@sel\endcsname
8.301             {language@active@arg\string##1}{system@active\string##1}%
8.302     \else
8.303         \bbl@afterfi
8.304         \csname \language@group @sh@\string##1@\endcsname
8.305     \fi}%
8.306     \long\@namedef{language@active@arg\string##1}####1{%
8.307     \expandafter\ifx
8.308     \csname \language@group @sh@\string##1\string####1@\endcsname
8.309     \relax
8.310         \bbl@afterelse
8.311         \csname system@active\string##1\endcsname####1%
8.312     \else
8.313         \bbl@afterfi
8.314         \csname \language@group @sh@\string##1\string####1@%
8.315         \endcsname
8.316     \fi}%

```

And the same goes for the system level.

```

8.317     \@namedef{system@active\string##1}{%
8.318     \expandafter\ifx
8.319     \csname \system@group @sh@\string##1@\endcsname
8.320     \relax
8.321         \bbl@afterelse\csname @sh@\string##1@sel\endcsname
8.322             {system@active@arg\string##1}{normal@char\string##1}%
8.323     \else
8.324         \bbl@afterfi\csname \system@group @sh@\string##1@\endcsname
8.325     \fi}%

```

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

```

8.326     \long\@namedef{system@active@arg\string##1}####1{%
8.327     \expandafter\ifx
8.328     \csname \system@group @sh@\string##1\string####1@\endcsname
8.329     \relax
8.330         \bbl@afterelse\csname normal@char\string##1\endcsname####1%
8.331     \else
8.332         \bbl@afterfi
8.333         \csname \system@group @sh@\string##1\string####1@\endcsname
8.334     \fi}%
8.335     \fi}%
8.336     }\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 `\protects` the active character whenever `\protect` is *not* `\@typeset@protect`.

```

8.337 \def\active@prefix#1{%
8.338   \ifx\protect\@typeset@protect
8.339   \else
8.340     \bbl@afterfi\protect#1\@gobble
8.341   \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 avail-

able. This setting of this switch should be checked in the first level expansion of `\active@char⟨char⟩`.

```
8.342 \newif\if@safe@actives
8.343 \@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⟩`.

```
8.344 \def\bbl@activate#1{%
8.345   \expandafter\def
8.346   \expandafter#1\expandafter{%
8.347     \expandafter\active@prefix
8.348     \expandafter#1\csname active@char\string#1\endcsname}%
8.349 }
```

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

```
8.350 \def\bbl@deactivate#1{%
8.351   \expandafter\def
8.352   \expandafter#1\expandafter{%
8.353     \expandafter\active@prefix
8.354     \expandafter#1\csname normal@char\string#1\endcsname}%
8.355 }
```

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

```
8.356 \def\bbl@firstcs#1#2{\csname#1\endcsname}
8.357 \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.

```
8.358 \def\declare@shorthand#1#2{\@decl@short{#1}#2@nil}
8.359 \def\@decl@short#1#2#3@nil#4{%
8.360   \def\tmp{#3}%
8.361   \ifx\tmp\empty
8.362     \expandafter\let\csname @sh@\string#2@sel\endcsname\bbl@scndcs
8.363   \else
8.364     \expandafter\let\csname @sh@\string#2@sel\endcsname\bbl@firstcs
8.365   \fi
8.366   \@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.

```
8.367 \def\textormath#1#2{%
8.368   \ifmmode
```

```

8.369 \bbl@afterelse#2%
8.370 \else
8.371 \bbl@afterfi#1%
8.372 \fi}

```

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

```

8.373 \def\user@group{}
8.374 \def\language@group{english}
8.375 \def\system@group{system}

```

`\usesshorthands` This is the user level command to tell L<sup>A</sup>T<sub>E</sub>X that user level shorthands will be used  
in the document. It takes one argument, the character that starts a shorthand.

```

8.376 \def\usesshorthands#1{%
8.377 \def\user@group{user}%
8.378 \initiate@active@char{#1}%
8.379 \bbl@activate{#1}}

```

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

```

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

```

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

```

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

```

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

```

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

```

When the right quote is made active we need to take care of handling it cor-  
rectly 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 defini-  
tion in mathmode. (Note that the right quote is ‘active’ in mathmode because of  
its mathcode.)

```

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

```

`\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.

```

8.386 \begingroup
8.387 \catcode'\'\active\let'\relax
8.388 \def\x{\endgroup
8.389 \def\bbl@pr@m@s{%
8.390 \ifx'\@let@token
8.391 \expandafter\pr@@@s
8.392 \else
8.393 \ifx^\@let@token
8.394 \expandafter\expandafter\expandafter\pr@@@t
8.395 \else

```

```

8.396         \egroup
8.397         \fi
8.398     \fi}%
8.399     }
8.400 \x
8.401 </core | shorthands>

```

Normally the `~` is active and expands to `\penalty\@M\_\_`. When it is written to the `.aux` file 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. To achieve that the character `~` first needs to have character code 12 (other).

```

8.402 (*core)
8.403 \catcode'~12\relax
8.404 \initiate@active@char{~}
8.405 \declare@shorthand{system}{~}{\penalty\@M\_\_ }
8.406 \bbl@activate{~}

```

`\OT1dqpos` 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.

```

8.407 \expandafter\def\csname OT1dqpos\endcsname{127}
8.408 \expandafter\def\csname T1dqpos\endcsname{4}

```

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

```

8.409 \ifx\f@encoding\undefined
8.410   \def\f@encoding{OT1}
8.411 \fi

```

## 8.4 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@savcnt` The initialization of a new save cycle: reset the counter to zero.  
`\babel@beginsave`<sub>8.412</sub> `\def\babel@beginsave{\babel@savcnt\z@}`

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

```

8.413 \newcount\babel@savcnt
8.414 \babel@beginsave

```

`\babel@save` The macro `\babel@save<cname>` saves the current meaning of the control sequence `<cname>` to `\originalTeX`<sup>6</sup>. 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.

---

<sup>6</sup>`\originalTeX` has to be expandable, i.e. you shouldn't let it to `\relax`.

```

8.415 \def\babel@save#1{%
8.416   \expandafter\let\csname babel@\number\babel@savecnt\endcsname #1\relax
8.417   \begingroup
8.418     \toks@\expandafter{\originalTeX \let#1=}
8.419     \edef\x{\endgroup
8.420       \def\noexpand\originalTeX{\the\toks@ \expandafter\noexpand
8.421         \csname babel@\number\babel@savecnt\endcsname\relax}}
8.422     \x
8.423   \advance\babel@savecnt\@ne}

```

`\babel@savevariable` The macro `\babel@savevariable` $\langle variable \rangle$  saves the value of the variable.  $\langle variable \rangle$  can be anything allowed after the `\the` primitive.

```

8.424 \def\babel@savevariable#1{\begingroup
8.425   \toks@\expandafter{\originalTeX #1=}
8.426   \edef\x{\endgroup
8.427     \def\noexpand\originalTeX{\the\toks@ \the#1\relax}}
8.428   \x}

```

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

```

8.429 \def\bbl@frenchspacing{%
8.430   \ifnum\the\sfcodes'\.=\@m
8.431     \let\bbl@nonfrenchspacing\relax
8.432   \else
8.433     \frenchspacing
8.434     \let\bbl@nonfrenchspacing\nonfrenchspacing
8.435   \fi}
8.436 \let\bbl@nonfrenchspacing\nonfrenchspacing

```

## 8.5 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  $\langle control sequence \rangle$  and T<sub>E</sub>X-code to be added to the  $\langle control sequence \rangle$ .

If the  $\langle control sequence \rangle$  has not been defined before it is defined now.

```

8.437 \def\addto#1#2{%
8.438   \ifx#1\undefined
8.439     \def#1{#2}
8.440   \else

```

Otherwise the replacement text for the  $\langle control sequence \rangle$  is expanded and stored in a token register, together with the T<sub>E</sub>X-code to be added. Finally the  $\langle control sequence \rangle$  is redefined, using the contents of the token register.

```

8.441   {\toks@\expandafter{#1#2}}
8.442   \xdef#1{\the\toks@}
8.443   \fi
8.444 }

```

## 8.6 Macros common to a number of languages

`\allowhyphens` This macro makes hyphenation possible. Basically its definition is nothing more than `\nobreak \hskip Opt plus Opt`<sup>7</sup>.

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

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

```
8.446 \def\setlow@box#1{\setbox\tw@\hbox{,}\setbox\z@\hbox{#1}%
```

```
8.447 \dimen\z@\ht\z@ \advance\dimen\z@ -\ht\tw@%
```

```
8.448 \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.

```
8.449 \def\save@sf@q#1{{\ifhmode
```

```
8.450 \edef\@SF{\spacefactor\the\spacefactor}\else
```

```
8.451 \let\@SF\empty \fi \leavevmode #1\@SF}}
```

## 8.7 Making glyphs available

The file `babel.dtx`<sup>8</sup> 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`.

## 8.8 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.

```
8.452 \ProvideTextCommand{\quotedblbase}{OT1}{%
```

```
8.453 \save@sf@q{\setlow@box{\textquotedblright\}}%
```

```
8.454 \box\z@\kern-.04em\allowhyphens}}
```

Make sure that when an encoding other than OT1 or T1 is used this glyph can still be typeset.

```
8.455 \ProvideTextCommandDefault{\quotedblbase}{%
```

```
8.456 \UseTextSymbol{OT1}{\quotedblbase}}
```

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

```
8.457 \ProvideTextCommand{\quotesinglbase}{OT1}{%
```

```
8.458 \save@sf@q{\setlow@box{\textquoteright\}}%
```

```
8.459 \box\z@\kern-.04em\allowhyphens}}
```

Make sure that when an encoding other than OT1 or T1 is used this glyph can still be typeset.

```
8.460 \ProvideTextCommandDefault{\quotesinglbase}{%
```

```
8.461 \UseTextSymbol{OT1}{\quotesinglbase}}
```

---

<sup>7</sup>TeX begins and ends a word for hyphenation at a glue node. The penalty prevents a linebreak at this glue node.

<sup>8</sup>The file described in this section has version number ?, and was last revised on ?.

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

```
\guillemotright 8.462 \ProvideTextCommand{\guillemotleft}{OT1}{%
  8.463 \ifmmode
  8.464 \ll
  8.465 \else
  8.466 \save@sf@q{\penalty\M
  8.467 \raise.2ex\hbox{\scriptscriptstyle\ll}\allowhyphens}%
  8.468 \fi}
8.469 \ProvideTextCommand{\guillemotright}{OT1}{%
  8.470 \ifmmode
  8.471 \gg
  8.472 \else
  8.473 \save@sf@q{\penalty\M
  8.474 \raise.2ex\hbox{\scriptscriptstyle\gg}\allowhyphens}%
  8.475 \fi}
```

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

```
8.476 \ProvideTextCommandDefault{\guillemotleft}{%
8.477 \UseTextSymbol{OT1}{\guillemotleft}}
8.478 \ProvideTextCommandDefault{\guillemotright}{%
8.479 \UseTextSymbol{OT1}{\guillemotright}}
```

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

```
\guilsinglright 8.480 \ProvideTextCommand{\guilsinglleft}{OT1}{%
  8.481 \ifmmode
  8.482 <%
  8.483 \else
  8.484 \save@sf@q{\penalty\M
  8.485 \raise.2ex\hbox{\scriptscriptstyle<}\allowhyphens}%
  8.486 \fi}
8.487 \ProvideTextCommand{\guilsinglright}{OT1}{%
  8.488 \ifmmode
  8.489 >%
  8.490 \else
  8.491 \save@sf@q{\penalty\M
  8.492 \raise.2ex\hbox{\scriptscriptstyle>}\allowhyphens}%
  8.493 \fi}
```

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

```
8.494 \ProvideTextCommandDefault{\guilsinglleft}{%
8.495 \UseTextSymbol{OT1}{\guilsinglleft}}
8.496 \ProvideTextCommandDefault{\guilsinglright}{%
8.497 \UseTextSymbol{OT1}{\guilsinglright}}
```

## 8.9 Letters

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

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

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

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

```
8.504 \ProvideTextCommandDefault{\ij}{%
8.505 \UseTextSymbol{OT1}{\ij}}
8.506 \ProvideTextCommandDefault{\IJ}{%
8.507 \UseTextSymbol{OT1}{\IJ}}
```

`\dj` The croatian language needs the letters `\dj` and `\DJ`; they are available in the T1 `\DJ` 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](mailto:stipcevic@olimp.irb.hr)).

```
8.508 \def\crrtic@{\hrule height0.1ex width0.3em}
8.509 \def\crttic@{\hrule height0.1ex width0.33em}
8.510 %
8.511 \def\dj@@#1#2#3{%
8.512 \leavevmode\kern#1em\raise#2ex\vbox{\crrtic@}\kern#3em}
8.513 \def\dj@{%
8.514 \ifcase\the\fam
8.515 \dj@@{0.25}{1.28}{-0.50}\or %roman
8.516 \dj@@{0.25}{1.28}{-0.50}\or %roman
8.517 \dj@@{0.25}{1.28}{-0.50}\or %roman
8.518 \dj@@{0.25}{1.28}{-0.50}\or %roman
8.519 \dj@@{0.33}{1.29}{-0.63}\or %italic
8.520 \dj@@{0.35}{1.30}{-0.65}\or %slanted
8.521 \dj@@{0.25}{1.25}{-0.55}\or %bold
8.522 \dj@@{0.25}{1.15}{-0.55}\or %tt
8.523 \dj@@{-0.04}{0.72}{-0.26}\or%csc
8.524 \dj@@{0.35}{1.30}{-0.65} %slanted
8.525 \else\dj@@{0.25}{1.25}{-0.55}\fi}%bold
8.526 %
8.527 \def\DJ@@#1#2#3{%
8.528 \leavevmode\kern#1em\raise#2ex\vbox{\crttic@}\kern#3em}
8.529 \def\DJ@{%
8.530 \ifcase\the\fam
8.531 \DJ@@{0.11}{0.92}{-0.36}\or
8.532 \DJ@@{0.11}{0.92}{-0.36}\or
8.533 \DJ@@{0.11}{0.92}{-0.36}\or
8.534 \DJ@@{0.11}{0.92}{-0.36}\or
8.535 \DJ@@{0.25}{0.92}{-0.50}\or
8.536 \DJ@@{0.22}{0.92}{-0.47}\or
8.537 \DJ@@{0.09}{0.90}{-0.34}\or
8.538 \DJ@@{0.02}{0.85}{-0.27}\or
8.539 \DJ@@{0.1}{0.86}{-0.35}\or
8.540 \DJ@@{0.22}{0.92}{-0.47}\else
8.541 \DJ@@{0.09}{0.90}{-0.34}\fi}
8.542 \DeclareTextCommand{\dj}{OT1}{\dj@ d}
8.543 \DeclareTextCommand{\DJ}{OT1}{\DJ@ D}
```

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

```
8.544 \ProvideTextCommandDefault{\dj}{%
```

```

8.545 \UseTextSymbol{OT1}{\dj}}
8.546 \ProvideTextCommandDefault{\DJ}{%
8.547 \UseTextSymbol{OT1}{\DJ}}

```

## 8.10 Shorthands for quotation marks

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

`\glq` The ‘german’ single quotes.

```

\grq8.548 \DeclareRobustCommand{\glq}{%
8.549 \ifmmode\mbox{\quotesinglbase}\else\quotesinglbase\fi}
8.550 \DeclareRobustCommand{\grq}{%
8.551 \ifmmode\mbox{\textquoteleft}\else\textquoteleft\fi}

```

`\glqq` The ‘german’ double quotes.

```

\grqq8.552 \DeclareRobustCommand{\glqq}{%
8.553 \ifmmode\mbox{\quotedblbase}\else\quotedblbase\fi}
8.554 \DeclareRobustCommand{\grqq}{%
8.555 \ifmmode\mbox{\textquotedblleft}\else\textquotedblleft\fi}

```

`\flq` The ‘french’ single quillemts.

```

\frq8.556 \DeclareRobustCommand{\flq}{%
8.557 \ifmmode\mbox{\quilsinglleft}\else\quilsinglleft\fi}
8.558 \DeclareRobustCommand{\frq}{%
8.559 \ifmmode\mbox{\quilsinglright}\else\quilsinglright\fi}

```

`\flqq` The ‘french’ double quillemts.

```

\frqq8.560 \DeclareRobustCommand{\flqq}{%
8.561 \ifmmode\mbox{\quilletleft}\else\quilletleft\fi}
8.562 \DeclareRobustCommand{\frqq}{%
8.563 \ifmmode\mbox{\quilletright}\else\quilletright\fi}

```

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

```

8.564 \def\umlauthigh{%
8.565 \def\bbl@umlauta##1{%
8.566 \expandafter\accent\csname\f@encoding dqpos\endcsname
8.567 ##1\allowhyphens}}%
8.568 \let\bbl@umlaute\bbl@umlauta}
8.569 \def\umlautlow{%
8.570 \def\bbl@umlauta{\protect\lower@umlaut}}
8.571 \def\umlautelow{%
8.572 \def\bbl@umlaute{\protect\lower@umlaut}}
8.573 \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  $\langle dimen \rangle$  register.

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

The following code fools T<sub>E</sub>X's `make_accent` procedure about the current x-height of the font to force another placement of the umlaut character.

```
8.577 \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.

```
8.578   {\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.)

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

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

```
8.582   \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.

```
8.583   \expandafter\accent\csname\f@encoding dqpos\endcsname
8.584   \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.

```
8.585 \AtBeginDocument{%
8.586   \DeclareTextCompositeCommand{\"}{OT1}{a}{\bbl@umlauta{a}}%
8.587   \DeclareTextCompositeCommand{\"}{OT1}{e}{\bbl@umlaute{e}}%
8.588   \DeclareTextCompositeCommand{\"}{OT1}{i}{\bbl@umlaute{\i}}%
8.589   \DeclareTextCompositeCommand{\"}{OT1}{i}{\bbl@umlaute{\i}}%
8.590   \DeclareTextCompositeCommand{\"}{OT1}{o}{\bbl@umlauta{o}}%
8.591   \DeclareTextCompositeCommand{\"}{OT1}{u}{\bbl@umlauta{u}}%
8.592   \DeclareTextCompositeCommand{\"}{OT1}{A}{\bbl@umlauta{A}}%
8.593   \DeclareTextCompositeCommand{\"}{OT1}{E}{\bbl@umlaute{E}}%
8.594   \DeclareTextCompositeCommand{\"}{OT1}{I}{\bbl@umlaute{I}}%
8.595   \DeclareTextCompositeCommand{\"}{OT1}{O}{\bbl@umlauta{O}}%
8.596   \DeclareTextCompositeCommand{\"}{OT1}{O}{\bbl@umlauta{U}}%
8.597 }
```

## 8.12 The redefinition of the style commands

The rest of the code in this file can only be processed by L<sup>A</sup>T<sub>E</sub>X, so we check the current format. If it is plain T<sub>E</sub>X, 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`.

```
8.598 {\def\format{lplain}
8.599 \ifx\fmtname\format
8.600 \else
8.601   \def\format{LaTeX2e}
8.602   \ifx\fmtname\format
8.603   \else
8.604     \aftergroup\endinput
8.605   \fi
8.606 \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`.

```
8.607 <*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 <code>\chapter</code> and <code>\opening</code> macros are undefined
report and book	:	the <code>\chapter</code> macro is defined and the <code>\opening</code> is undefined
letter	:	the <code>\chapter</code> macro is undefined and the <code>\opening</code> 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 `artike11.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'.

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

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

```
8.610 \ifx\undefined\opening
8.611   \ifx\undefined\chapter
8.612     \def\doc@style{1}%
8.613   \else
```

```

8.614     \def\doc@style{2}%
8.615     \fi
8.616     \else
8.617     \def\doc@style{3}%
8.618     \fi%
8.619 \fi%

```

### 8.12.1 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:

```

8.620 \@ifundefined{figurename}{\def\fnm@figure{\figurename} \thefigure}}{}
8.621 \@ifundefined{tablename}{\def\fnm@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.

```

8.622 \ifcase \doc@style\relax
8.623 \or

```

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

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

```

8.624 \@ifundefined{contentsname}%
8.625     {\def\tableofcontents{\section*{\contentsname\@mkboth
8.626         {\uppercase{\contentsname}}{\uppercase{\contentsname}}}%
8.627         \@starttoc{toc}}}{}}
8.628
8.629 \@ifundefined{listfigurename}%
8.630     {\def\listoffigures{\section*{\listfigurename\@mkboth
8.631         {\uppercase{\listfigurename}}{\uppercase{\listfigurename}}}%
8.632         \@starttoc{lof}}}{}}
8.633
8.634 \@ifundefined{listtablename}%
8.635     {\def\listoftables{\section*{\listtablename\@mkboth
8.636         {\uppercase{\listtablename}}{\uppercase{\listtablename}}}%
8.637         \@starttoc{lot}}}{}}

```

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

```

8.638 \@ifundefined{refname}%
8.639     {\def\thebibliography#1{\section*{\refname
8.640         \@mkboth{\uppercase{\refname}}{\uppercase{\refname}}}%
8.641         \list{[arabic{enumi}]}{\settowidth\labelwidth{[#1]}%
8.642             \leftmargin\labelwidth
8.643             \advance\leftmargin\labelsep
8.644             \usecounter{enumi}}}%
8.645         \def\newblock{\hskip.11em plus.33em minus.07em}%
8.646         \sloppy\clubpenalty4000\widowpenalty\clubpenalty
8.647         \sfcode'\.=1000\relax}}{}
8.648

```

---

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

```

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

```

The abstract environment:

```

8.656 \@ifundefined{abstractname}%
8.657   {\def\abstract{\if@twocolumn
8.658     \section*{\abstractname}%
8.659     \else \small
8.660     \begin{center}%
8.661     {\bf \abstractname\vspace{-.5em}\vspace{\z@}}%
8.662     \end{center}%
8.663     \quotation
8.664     \fi}}{}

```

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

```

8.665 \@ifundefined{partname}%
8.666   {\def\@part[#1]#2{\ifnum \c@secnumdepth >\m@ne
8.667     \refstepcounter{part}%
8.668     \addcontentsline{toc}{part}{\thepart
8.669     \hspace{1em}#1}\else
8.670     \addcontentsline{toc}{part}{#1}\fi
8.671     {\parindent\z@ \raggedright
8.672     \ifnum \c@secnumdepth >\m@ne
8.673       \Large \bf \partname{} \thepart
8.674       \par \nobreak
8.675       \fi
8.676       \huge \bf
8.677       #2\markboth{}{}\par}%
8.678     \nobreak
8.679     \vskip 3ex\@afterheading}%
8.680   }{}

```

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

```
8.681 \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:

```

8.682 \@ifundefined{contentsname}%
8.683   {\def\tableofcontents{\@restonecolfalse
8.684     \if@twocolumn\@restonecoltrue\onecolumn
8.685     \fi\chapter*{\contentsname\@mkboth
8.686       {\uppercase{\contentsname}}{\uppercase{\contentsname}}}%
8.687     \@starttoc{toc}%
8.688     \csname if@restonecol\endcsname\twocolumn
8.689     \csname fi\endcsname}}{}
8.690
8.691 \@ifundefined{listfigurename}

```

```

8.692   {\def\listoffigures{\@restonecolfalse
8.693     \if@twocolumn\@restonecoltrue\onecolumn
8.694     \fi\chapter*{\listfigurename\@mkboth
8.695       {\uppercase{\listfigurename}}{\uppercase{\listfigurename}}}%
8.696     \@starttoc{lof}%
8.697     \csname if@restonecol\endcsname\twocolumn
8.698     \csname fi\endcsname}}{}
8.699
8.700 \@ifundefined{listtablename}
8.701   {\def\listoftables{\@restonecolfalse
8.702     \if@twocolumn\@restonecoltrue\onecolumn
8.703     \fi\chapter*{\listtablename\@mkboth
8.704       {\uppercase{\listtablename}}{\uppercase{\listtablename}}}%
8.705     \@starttoc{lot}%
8.706     \csname if@restonecol\endcsname\twocolumn
8.707     \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.

```

8.708 \@ifundefined{bibname}
8.709   {\def\thebibliography#1{\chapter*{\bibname
8.710     \@mkboth{\uppercase{\bibname}}{\uppercase{\bibname}}}%
8.711     \list{[arabic{enumi}]}{\settowidth\labelwidth{[#1]}%
8.712     \leftmargin\labelwidth \advance\leftmargin\labelsep
8.713     \usecounter{enumi}}}%
8.714     \def\newblock{\hskip.11em plus.33em minus.07em}%
8.715     \sloppy\clubpenalty4000\widowpenalty\clubpenalty
8.716     \sfcode\.=1000\relax}}{}
8.717
8.718 \@ifundefined{indexname}
8.719   {\def\theindex{\@restonecoltrue\if@twocolumn\@restonecolfalse\fi
8.720     \columnseprule \z@
8.721     \columnsep 35pt\twocolumn[\@makeschapterhead{\indexname}]%
8.722     \@mkboth{\uppercase{\indexname}}{\uppercase{\indexname}}%
8.723     \thispagestyle{plain}%
8.724     \parskip\z@ plus.3pt\parindent\z@ \let\item\@idxitem}}{}

```

Here is the abstract environment:

```

8.725 \@ifundefined{abstractname}
8.726   {\def\abstract{\titlepage
8.727     \null\vfil
8.728     \begin{center}%
8.729     {\bf \abstractname}%
8.730     \end{center}}{}

```

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

```

8.731 \@ifundefined{chaptername}{\def\@chapapp{\chaptername}}{}
8.732 %
8.733 \@ifundefined{appendixname}
8.734   {\def\appendix{\par
8.735     \setcounter{chapter}{0}%
8.736     \setcounter{section}{0}%

```

```

8.737     \def\@chapapp{\appendixname}%
8.738     \def\thechapter{\Alph{chapter}}}\fi}
8.739 %
8.740 \@ifundefined{partname}
8.741   {\def\@part[#1]#2{\ifnum \c@secnumdepth >-2\relax
8.742     \refstepcounter{part}%
8.743     \addcontentsline{toc}{part}{\thepart
8.744     \hspace{1em}#1}\else
8.745     \addcontentsline{toc}{part}{#1}\fi
8.746     \markboth{}{}}%
8.747   {\centering
8.748     \ifnum \c@secnumdepth >-2\relax
8.749     \huge\bf \partname{} \thepart
8.750     \par
8.751     \vskip 20pt \fi
8.752     \Huge \bf
8.753     #1\par}\@endpart}}}\fi}
8.754 \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.

```

8.755 \@ifundefined{ccname}%
8.756   {\def\cc#1{\par\noindent
8.757     \parbox[t]{\textwidth}%
8.758     {\@hangfrom{\rm \ccname : }\ignorespaces #1\strut}\par}}\fi}
8.759
8.760 \@ifundefined{enclname}%
8.761   {\def\encl#1{\par\noindent
8.762     \parbox[t]{\textwidth}%
8.763     {\@hangfrom{\rm \enclname : }\ignorespaces #1\strut}\par}}\fi}

```

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

```

8.764 \@ifundefined{headtoname}
8.765   {\def\ps@headings{%
8.766     \def\@oddhead{sl \headtoname} \ignorespaces\toname \hfil
8.767     \@date \hfil \pagename{} \thepage}%
8.768     \def\@oddfoot{}}}\fi}

```

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.

```
8.769 \fi
```

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

```

8.770 </names>
8.771 </core>

```

## 8.13 Cross referencing macros

The  $\text{\LaTeX}$  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 T<sub>E</sub>Xbook [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’.

`\babel@sanitize@arg` To call a macro with a ‘sanitized’ argument, instead of `\A{\B}` one would write `\babel@sanitize@arg{\A}{\B}`. (But be careful, this macro is not fully expandable!)

```
8.772 %tmp%\long\def\babel@sanitize@arg#1#2{\bgroup\def\@tempa{#2}%
8.773 %tmp% \expandafter\babel@strip@meaning\meaning\@tempa\relax{#1}}
8.774 %tmp%\def\babel@strip@meaning#1->#2\relax#3{\egroup #3{#2}}
```

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 L<sup>A</sup>T<sub>E</sub>X macros completely in case their definitions change (they have changed in the past).

`\label` The `\label` macro is one of the cross referencing macros affected. First we save its original definition.

```
8.775 %tmp%\let\LTX@label=\label
```

Then the macro `label` is redefined.

```
8.776 %tmp%\def\label#1{\babel@sanitize@arg\LTX@label{#1}}
```

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

```
8.777 (*core | shorthands)
8.778 \let\bb1@newlabel\newlabel
8.779 \def\newlabel#1#2{%
8.780 \@safe@activestrue\bb1@newlabel{#1}{#2}\@safe@activesfalse}
```

`\@testdef` An internal L<sup>A</sup>T<sub>E</sub>X macro used to test if the labels that have been written on the `.aux` file have changed. It is called by the `\enddocument` macro.

```
8.781 \let\bb1@testdef@testdef
8.782 \def@testdef#1#2#3{%
8.783 \@safe@activestrue\bb1@testdef{#1}{#2}{#3}\@safe@activesfalse}
```

`\ref` The same holds for the macro `\ref` that references a label and `\pageref` to reference a page. While we change these macros, we make them robust as well to prevent problems when they should become expanded at the wrong moment.

```
8.784 \let\bb1@ref\ref
8.785 \edef\ref{\noexpand\protect
```

```

8.786 \expandafter\noexpand\csname bblref \endcsname}
8.787 \expandafter\def\csname bblref \endcsname#1{%
8.788 \@safe@activestrue\bbl@ref{#1}\@safe@activesfalse}

8.789 \let\bbl@pageref\pageref
8.790 \edef\pageref{\noexpand\protect
8.791 \expandafter\noexpand\csname bblpageref \endcsname}
8.792 \expandafter\def\csname bblpageref \endcsname#1{%
8.793 \@safe@activestrue\bbl@pageref{#1}\@safe@activesfalse}

```

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

```

8.794 %tmp%\let\bbl@cite\@cite
8.795 %tmp%\def\@cite[#1]#2{\babel@sanitize@arg{\bbl@cite[#1]}{#2}}

```

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

```

8.796 %tmp%\let\bbl@nocite\nocite
8.797 %tmp%\def\nocite#1{\babel@sanitize@arg\bbl@nocite{#1}}

```

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

```

8.798 %tmp%\let\bbl@bibcite\bibcite
8.799 %tmp%\def\bibcite#1#2{\babel@sanitize@arg\bbl@bibcite{#1}{#2}}

```

`\@bibitem` One of the two internal L<sup>A</sup>T<sub>E</sub>X macros called by `\bibitem` that write the citation label on the `.aux` file.

```

8.800 %tmp%\let\bbl@bibitem\@bibitem
8.801 %tmp%\def\@bibitem#1{\babel@sanitize@arg\bbl@bibitem{#1}}

```

`\@lbibitem` The other of the two internal L<sup>A</sup>T<sub>E</sub>X macros called by `\bibitem` that write the citation label on the `.aux` file.

```

8.802 %tmp%\let\bbl@lbibitem\@lbibitem
8.803 %tmp%\def\@lbibitem[#1]#2{\babel@sanitize@arg{\bbl@lbibitem[#1]}{#2}}

```

```

8.804 </core | shorthands)

```

## 9 Local Language Configuration

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

```

9.1 (*core)
9.2 \def\loadlocalcfg#1{%
9.3 \InputIfFileExists{#1.cfg}
9.4     {\typeout{*****~^~J%
9.5             * Local config file #1.cfg used~^~J%
9.6             *}%
9.7     }
9.8     {}}
9.9 </core)

```

## 10 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 specific files you can run the file `user.drv` through L<sup>A</sup>T<sub>E</sub>X to get a user guide.

```

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

```

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

```

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

```

The code lines are numbered within sections,

```
10.45 <!\user>
10.46 \@addtoreset{CodelineNo}{section}
10.47 \renewcommand\theCodelineNo{%
10.48   \reset@font\scriptsize\thesection.\arabic{CodelineNo}}
```

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

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

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

```
10.53 \renewenvironment{theglossary}{%
10.54   \glossary@prologue%
10.55   \GlossaryParms \let\item\idxitem \ignorespaces}%
10.56   {}
10.57 </!\user>
10.58 \makeatother
```

A few shorthands used in the documentation

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

Some more definitions needed in the documentation.

```
10.65 \newcommand*\note[1]{\textbf{#1}}
10.66 %\newcommand*\note[1]{}
10.67 \newcommand*\bsl{\protect\bslash}
10.68 \newcommand*\Lopt[1]{\textsf{#1}}
10.69 \newcommand*\file[1]{\texttt{#1}}
10.70 \newcommand*\pkg[1]{\texttt{#1}}
10.71 \newcommand*\langdefile[1]{%
10.72 (-user) \clearpage
10.73 \DocInput{#1}}
```

When a full index should be generated unomment the line with `\EnableCrossres`.

Beware, processing may take some time. Use `\DisableCrossrefs` when the index is ready.

```
10.74 % \EnableCrossrefs
10.75 \DisableCrossrefs
```

Include the change log.

```
10.76 (-user)\RecordChanges
```

The index should use the linenumbers of the code.

```
10.77 (-user)\CodelineIndex
```

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

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

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

10.79 `(+user)\OnlyDescription`

Here starts the document

10.80 `\begin{document}`

10.81 `\DocInput{babel.dtx}`

All the language definition files.

10.82 `(+user)\clearpage`

10.83 `\langdeffile{esperant.dtx}`

10.84 `\langdeffile{dutch.dtx}`

10.85 `\langdeffile{english.dtx}`

10.86 `\langdeffile{germanb.dtx}`

10.87 `%`

10.88 `\langdeffile{breton.dtx}`

10.89 `\langdeffile{irish.dtx}`

10.90 `\langdeffile{scottish.dtx}`

10.91 `%`

10.92 `\langdeffile{francais.dtx}`

10.93 `\langdeffile{italian.dtx}`

10.94 `\langdeffile{portuges.dtx}`

10.95 `\langdeffile{spanish.dtx}`

10.96 `\langdeffile{catalan.dtx}`

10.97 `\langdeffile{galician.dtx}`

10.98 `\langdeffile{romanian.dtx}`

10.99 `%`

10.100 `\langdeffile{danish.dtx}`

10.101 `\langdeffile{norsk.dtx}`

10.102 `\langdeffile{swedish.dtx}`

10.103 `%`

10.104 `\langdeffile{finnish.dtx}`

10.105 `\langdeffile{magyar.dtx}`

10.106 `\langdeffile{estonian.dtx}`

10.107 `%`

10.108 `\langdeffile{croatian.dtx}`

10.109 `\langdeffile{czech.dtx}`

10.110 `\langdeffile{polish.dtx}`

10.111 `\langdeffile{slovak.dtx}`

10.112 `\langdeffile{slovene.dtx}`

10.113 `%\langdeffile{russian.dtx}`

10.114 `%`

10.115 `\langdeffile{lsorbian.dtx}`

10.116 `\langdeffile{usorbian.dtx}`

10.117 `\langdeffile{turkish.dtx}`

10.118 `%`

10.119 `\langdeffile{bahasa.dtx}`

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

10.120 `(*!user)`

10.121 `\clearpage`

10.122 `\def\filename{index}`

10.123 `\PrintIndex`

10.124 `\clearpage`

10.125 `\def\filename{changes}`

10.126 `\PrintChanges`

```
10.127 </!user>
10.128 \end{document}
10.129 </driver>
```

## 11 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.

## 12 Acknowledgements

I would like to thank all who volunteered as  $\beta$ -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. Werenfried Spit supplied the files for the Russian 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  $\TeX$ book*, 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:  $\TeX$ hax 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.

## 13 The Esperanto language

The file `esperant.dtx`<sup>10</sup> 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.

<code>^u</code>	gives <code>ŭ</code> , with hyphenation in the rest of the word allowed
<code>^U</code>	gives <code>Ŭ</code> , with hyphenation in the rest of the word allowed
<code>^h</code>	prevents <code>ĥ</code> from becoming too tall
<code>^j</code>	gives <code>ĵ</code>
<code>^l</code>	inserts a <code>\discretionary{-}{ }{ }</code>
<code>^a</code>	gives <code>â</code> with hyphenation in the rest of the word allowed, this works for all other letters as well.

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

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

```
13.1 <*code>
13.2 \ifx\undefined\captionesperanto
13.3 \else
13.4 \selectlanguage{esperanto}
13.5 \expandafter\endinput
13.6 \fi
```

`\atcatcode` This file, `esperant.ldf`, may have been read while  $\TeX$  is in the middle of processing a document, so we have to make sure the category code of `@` is ‘letter’ while this file is being read. We save the category code of the `@`-sign in `\atcatcode` and make it ‘letter’. Later the category code can be restored to whatever it was before.

```
13.7 \chardef\atcatcode=\catcode‘\@
13.8 \catcode‘\@=11\relax
```

Now we determine whether the the common macros from the file `babel.def` need to be read. We can be in one of two situations: either another language option has been read earlier on, in which case that other option has already read `babel.def`, or `esperanto` is the first language option to be processed. In that case we need to read `babel.def` right here before we continue.

```
13.9 \ifx\undefined\babel@core@loaded\input babel.def\relax\fi
```

Another check that has to be made, is if another language definition file has been read already. In that case its definitions have been activated. This might interfere with definitions this file tries to make. Therefore we make sure that we cancel any special definitions. This can be done by checking the existence of the

---

<sup>10</sup>The file described in this section has version number ? and was last revised on ?. 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.

macro `\originalTeX`. If it exists we simply execute it, otherwise it is `\let` to `\empty`.

```
13.10 \ifx\undefined\originalTeX \let\originalTeX\empty \else\originalTeX\fi
```

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.

```
13.11 \ifx\undefined\l@esperanto
```

```
13.12 \nopatterns{Esperanto}
```

```
13.13 \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.

`\captionesperanto` The macro `\captionesperanto` defines all strings used in the four standard documentclasses provided with L<sup>A</sup>T<sub>E</sub>X.

```
13.14 \addto\captionesperanto{%
13.15 \def\prefacename{Anta\u{u}parolo}%
13.16 \def\refname{Cita\`j{oj}}%
13.17 \def\abstractname{Resumo}%
13.18 \def\bibName{Bibliografio}%
13.19 \def\chaptername{{\^C}apitro}%
13.20 \def\appendixname{Apendico}%
13.21 \def\contentsname{Enhavo}%
13.22 \def\listfigurename{Listo de figuroj}%
13.23 \def\listtablename{Listo de tabeloj}%
13.24 \def\indexname{Indekso}%
13.25 \def\figurename{Figuro}%
13.26 \def\tablename{Tabelo}%
13.27 \def\partname{Parto}%
13.28 \def\enclname{Aldono(j)}%
13.29 \def\ccname{Kopie al}%
13.30 \def\headtoname{Al}%
13.31 \def\pagename{Pa\^go}%
13.32 \def\subjectname{Temo}%
13.33 \def\seenname{vidu} a~u: vd.
13.34 \def\alsoname{vidu anka\u{u}} a~u vd. anka\u{u}
13.35 \def\proofname{Proof}% <-- needs translation
13.36 }
```

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

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

`\extrasesperanto` 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  $\hat{}$  character is made active. This is done once, later on its definition may vary.

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

And here are the uses of the active  $\hat{}$ :

```
13.46 \declare@shorthand{esperanto}{^u}{\u u\allowhyphens}
13.47 \declare@shorthand{esperanto}{^U}{\u U\allowhyphens}
13.48 \declare@shorthand{esperanto}{^h}{h\llap{^}}\allowhyphens}
13.49 \declare@shorthand{esperanto}{^j}{\^j}\allowhyphens}
13.50 \declare@shorthand{esperanto}{^|}{\discretionary{-}{-}{}}\allowhyphens}
```

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

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

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

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

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

And the 33 other cases.

```
13.55 \def\@iesper#1{\ifcase#1\or \or \or \or \or \or e\or f\or g\or ^g\or
13.56 h\or h\llap{^}}\or i\or j\or ^j\or k\or l\or m\or n\or o\or
13.57 p\or s\or ^s\or t\or u\or \u{u}\or v\or z\else\@ctrerr\fi}
13.58 \def\@Iesper#1{\ifcase#1\or \or \or \or \or \or E\or F\or G\or ^G\or
13.59 H\or ^H\or I\or J\or ^J\or K\or L\or M\or N\or O\or
13.60 P\or S\or ^S\or T\or U\or \u{U}\or V\or Z\else\@ctrerr\fi}
```

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

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

It is possible that a site might need to add some extra code to the babel macros. To enable this we load a local configuration file, `esperant.cfg` if it is found on  $\TeX$ ' search path.

```
13.66 \loadlocalcfg{esperant}
```

Our last action is to make a note that the commands we have just defined, will be executed by calling the macro  $\backslash$ selectlanguage at the beginning of the document.

```
13.67 \main@language{esperanto}
```

Finally, the category code of @ is reset to its original value. The macrospace used by `\atcatcode` is freed.

```
13.68 \catcode'\@=\atcatcode \let\atcatcode\relax
13.69 </code>
```

## 14 The Dutch language

The file `dutch.dtx`<sup>11</sup> defines all the language-specific macros for the Dutch language.

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.
"‘	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`

As this file needs to be read only once, we check whether it was read before. If it was, the command `\captionsdutch` is already defined, so we can stop processing. If this command is undefined we proceed with the various definitions and first show the current version of this file.

```
14.1 {*code}
14.2 \ifx\undefined\captionsdutch
14.3 \else
14.4 \selectlanguage{dutch}
14.5 \expandafter\endinput
14.6 \fi
```

`\atcatcode` This file, `dutch.ldf`, may have been read while  $\TeX$  is in the middle of processing a document, so we have to make sure the category code of `@` is ‘letter’ while this file is being read. We save the category code of the `@`-sign in `\atcatcode` and make it ‘letter’. Later the category code can be restored to whatever it was before.

```
14.7 \chardef\atcatcode=\catcode'\@
14.8 \catcode'\@=11\relax
```

Now we determine whether the common macros from the file `babel.def` need to be read. We can be in one of two situations: either another language option has

---

<sup>11</sup>The file described in this section has version number ?, and was last revised on ?.

been read earlier on, in which case that other option has already read `babel.def`, or `dutch` is the first language option to be processed. In that case we need to read `babel.def` right here before we continue.

```
14.9 \ifx\undefined\babel@core@loaded\input babel.def\relax\fi
```

Another check that has to be made, is if another language definition file has been read already. In that case its definitions have been activated. This might interfere with definitions this file tries to make. Therefore we make sure that we cancel any special definitions. This can be done by checking the existence of the macro `\originalTeX`. If it exists we simply execute it, otherwise it is `\let` to `\empty`.

```
14.10 \ifx\undefined\originalTeX \let\originalTeX\empty \fi
```

```
14.11 \originalTeX
```

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` to see whether we have to do something here.

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

```
14.13 \nopatterns{Dutch}
```

```
14.14 \adddialect\l@dutch0
```

```
14.15 \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 L<sup>A</sup>T<sub>E</sub>X.

```
14.16 \begingroup
```

```
14.17 \catcode'\\"active
```

```
14.18 \def\x{\endgroup
```

```
14.19 \addto\captionsdutch{%
```

```
14.20 \def\prefacename{Voorwoord}%
```

```
14.21 \def\refname{Referenties}%
```

```
14.22 \def\abstractname{Samenvatting}%
```

```
14.23 \def\bibName{Bibliografie}%
```

```
14.24 \def\chaptername{Hoofdstuk}%
```

```
14.25 \def\appendixname{B"ylage}%
```

```
14.26 \def\contentsname{Inhoudsopgave}%
```

```
14.27 \def\listfigurename{L"yst van figuren}%
```

```
14.28 \def\listtablename{L"yst van tabellen}%
```

```
14.29 \def\indexname{Index}%
```

```
14.30 \def\figurename{Figuur}%
```

```
14.31 \def\tablename{Tabel}%
```

```
14.32 \def\partname{Deel}%
```

```
14.33 \def\enclname{B"ylage(n)}%
```

```
14.34 \def\ccname{cc}%
```

```
14.35 \def\headtoname{Aan}%
```

```
14.36 \def\pagename{Pagina}%
```

```
14.37 \def\seename{zie}%
```

```
14.38 \def\alsoname{zie ook}%
```

```
14.39 \def\proofname{Bewijs}%
```

```
14.40 }
```

```
14.41 }\x
```

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

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

`\extrasdutch` The macro `\extrasdutch` will perform all the extra definitions needed for the Dutch language. The macro `\noextrasdutch` is used to cancel the actions of `\extrasdutch`.

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.

```
14.47 \initiate@active@char{"}
14.48 \addto\extrasdutch{\languageshorthands{dutch}}
14.49 \addto\extrasdutch{\bbl@activate{}}
14.50 %\addto\noextrasdutch{\bbl@deactivate{}}
```

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

```
14.51 \addto\extrasdutch{\umlautlow\umlautelow}
14.52 \addto\noextrasdutch{\umlauthigh}
```

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

```
14.53 \def\dutchhyphenmins{\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.

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

Now we can define the doublequote macros: the tremas,

```
14.55 \declare@shorthand{dutch}{a}{\textormath{\@trema a}{\ddot a}}
14.56 \declare@shorthand{dutch}{e}{\textormath{\@trema e}{\ddot e}}
14.57 \declare@shorthand{dutch}{i}{%
14.58 \textormath{\discretionary{-}{i}{\{"i}}{\ddot \imath}}
14.59 \declare@shorthand{dutch}{o}{\textormath{\@trema o}{\ddot o}}
14.60 \declare@shorthand{dutch}{u}{\textormath{\@trema u}{\ddot u}}
```

dutch quotes,

```
14.61 \declare@shorthand{dutch}{"'}{%
14.62 \textormath{\quotedblbase}}{\mbox{\quotedblbase}}
14.63 \declare@shorthand{dutch}{"'}{%
14.64 \textormath{\textquotedblright}}{\mbox{\textquotedblright}}
```

and some additional commands:

```
14.65 \declare@shorthand{dutch}{-}{\allowhyphens-\allowhyphens}
14.66 \declare@shorthand{dutch}{"}{|}{%
14.67   \textormath{\discretionary{-}{}{\kern.03em}}{}}
14.68 \declare@shorthand{dutch}{"}{z}{\hskip\z@skip}
14.69 \declare@shorthand{dutch}{"}y}{\textormath{\i j}}{\ddot y}}
14.70 \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.

```
14.71 \addto\extrasdutch{\babel@save\-\}
14.72 \addto\extrasdutch{\def\-\{\allowhyphens
14.73   \discretionary{-}{}{\allowhyphens}}
```

It is possible that a site might need to add some extra code to the babel macros. To enable this we load a local configuration file, `dutch.cfg` if it is found on  $\TeX$ ' search path.

```
14.74 \loadlocalcfg{dutch}
```

Our last action is to make a note that the commands we have just defined, will be executed by calling the macro `\selectlanguage` at the beginning of the document.

```
14.75 \main@language{dutch}
```

Finally, the category code of `@` is reset to its original value. The macrospace used by `\atcatcode` is freed.

```
14.76 \catcode'\@=\atcatcode \let\atcatcode\relax
14.77 </code>
```

## 15 The English language

The file `english.dtx`<sup>12</sup> 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.

As this file needs to be read only once, we check whether it was read before. If it was, the command `\captionenglish` is already defined, so we can stop processing. If this command is undefined we proceed with the various definitions and first show the current version of this file.

```
15.1 <*code>
15.2 \ifx\undefined\captionenglish
15.3 \else
15.4   \selectlanguage{english}
15.5   \expandafter\endinput
15.6 \fi
```

`\atcatcode` This file, `english.ldf`, may have been read while  $\TeX$  is in the middle of processing a document, so we have to make sure the category code of `@` is ‘letter’ while this file is being read. We save the category code of the `@`-sign in `\atcatcode` and make it ‘letter’. Later the category code can be restored to whatever it was before.

```
15.7 \chardef\atcatcode=\catcode'\@
15.8 \catcode'\@=11\relax
```

Now we determine whether the common macros from the file `babel.def` need to be read. We can be in one of two situations: either another language option has been read earlier on, in which case that other option has already read `babel.def`, or `english` is the first language option to be processed. In that case we need to read `babel.def` right here before we continue.

```
15.9 \ifx\undefined\babel@core@loaded\input babel.def\relax\fi
```

Another check that has to be made, is if another language definition file has been read already. In that case its definitions have been activated. This might interfere with definitions this file tries to make. Therefore we make sure that we cancel any special definitions. This can be done by checking the existence of the macro `\originalTeX`. If it exists we simply execute it, otherwise it is `\let` to `\empty`.

```
15.10 \ifx\undefined\originalTeX \let\originalTeX\empty\fi
15.11 \originalTeX
```

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.

```
15.12 \ifx\undefined\l@english
15.13   \ifx\undefined\l@UKenglish
15.14     \nopatterns{English}
15.15     \adddialect\l@english0
15.16   \else
15.17     \let\l@english\l@UKenglish
```

---

<sup>12</sup>The file described in this section has version number ? and was last revised on ?.

```
15.18 \fi
15.19 \fi
```

For the American version of these definitions we just add a “dialect”. Also, the macros `\captionsamerican` and `\extrasamerican` are `\let` to their English counterparts when these parts are defined.

```
15.20 \ifx\l@american\undefined
15.21 \adddialect\l@american\l@english
15.22 \fi
```

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

`\captionsenglish` The macro `\captionsenglish` defines all strings used in the four standard document classes provided with L<sup>A</sup>T<sub>E</sub>X.

```
15.23 \addto\captionsenglish{%
15.24 \def\prefacename{Preface}%
15.25 \def\refname{References}%
15.26 \def\abstractname{Abstract}%
15.27 \def\bibname{Bibliography}%
15.28 \def\chaptername{Chapter}%
15.29 \def\appendixname{Appendix}%
15.30 \def\contentsname{Contents}%
15.31 \def\listfigurename{List of Figures}%
15.32 \def\listtablename{List of Tables}%
15.33 \def\indexname{Index}%
15.34 \def\figurename{Figure}%
15.35 \def\tablename{Table}%
15.36 \def\partname{Part}%
15.37 \def\enclname{encl}%
15.38 \def\ccname{cc}%
15.39 \def\headtoname{To}%
15.40 \def\pagename{Page}%
15.41 \def\seename{see}%
15.42 \def\alsoname{see also}%
15.43 \def\proofname{Proof}%
15.44 }
```

`\captionsamerican` The ‘captions’ are the same for both versions of the language, so we can `\let` the macro `\captionsamerican` be equal to `\captionsenglish`.

```
15.45 \let\captionsamerican\captionsenglish
```

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

```
15.46 \def\dateenglish{%
15.47 \def\today{\ifcase\day\or
15.48 1st\or 2nd\or 3rd\or 4th\or 5th\or
15.49 6th\or 7th\or 8th\or 9th\or 10th\or
15.50 11th\or 12th\or 13th\or 14th\or 15th\or
15.51 16th\or 17th\or 18th\or 19th\or 20th\or
15.52 21st\or 22nd\or 23rd\or 24th\or 25th\or
15.53 26th\or 27th\or 28th\or 29th\or 30th\or
15.54 31st\fi~\ifcase\month\or
15.55 January\or February\or March\or April\or May\or June\or
```

```

15.56 July\or August\or September\or October\or November\or December\fi
15.57 \space \number\year}}

```

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

```

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

```

`\extrasenglish` 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.

```

15.63 \addto\extrasenglish{}
15.64 \addto\noextrasenglish{}

```

`\extrasamerican` Also for the “american” variant no extra definitions are needed at the moment.

```

\noextrasamerican15.65 \let\extrasamerican\extrasenglish
15.66 \let\noextrasamerican\noextrasenglish

```

It is possible that a site might need to add some extra code to the babel macros. To enable this we load a local configuration file, `english.cfg` if it is found on T<sub>E</sub>X’ search path.

```

15.67 \loadlocalcfg{english}

```

Our last action is to make a note that the commands we have just defined, will be executed by calling the macro `\selectlanguage` at the beginning of the document.

```

15.68 \main@language{english}

```

Finally, the category code of `@` is reset to its original value. The macrospace used by `\atcatcode` is freed.

```

15.69 \catcode'\@=\atcatcode \let\atcatcode\relax
15.70 </code>

```

## 16 The German language

The file `germanb.dtx`<sup>13</sup> defines all the language definition macros for the German language as well as for the Austrian dialect of this language<sup>14</sup>.

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 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.
"‘	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.

\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`

---

<sup>13</sup>The file described in this section has version number ? and was last revised on ?.

<sup>14</sup>This file is a re-implementation of Hubert Partl's `german.sty` version 2.5b, see [4].

As this file, `germanb.1df`, needs to be read only once, we check whether it was read before. If it was, the command `\captionsgerman` is already defined, so we can stop processing. If this command is undefined we proceed with the various definitions and first show the current version of this file.

```

16.1 (*code)
16.2 \ifx\undefined\captionsgerman
16.3 \else
16.4 \selectlanguage{german}
16.5 \expandafter\endinput
16.6 \fi

```

`\atcatcode` This file, `germanb.1df`, may have been read while  $\TeX$  is in the middle of processing a document, so we have to make sure the category code of `@` is ‘letter’ while this file is being read. We save the category code of the `@`-sign in `\atcatcode` and make it ‘letter’. Later the category code can be restored to whatever it was before.

```

16.7 \chardef\atcatcode=\catcode'\@
16.8 \catcode'\@=11\relax

```

Now we determine whether the common macros from the file `babel.def` need to be read. We can be in one of two situations: either another language option has been read earlier on, in which case that other option has already read `babel.def`, or `germanb` is the first language option to be processed. In that case we need to read `babel.def` right here before we continue.

```

16.9 \ifx\undefined\babel@core@loaded\input babel.def\relax\fi

```

Another check that has to be made, is if another language definition file has been read already. In that case its definitions have been activated. This might interfere with definitions this file tries to make. Therefore we make sure that we cancel any special definitions. This can be done by checking the existence of the macro `\originalTeX`. If it exists we simply execute it, otherwise it is `\let` to `\empty`.

```

16.10 \ifx\undefined\originalTeX \let\originalTeX\empty\fi
16.11 \originalTeX

```

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

```

16.12 \ifx\undefined\l@german
16.13 \@nopatterns{German}
16.14 \adddialect\l@german0
16.15 \fi

```

For the Austrian version of these definitions we just add another language. Also, the macros `\captionsaustrian` and `\extrasaustrian` are `\let` to their German counterparts if these parts are defined.

```

16.16 \adddialect\l@austrian\l@german

```

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

`\captionsgerman` The macro `\captionsgerman` defines all strings used in the four standard document classes provided with L<sup>A</sup>T<sub>E</sub>X.

```

16.17 \addto\captionsgerman{%
16.18   \def\prefacename{Vorwort}%
16.19   \def\refname{Literatur}%
16.20   \def\abstractname{Zusammenfassung}%
16.21   \def\bibName{Literaturverzeichnis}%
16.22   \def\chaptername{Kapitel}%
16.23   \def\appendixname{Anhang}%
16.24   \def\contentsname{Inhaltsverzeichnis}%   % oder nur: Inhalt
16.25   \def\listfigurename{Abbildungsverzeichnis}%
16.26   \def\listtablename{Tabellenverzeichnis}%
16.27   \def\indexname{Index}%
16.28   \def\figurename{Abbildung}%
16.29   \def\tablename{Tabelle}%                 % oder: Tafel
16.30   \def\partname{Teil}%
16.31   \def\enclname{Anlage(n)}%               % oder: Beilage(n)
16.32   \def\ccname{Verteiler}%                 % oder: Kopien an
16.33   \def\headtoname{An}%
16.34   \def\pagename{Seite}%
16.35   \def\seename{siehe}%
16.36   \def\alsoname{siehe auch}%
16.37   \def\proofname{Beweis}%
16.38 }

```

`\captionsgerman` The ‘captions’ are the same for both version of the language, so we can `\let` the macro `\captionsaustrian` be equal to `\captionsgerman`.

```
16.39 \let\captionsaustrian\captionsgerman
```

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

```

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

```

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

```

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

```

`\extrasgerman` The macro `\extrasgerman` 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.

```

16.47 \initiate@active@char{"}
16.48 \addto\extrasgerman{\languageshorthands{german}}
16.49 \addto\extrasgerman{\bbl@activate{"}}
16.50 %\addto\noextrasgerman{\bbl@deactivate{"}}

```

In order for T<sub>E</sub>X to be able to hyphenate German words which contain ‘ß’ (in the OT1 position  $\hat{\sim}Y$ ) we have to give the character a nonzero `\lccode` (see Appendix H, the T<sub>E</sub>Xbook).

```
16.51 \addto\extrasgerman{%
16.52   \babel@savevariable{\lccode'\hat{\sim}Y}%
16.53   \lccode'\hat{\sim}Y'\hat{\sim}Y}
```

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

```
16.54 \addto\extrasgerman{\babel@save\\"\umlautlow}
16.55 \addto\noextrasgerman{\umlauthigh}
```

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

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

`\extrasaustrian` For both versions of the language the same special macros are used, so we can  
`\noextrasaustrian` `\let` the austrian macros be equal to their german counterparts.

```
16.57 \let\extrasaustrian\extrasgerman
16.58 \let\noextrasaustrian\noextrasgerman
```

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 `\"`.

```
16.59 \begingroup \catcode'\\"12
16.60 \def\x{\endgroup
16.61   \def\@SS{\mathchar"7019 }
16.62   \def\dq{"}}
16.63 \x
```

`\german@dq@disc` For the discretionary macros we use this macro:

```
16.64 \def\german@dq@disc#1#2{%
16.65   \penalty\@M\discretionary{#2-}{\#1}\allowhyphens}
```

Now we can define the doublequote macros: the umlauts,

```
16.66 \declare@shorthand{german}{a}{\textormath{"{a}}{\ddot a}}
16.67 \declare@shorthand{german}{o}{\textormath{"{o}}{\ddot o}}
16.68 \declare@shorthand{german}{u}{\textormath{"{u}}{\ddot u}}
16.69 \declare@shorthand{german}{A}{\textormath{"{A}}{\ddot A}}
16.70 \declare@shorthand{german}{O}{\textormath{"{O}}{\ddot O}}
16.71 \declare@shorthand{german}{U}{\textormath{"{U}}{\ddot U}}

tremas,

16.72 \declare@shorthand{german}{e}{\textormath{"{e}}{\ddot e}}
16.73 \declare@shorthand{german}{E}{\textormath{"{E}}{\ddot E}}
16.74 \declare@shorthand{german}{i}{\textormath{"{i}}{\ddot i}}
16.75 \declare@shorthand{german}{I}{\textormath{"{I}}{\ddot I}}
```

```

    german es-zet (sharp s),
16.76 \declare@shorthand{german}{"s"}{\textormath{\ss}}{\@SS}}
16.77 \declare@shorthand{german}{"S"}{SS}
16.78 \declare@shorthand{german}{"z"}{\textormath{\ss}}{\@SS}}
16.79 \declare@shorthand{german}{"Z"}{SZ}

    german and french quotes,
16.80 \declare@shorthand{german}{"´"}{%
16.81   \textormath{\quotedblbase}}{\mbox{\quotedblbase}}
16.82 \declare@shorthand{german}{"’"}{%
16.83   \textormath{\textquotedblleft}}{\mbox{\textquotedblleft}}
16.84 \declare@shorthand{german}{"<"}{%
16.85   \textormath{\guillemotleft}}{\mbox{\guillemotleft}}
16.86 \declare@shorthand{german}{">"}{%
16.87   \textormath{\guillemotright}}{\mbox{\guillemotright}}

    discretionary commands
16.88 \declare@shorthand{german}{"c"}{\textormath{\german@dq@disc ck}}{c}
16.89 \declare@shorthand{german}{"C"}{\textormath{\german@dq@disc CK}}{C}
16.90 \declare@shorthand{german}{"f"}{\textormath{\german@dq@disc f{ff}}}{f}
16.91 \declare@shorthand{german}{"F"}{\textormath{\german@dq@disc F{FF}}}{F}
16.92 \declare@shorthand{german}{"l"}{\textormath{\german@dq@disc l{ll}}}{l}
16.93 \declare@shorthand{german}{"L"}{\textormath{\german@dq@disc L{LL}}}{L}
16.94 \declare@shorthand{german}{"m"}{\textormath{\german@dq@disc m{mm}}}{m}
16.95 \declare@shorthand{german}{"M"}{\textormath{\german@dq@disc M{MM}}}{M}
16.96 \declare@shorthand{german}{"n"}{\textormath{\german@dq@disc n{nn}}}{n}
16.97 \declare@shorthand{german}{"N"}{\textormath{\german@dq@disc N{NN}}}{N}
16.98 \declare@shorthand{german}{"p"}{\textormath{\german@dq@disc p{pp}}}{p}
16.99 \declare@shorthand{german}{"P"}{\textormath{\german@dq@disc P{PP}}}{P}
16.100 \declare@shorthand{german}{"r"}{\textormath{\german@dq@disc r{rr}}}{r}
16.101 \declare@shorthand{german}{"R"}{\textormath{\german@dq@disc R{RR}}}{R}
16.102 \declare@shorthand{german}{"t"}{\textormath{\german@dq@disc t{tt}}}{t}
16.103 \declare@shorthand{german}{"T"}{\textormath{\german@dq@disc T{TT}}}{T}

    and some additional commands:
16.104 \declare@shorthand{german}{"-"}{\penalty\@M-\allowhyphens}
16.105 \declare@shorthand{german}{"|"}{%
16.106   \textormath{\penalty\@M\discretionary{-}{-}{\kern.03em}%
16.107     \allowhyphens}}
16.108 \declare@shorthand{german}{""}{\hskip\z@skip}
16.109 \declare@shorthand{german}{"~"}{\textormath{\leavevmode\hbox{-}}{-}}
16.110 \declare@shorthand{german}{"="}{\penalty\@M-\hskip\z@skip}

```

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

```

\ck16.111 \def\mdqon{\bbl@activate{}}
16.112 \def\mdqoff{\bbl@deactivate{}}
16.113 \def\ck{\allowhyphens\discretionary{k-}{k}{ck}\allowhyphens}

```

It is possible that a site might need to add some extra code to the babel macros. To enable this we load a local configuration file, `germanb.cfg` if it is found on  $\TeX$ ' search path.

```

16.114 \loadlocalcfg{germanb}

```

Our last action is to make a note that the commands we have just defined, will be executed by calling the macro `\selectlanguage` at the beginning of the document.

```
16.115 \main@language{german}
```

Finally, the category code of `@` is reset to its original value.

```
16.116 \catcode'\@=\atcatcode
```

```
16.117 \code
```

## 17 The Breton language

The file `breton.dtx`<sup>15</sup> 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.

As this file needs to be read only once, we check whether it was read before. If it was, the `\captionbreton` is already defined, so we can stop processing. If this command is undefined we proceed with the various definitions and first show the current version of this file.

```
17.1 <*code>
17.2 \ifx\undefined\captionbreton
17.3 \else
17.4   \selectlanguage{breton}
17.5   \expandafter\endinput
17.6 \fi
```

`\atcatcode` This file, `breton.ldf`, may have been read while  $\TeX$  is in the middle of processing a document, so we have to make sure the category code of `@` is 'letter' while this file is being read. We save the category code of the `@`-sign in `\atcatcode` and make it 'letter'. Later the category code can be restored to whatever it was before.

```
17.7 \chardef\atcatcode=\catcode'\@
17.8 \catcode'\@=11\relax
```

Now we determine whether the the common macros from the file `babel.def` need to be read. We can be in one of two situations: either another language option has been read earlier on, in which case that other option has already read `babel.def`, or `breton` is the first language option to be processed. In that case we need to read `babel.def` right here before we continue.

```
17.9 \ifx\undefined\babel@core@loaded\input babel.def\relax\fi
```

Another check that has to be made, is if another language definition file has been read already. In that case its definitions have been activated. This might interfere with definitions this file tries to make. Therefore we make sure that we cancel any special definitions. This can be done by checking the existence of the macro `\originalTeX`. If it exists we simply execute it, otherwise it is `\let` to `\empty`.

```
17.10 \ifx\undefined\originalTeX \let\originalTeX\empty \fi
17.11 \originalTeX
```

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.

```
17.12 \ifx\undefined\l@breton
17.13   \nopatterns{Breton}
17.14   \addialect\l@breton0\fi
```

---

<sup>15</sup>The file described in this section has version number ? and was last revised on ?.

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 L<sup>A</sup>T<sub>E</sub>X.

```

17.15 \addto\captionsbreton{%
17.16   \def\prefacename{Rakskrid}%
17.17   \def\refname{Daveenno\'u}%
17.18   \def\abstractname{Dvierra\~n}%
17.19   \def\bibName{Lennadurezh}%
17.20   \def\chaptername{Pennad}%
17.21   \def\appendixname{Stagadenn}%
17.22   \def\contentsname{Taolenn}%
17.23   \def\listfigurename{Listenn ar Figurenno\'u}%
17.24   \def\listtablename{Listenn an taolenn\'u}%
17.25   \def\indexname{Meneger}%
17.26   \def\figurename{Figurenn}%
17.27   \def\tablename{Taolenn}%
17.28   \def\partname{Lodenn}%
17.29   \def\enclname{Diello\'u kevret}%
17.30   \def\ccname{Eilskrid da}%
17.31   \def\headtoname{evit}
17.32   \def\pagename{Pajenn}%
17.33   \def\seename{Gwelout}%
17.34   \def\alsoname{Gwelout ivez}%
17.35   \def\proofname{Proof}% <-- needs translation
17.36 }

```

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

```

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

```

`\extrasbreton` 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.

```

17.43 \initiate@active@char{:}
17.44 \initiate@active@char{;}
17.45 \initiate@active@char{!}
17.46 \initiate@active@char{?}

```

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

```

17.47 \addto\extrasbreton{\languageshorthands{breton}}

```

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

```

17.48 \addto\extrasbreton{%
17.49   \bbl@activate{:}\bbl@activate{;}%
17.50   \bbl@activate{!}\bbl@activate{?}}

```

```

17.51 %\addto\noextrasbreton{%
17.52 % \bbl@deactivate{:}\bbl@deactivate{;}%
17.53 % \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.

```

17.54 \addto\extrasbreton{\bbl@frenchspacing}
17.55 \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`.

```

17.56 \declare@shorthand{breton}{;}{%
17.57 \ifmmode
17.58 \string;\space
17.59 \else\relax

```

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

```

17.60 \ifhmode
17.61 \ifdim\lastskip>\z@
17.62 \unskip\penalty\M\thinspace
17.63 \fi
17.64 \fi
17.65 \string;\space
17.66 \fi}%

```

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

```

17.67 \declare@shorthand{breton}{:}{;%
17.68 \ifmmode\string;\space
17.69 \else\relax
17.70 \ifhmode
17.71 \ifdim\lastskip>\z@\unskip\penalty\M\thinspace\fi
17.72 \fi
17.73 \string;\space
17.74 \fi}
17.75 \declare@shorthand{breton}{!}{;%
17.76 \ifmmode\string!\space
17.77 \else\relax
17.78 \ifhmode
17.79 \ifdim\lastskip>\z@\unskip\penalty\M\thinspace\fi
17.80 \fi
17.81 \string!\space
17.82 \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.

```

17.83 \declare@shorthand{breton}{?}{;%
17.84 \ifmmode
17.85 \string?\space

```

```

17.86 \else\relax
17.87 \ifhmode
17.88 \ifdim\lastskip>\z@
17.89 \unskip
17.90 \kern\fontdimen2\font
17.91 \kern-1.4\fontdimen3\font
17.92 \fi
17.93 \fi
17.94 \string?\space
17.95 \fi}

```

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

```

17.96 \DeclareTextSymbol{\at}{OT1}{64}
17.97 \DeclareTextSymbol{\at}{T1}{64}
17.98 \DeclareTextSymbolDefault{\at}{OT1}
17.99 \DeclareTextSymbol{\boi}{OT1}{92}
17.100 \DeclareTextSymbol{\boi}{T1}{16}
17.101 \DeclareTextSymbolDefault{\boi}{OT1}
17.102 \DeclareTextSymbol{\circonflexe}{OT1}{94}
17.103 \DeclareTextSymbol{\circonflexe}{T1}{2}
17.104 \DeclareTextSymbolDefault{\circonflexe}{OT1}
17.105 \DeclareTextSymbol{\tild}{OT1}{126}
17.106 \DeclareTextSymbol{\tild}{T1}{3}
17.107 \DeclareTextSymbolDefault{\tild}{OT1}
17.108 \DeclareTextSymbol{\degre}{OT1}{23}
17.109 \DeclareTextSymbol{\degre}{T1}{6}
17.110 \DeclareTextSymbolDefault{\degre}{OT1}

```

The following macros are used in the redefinition of  $\wedge$  and  $\backslash$  to handle the letter i.

```

17.111 \AtBeginDocument{%
17.112 \DeclareTextCompositeCommand{\wedge}{OT1}{i}{\wedge i}
17.113 \DeclareTextCompositeCommand{\backslash}{OT1}{i}{\backslash i}}

```

And some more macros for numbering.

```

17.114 \def\kentan{1\$/\${}~{\rm a\tilde{n}}\}$}
17.115 \def\eil{2\$/\${}~{\rm l}\}$}
17.116 \def\re{\$/\${}~{\rm re}\}$}
17.117 \def\tredef{3\re}
17.118 \def\pevare{4\re}
17.119 \def\vet{\$/\${}~{\rm vet}\}$}
17.120 \def\pempvet{5\vet}

```

It is possible that a site might need to add some extra code to the babel macros.

To enable this we load a local configuration file, `breton.cfg` if it is found on  $\TeX$ ' search path.

```

17.121 \loadlocalcfg{breton}

```

Our last action is to make a note that the commands we have just defined, will be executed by calling the macro `\selectlanguage` at the beginning of the document.

```

17.122 \main@language{breton}

```

Finally, the category code of @ is reset to its original value. The macrospace used by `\atcatcode` is freed.

```
17.123 \catcode'\@=\atcatcode \let\atcatcode\relax
17.124 </code>
```

## 18 The Irish language

The file `irish.dtx`<sup>16</sup> defines all the language definition macros for the Irish language.

As this file needs to be read only once, we check whether it was read before. If it was, the command `\captionisirish` is already defined, so we can stop processing. If this command is undefined we proceed with the various definitions and first show the current version of this file.

```
18.1 <*code>
18.2 \ifx\undefined\captionisirish
18.3 \else
18.4   \selectlanguage{irish}
18.5   \expandafter\endinput
18.6 \fi
```

`\atcatcode` This file, `irish.ldf`, may have been read while  $\TeX$  is in the middle of processing a document, so we have to make sure the category code of `@` is ‘letter’ while this file is being read. We save the category code of the `@`-sign in `\atcatcode` and make it ‘letter’. Later the category code can be restored to whatever it was before.

```
18.7 \chardef\atcatcode=\catcode‘\@
18.8 \catcode‘\@=11\relax
```

Now we determine whether the the common macros from the file `babel.def` need to be read. We can be in one of two situations: either another language option has been read earlier on, in which case that other option has already read `babel.def`, or `irish` is the first language option to be processed. In that case we need to read `babel.def` right here before we continue.

```
18.9 \ifx\undefined\babel@core@loaded\input babel.def\relax\fi
```

Another check that has to be made, is if another language definition file has been read already. In that case its definitions have been activated. This might interfere with definitions this file tries to make. Therefore we make sure that we cancel any special definitions. This can be done by checking the existence of the macro `\originalTeX`. If it exists we simply execute it.

```
18.10 \ifx\undefined\originalTeX
18.11   \let\originalTeX\empty
18.12 \fi
18.13 \originalTeX
```

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.

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

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

---

<sup>16</sup>The file described in this section has version number ? and was last revised on ?. A contribution was made by Marion Gunn.

`\captionsirish` The macro `\captionsirish` defines all strings used in the four standard documentclasses provided with L<sup>A</sup>T<sub>E</sub>X.

```

18.17 \addto\captionsirish{%
18.18 \def\prefacename{Preface}% <-- needs translation
18.19 \def\refname{Tagairt'\{i}}%
18.20 \def\abstractname{Achoimre}%
18.21 \def\bibName{Leabharliosta}%
18.22 \def\chaptername{Caibidil}%
18.23 \def\appendixname{Aguis'\{i}n}%
18.24 \def\contentsname{Cl'ar 'Abhair}%
18.25 \def\listfigurename{L'ear'aid'\{i}}%
18.26 \def\listtablename{T'abla'\{i}}%
18.27 \def\indexname{Inn'eacs}%
18.28 \def\figurename{L'ear'aid}%
18.29 \def\tablename{T'abla}%
18.30 \def\partname{Cuid}%
18.31 \def\enclname{faoi iamh}%
18.32 \def\ccname{cc}% abrv. 'c'oip chuig'
18.33 \def\headtoname{Go}%
18.34 \def\pagename{Leathanach}%
18.35 \def\seename{see}% <-- needs translation
18.36 \def\alsaname{see also}% <-- needs translation
18.37 \def\proofname{Proof}% <-- needs translation
18.38 }
```

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

```

18.39 \def\dateirish{%
18.40 \number\day\space \ifcase\month\or
18.41 Ean'air\or Feabhra\or M'arta\or Aibre'an\or
18.42 Bealtaine\or Meitheamh\or I'uil\or L'unasa\or
18.43 Me'an F'omhair\or Deireadh F'omhair\or
18.44 M'\{i} na Samhna\or M'\{i} na Nollag\fi
18.45 \space \number\year}
```

`\extrasirish` 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.

```

18.46 \addto\extrasirish{}
18.47 \addto\noextrasirish{}
```

It is possible that a site might need to add some extra code to the babel macros. To enable this we load a local configuration file, `danish.cfg` if it is found on T<sub>E</sub>X' search path.

```
18.48 \loadlocalcfg{danish}
```

Our last action is to make a note that the commands we have just defined, will be executed by calling the macro `\selectlanguage` at the beginning of the document.

```
18.49 \main@language{irish}
```

Finally, the category code of @ is reset to its original value. The macrospace used by `\atcatcode` is freed.

```
18.50 \catcode'\@=\atcatcode \let\atcatcode\relax
18.51 \code
```

## 19 The scottish language

The file `scottish.dtx`<sup>17</sup> defines all the language definition macros for the scottish language.

As this file needs to be read only once, we check whether it was read before. If it was, the command `\captionsscottish` is already defined, so we can stop processing. If this command is undefined we proceed with the various definitions and first show the current version of this file.

```
19.1 <*code>
19.2 \ifx\undefined\captionsscottish
19.3 \else
19.4   \selectlanguage{scottish}
19.5   \expandafter\endinput
19.6 \fi
```

`\atcatcode` This file, `scottish.ldf`, may have been read while  $\TeX$  is in the middle of processing a document, so we have to make sure the category code of `@` is ‘letter’ while this file is being read. We save the category code of the `@`-sign in `\atcatcode` and make it ‘letter’. Later the category code can be restored to whatever it was before.

```
19.7 \chardef\atcatcode=\catcode‘\@
19.8 \catcode‘\@=11\relax
```

Now we determine whether the the common macros from the file `babel.def` need to be read. We can be in one of two situations: either another language option has been read earlier on, in which case that other option has already read `babel.def`, or `scottish` is the first language option to be processed. In that case we need to read `babel.def` right here before we continue.

```
19.9 \ifx\undefined\babel@core@loaded\input babel.def\relax\fi
```

Another check that has to be made, is if another language definition file has been read already. In that case its definitions have been activated. This might interfere with definitions this file tries to make. Therefore we make sure that we cancel any special definitions. This can be done by checking the existence of the macro `\originalTeX`. If it exists we simply execute it.

```
19.10 \ifx\undefined\originalTeX
19.11   \let\originalTeX\empty
19.12 \fi
19.13 \originalTeX
```

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.

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

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

---

<sup>17</sup>The file described in this section has version number ? and was last revised on ?. A contribution was made by Fraser Grant (`FRASER@CERNVM`).

`\captionsscottish` The macro `\captionsscottish` defines all strings used in the four standard documentclasses provided with L<sup>A</sup>T<sub>E</sub>X.

```
19.17 \addto\captionsscottish{%
19.18   \def\prefacename{Preface}%      <-- needs translation
19.19   \def\refname{Iomraidh}%
19.20   \def\abstractname{Br'{'\i}gh}%
19.21   \def\bibName{Leabhraichean}%
19.22   \def\chaptername{Caibideil}%
19.23   \def\appendixname{Ath-sgr'{'\i}obhadh}%
19.24   \def\contentsname{Cl'\ar-obraich}%
19.25   \def\listfigurename{Liosta Dhealbh }%
19.26   \def\listtablename{Liosta Chl'\ar}%
19.27   \def\indexname{Cl'\ar-innse}%
19.28   \def\figurename{Dealbh}%
19.29   \def\tablename{Cl'\ar}%
19.30   \def\partname{Cuid}%
19.31   \def\enclname{a-staigh}%
19.32   \def\ccname{lethbhreac gu}%
19.33   \def\headtoname{gu}%
19.34   \def\pagename{t.d.}%          abrv. 'taobh duilleag'
19.35   \def\seename{see}%           <-- needs translation
19.36   \def\alsaname{see also}%     <-- needs translation
19.37   \def\proofname{Proof}%      <-- needs translation
19.38 }
```

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

```
19.39 \def\datescottish{%
19.40   \number\day\space \ifcase\month\or
19.41   am Faoilteach\or an Gearran\or am M'\art\or an Giblean\or
19.42   an C'\eitean\or an t-\`Og mhios\or an t-Iuchar\or
19.43   L'\unasdal\or an Sultuine\or an D'\amhar\or
19.44   an t-Samhainn\or an Dubhlachd\fi
19.45   \space \number\year}}
```

`\extrasscottish` 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.

```
19.46 \addto\extrasscottish{}
19.47 \addto\noextrasscottish{}
```

It is possible that a site might need to add some extra code to the babel macros. To enable this we load a local configuration file, `scottish.cfg` if it is found on T<sub>E</sub>X' search path.

```
19.48 \loadlocalcfg{scottish}
```

Our last action is to make a note that the commands we have just defined, will be executed by calling the macro `\selectlanguage` at the beginning of the document.

```
19.49 \main@language{scottish}
```

Finally, the category code of @ is reset to its original value. The macrospace used by `\atcatcode` is freed.

```
19.50 \catcode'\@=\atcatcode \let\atcatcode\relax
19.51 </code>
```

## 20 The French language

The file `francais.dtx`<sup>18</sup> defines all the language definition macros for the French language.

French typographic rules specify that a little white space should be present before ‘double punctuation’ characters. These characters are `:`, `;`, `!` and `?`. In order to get this whitespace automatically the category code of these characters is made `\active`. The user should input these four characters preceded with a space; the space will then be replaced by a `\thinspace`.

As this file needs to be read only once, we check whether it was read before. If it was, the `\captionsfrancais` is already defined, so we can stop processing. If this command is undefined we proceed with the various definitions and first show the current version of this file.

```
20.1 (*code)
20.2 \ifx\undefined\captionsfrancais
20.3 \else
20.4   \selectlanguage{francais}
20.5   \expandafter\endinput
20.6 \fi
```

`\atcatcode` This file, `francais.ldf`, may have been read while  $\TeX$  is in the middle of processing a document, so we have to make sure the category code of `@` is ‘letter’ while this file is being read. We save the category code of the `@`-sign in `\atcatcode` and make it ‘letter’. Later the category code can be restored to whatever it was before.

```
20.7 \chardef\atcatcode=\catcode'\@
20.8 \catcode'\@=11\relax
```

Now we determine whether the the common macros from the file `babel.def` need to be read. We can be in one of two situations: either another language option has been read earlier on, in which case that other option has already read `babel.def`, or `francais` is the first language option to be processed. In that case we need to read `babel.def` right here before we continue.

```
20.9 \ifx\undefined\babel@core@loaded\input babel.def\relax\fi
```

Another check that has to be made, is if another language definition file has been read already. In that case its definitions have been activated. This might interfere with definitions this file tries to make. Therefore we make sure that we cancel any special definitions. This can be done by checking the existence of the macro `\originalTeX`. If it exists we simply execute it, otherwise it is `\let` to `\empty`.

```
20.10 \ifx\undefined\originalTeX \let\originalTeX\empty \fi
20.11 \originalTeX
```

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

---

<sup>18</sup>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 French. Later the definitions from `french.sty` version 2 were added.

```

20.12 \ifx\l@français\undefined
20.13 \ifx\l@french\undefined
20.14   \nopatterns{Français}
20.15   \adddialect\l@français0
20.16   \let\l@french\l@français
20.17 \else
20.18   \let\l@français\l@french
20.19 \fi
20.20 \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.

`\captionsfrançais` The macro `\captionsfrançais` defines all strings used in the four standard document classes provided with L<sup>A</sup>T<sub>E</sub>X.

```

20.21 \addto\captionsfrançais{%
20.22   \def\prefacename{Préface}%
20.23   \def\refname{Références}%
20.24   \def\abstractname{Résumé}%
20.25   \def\bibname{Bibliographie}%
20.26   \def\chaptername{Chapitre}%
20.27   \def\appendixname{Annexe}%
20.28   \def\contentsname{Table des matières}%
20.29   \def\listfigurename{Liste des figures}%
20.30   \def\listtablename{Liste des tableaux}%
20.31   \def\indexname{Index}%
20.32   \def\figurename{Figure}%
20.33   \def\tablename{Tableau}%
20.34   \def\partname{Partie}%
20.35   \def\enclname{P.~J.}%
20.36   \def\ccname{Copie \a}%
20.37   \def\headtoname{A}
20.38   \def\pagename{Page}%
20.39   \def\seename{voir}%
20.40   \def\alsoname{voir aussi}%
20.41   \def\proofname{Proof}% <-- needs translation!
20.42 }

```

`\datefrançais` The macro `\datefrançais` redefines the command `\today` to produce French dates.

```

20.43 \def\datefrançais{%
20.44 \def\today{\ifnum\day=1\relax 1/\${\rm er}$\else
20.45   \number\day\fi \space\ifcase\month\or
20.46   janvier\or f'évrier\or mars\or avril\or mai\or juin\or
20.47   juillet\or août\or septembre\or octobre\or novembre\or
20.48   d'ecembre\fi
20.49   \space\number\year}}

```

`\extrasfrançais` The macro `\extrasfrançais` will perform all the extra definitions needed for the French language. The macro `\noextrasfrançais` is used to cancel the actions of `\extrasfrançais`.

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

```

20.50 \initiate@active@char{:}
20.51 \initiate@active@char{;}
20.52 \initiate@active@char{!}
20.53 \initiate@active@char{?}

```

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

```
20.54 \addto\extrasfrancais{\languageshorthands{french}}
```

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

```

20.55 \addto\extrasfrancais{%
20.56   \bbl@activate{:}\bbl@activate{;}%
20.57   \bbl@activate{!}\bbl@activate{?}}
20.58 %\addto\noextrasfrancais{%
20.59 %  \bbl@deactivate{:}\bbl@deactivate{;}%
20.60 %  \bbl@deactivate{!}\bbl@deactivate{?}}

```

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

```

20.61 \addto\extrasfrancais{\bbl@frenchspacing}
20.62 \addto\noextrasfrancais{\bbl@nonfrenchspacing}

```

`\french@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`.

```

20.63 \declare@shorthand{french}{;}{;%
20.64   \ifmmode
20.65     \string;\space
20.66   \else\relax

```

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

```

20.67   \ifhmode
20.68     \ifdim\lastskip>\z@
20.69       \unskip\penalty\@M\thinspace
20.70     \fi
20.71   \fi

```

Now we can insert a `;` character.

```

20.72     \string;\space
20.73   \fi}

```

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

```

20.74 \declare@shorthand{french}{:}{;%
20.75   \ifmmode\string;\space
20.76   \else\relax
20.77   \ifhmode
20.78     \ifdim\lastskip>\z@\unskip\penalty\@M\thinspace\fi
20.79   \fi
20.80   \string;\space
20.81   \fi}
20.82 \declare@shorthand{french}{!}{;%
20.83   \ifmmode\string!\space
20.84   \else\relax

```

```

20.85 \ifhmode
20.86 \ifdim\lastskip>\z@\unskip\penalty\@M\thinspace\fi
20.87 \fi
20.88 \string!\space
20.89 \fi}

```

`\french@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.

```

20.90 \declare@shorthand{french}{?}{%
20.91 \ifmmode\string?\space
20.92 \else\relax
20.93 \ifhmode
20.94 \ifdim\lastskip>\z@
20.95 \unskip
20.96 \kern\fontdimen2\font
20.97 \kern-1.4\fontdimen3\font
20.98 \fi
20.99 \fi
20.100 \string?\space
20.101 \fi}

```

`\system@sh@:@` When the active characters appear in an environment where their french behaviour  
`\system@sh@!@` is not wanted they should give an ‘expected’ result, ie not gobble up the space  
`\system@sh@?@` that follows them. Therefore we define shorthands at system level as well.

```

\system@sh@;@0.102 \declare@shorthand{system}{:}{\string:\space}
20.103 \declare@shorthand{system}{!}{\string!\space}
20.104 \declare@shorthand{system}{?}{\string?\space}
20.105 \declare@shorthand{system}{;}{\string;\space}

```

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

```

20.106 \DeclareTextSymbol{\at}{OT1}{64}
20.107 \DeclareTextSymbol{\at}{T1}{64}
20.108 \DeclareTextSymbolDefault{\at}{OT1}
20.109 \DeclareTextSymbol{\boi}{OT1}{92}
20.110 \DeclareTextSymbol{\boi}{T1}{16}
20.111 \DeclareTextSymbolDefault{\boi}{OT1}
20.112 \DeclareTextSymbol{\circonflexe}{OT1}{94}
20.113 \DeclareTextSymbol{\circonflexe}{T1}{2}
20.114 \DeclareTextSymbolDefault{\circonflexe}{OT1}
20.115 \DeclareTextSymbol{\tild}{OT1}{126}
20.116 \DeclareTextSymbol{\tild}{T1}{3}
20.117 \DeclareTextSymbolDefault{\tild}{OT1}
20.118 \DeclareTextSymbol{\degre}{OT1}{23}
20.119 \DeclareTextSymbol{\degre}{T1}{6}
20.120 \DeclareTextSymbolDefault{\degre}{OT1}

```

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

```

20.121 \AtBeginDocument{%
20.122 \DeclareTextCompositeCommand{\^}{OT1}{i}{\^i}
20.123 \DeclareTextCompositeCommand{\"}{OT1}{i}{\"}i}

```

A macro for typesetting things like 1<sup>er</sup> as proposed by Raymon Seroul.

```
20.124 \def\up#1{\raise 1ex\hbox{\small#1}}
```

Definitions as provided by Nicolas Brouard for typing \No3 to get 3<sup>o</sup> and for typing 4\ieme to get 4<sup>e</sup>.

```
20.125 \def\No{\kern-.25em\lower.2ex\hbox{\degre}}
```

```
20.126 \def\ieme{$^\{\rm e }\kern+.17em}
```

And some more macros for numbering. First two support macros.

```
20.127 \def\FrenchEnumerate#1{#1^\{\rm o}\kern+.29em}
```

```
20.128 \def\FrenchPopularEnumerate#1{#1\No\kern-.25em)\kern+.3em}
```

Typing \primo should result in ‘1<sup>o</sup>’,

```
20.129 \def\primo{\FrenchEnumerate1}
```

```
20.130 \def\secundo{\FrenchEnumerate2}
```

```
20.131 \def\tertio{\FrenchEnumerate3}
```

```
20.132 \def\quatro{\FrenchEnumerate4}
```

while typing \fprimo) gives ‘1<sup>o</sup>’.

```
20.133 \def\fprimo{\FrenchPopularEnumerate1}
```

```
20.134 \def\fsecundo{\FrenchPopularEnumerate2}
```

```
20.135 \def\ftertio{\FrenchPopularEnumerate3}
```

```
20.136 \def\fquatro{\FrenchPopularEnumerate4}
```

It is possible that a site might need to add some extra code to the babel macros.

To enable this we load a local configuration file, `francais.cfg` if it is found on `TEX`’ search path.

```
20.137 \loadlocalcfg{francais}
```

Our last action is to make a note that the commands we have just defined, will be executed by calling the macro `\selectlanguage` at the beginning of the document.

```
20.138 \main@language{francais}
```

Finally, the category code of `@` is reset to its original value. The macrospace used by `\atcatcode` is freed.

```
20.139 \catcode'\@=\atcatcode \let\atcatcode\relax
```

```
20.140 </code>
```

## 21 The Italian language

The file `italian.dtx`<sup>19</sup> 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.

As this file needs to be read only once, we check whether it was read before. If it was, the command `\captionitalian` is already defined, so we can stop processing. If this command is undefined we proceed with the various definitions and first show the current version of this file.

```
21.1 (*code)
21.2 \ifx\undefined\captionitalian
21.3 \else
21.4   \selectlanguage{italian}
21.5   \expandafter\endinput
21.6 \fi
```

`\atcatcode` This file, `italian.sty`, may have been read while TeX is in the middle of processing a document, so we have to make sure the category code of `@` is ‘letter’ while this file is being read. We save the category code of the `@`-sign in `\atcatcode` and make it ‘letter’. Later the category code can be restored to whatever it was before.

```
21.7 \chardef\atcatcode=\catcode‘\@
21.8 \catcode‘\@=11\relax
```

Now we determine whether the the common macros from the file `babel.def` need to be read. We can be in one of two situations: either another language option has been read earlier on, in which case that other option has already read `babel.def`, or `italian` is the first language option to be processed. In that case we need to read `babel.def` right here before we continue.

```
21.9 \ifx\undefined\babel@core@loaded\input babel.def\relax\fi
```

Another check that has to be made, is if another language definition file has been read already. In that case its definitions have been activated. This might interfere with definitions this file tries to make. Therefore we make sure that we cancel any special definitions. This can be done by checking the existence of the macro `\originalTeX`. If it exists we simply execute it, otherwise it is `\let` to `\empty`.

```
21.10 \ifx\undefined\originalTeX \let\originalTeX\empty \fi
21.11 \originalTeX
```

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.

```
21.12 \ifx\undefined\l@italian
21.13   \@nopatterns{Italian}
21.14   \adddialect\l@italian0\fi
```

---

<sup>19</sup>The file described in this section has version number ? and was last revised on ?. The original author is Maurizio Codogno, (`urcm@ur785.csel.tstet.it`).

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 L<sup>A</sup>T<sub>E</sub>X.

```
21.15 \addto\captionsitalian{%
21.16   \def\prefacename{Prefazione}%
21.17   \def\refname{Riferimenti bibliografici}%
21.18   \def\abstractname{Sommario}%
21.19   \def\bibName{Bibliografia}%
21.20   \def\chaptername{Capitolo}%
21.21   \def\appendixname{Appendice}%
21.22   \def\contentsname{Indice}%
21.23   \def\listfigurename{Elenco delle figure}%
21.24   \def\listtablename{Elenco delle tabelle}%
21.25   \def\indexname{Indice analitico}%
21.26   \def\figurename{Figura}%
21.27   \def\tablename{Tabella}%
21.28   \def\partname{Parte}%
21.29   \def\enclname{Allegati}%
21.30   \def\ccname{e~p.~c.}%
21.31   \def\headtoname{Per}%
21.32   \def\pagename{Pag.}%    % in Italian abbreviation is preferred
21.33   \def\seename{vedi}%
21.34   \def\alsoname{vedi anche}%
21.35   \def\proofname{Proof}% <-- needs translation
21.36 }
```

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

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

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

```
21.42 \def\italianhyphenmins{\tw@\tw@}
```

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

```
\noextrasitalian21.43 \addto\extrasitalian{%
21.44   \babel@savevariable\clubpenalty
21.45   \babel@savevariable\widowpenalty
21.46   \clubpenalty3000\widowpenalty3000}
```

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

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

It is possible that a site might need to add some extra code to the babel macros. To enable this we load a local configuration file, `italian.cfg` if it is found on T<sub>E</sub>X' search path.

```
21.50 \loadlocalcfg{italian}
```

Our last action is to make a note that the commands we have just defined, will be executed by calling the macro `\selectlanguage` at the beginning of the document.

```
21.51 \main@language{italian}
```

Finally, the category code of `@` is reset to its original value. The macrospace used by `\atcatcode` is freed.

```
21.52 \catcode'\@=\atcatcode \let\atcatcode\relax
```

```
21.53 </code>
```

## 22 The Portuguese language

The file `portuges.dtx`<sup>20</sup> defines all the language-specific macros for the Portuguese language as well as for the Brazilian version of this language.

For this language the character `"` is made active. In table 6 an overview is given of its purpose.

<code>" </code>	disable ligature at this position.
<code>"-</code>	an explicit hyphen sign, allowing hyphenation in the rest of the word.
<code>""</code>	like <code>"-</code> , but producing no hyphen sign (for words that should break at some sign such as “entrada/salida.”)
<code>"&lt;</code>	for French left double quotes (similar to <code>&lt;&lt;</code> ).
<code>"&gt;</code>	for French right double quotes (similar to <code>&gt;&gt;</code> ).
<code>\-</code>	like the old <code>\-</code> , but allowing hyphenation in the rest of the word.

Table 6: The extra definitions made by `portuges.ldf`

As this file needs to be read only once, we check whether it was read before. If it was, the command `\captionoportuges` is already defined, so we can stop processing. If this command is undefined we proceed with the various definitions and first show the current version of this file.

```
22.1 {*code}
22.2 \ifx\undefined\captionoportuges
22.3 \else
22.4 \selectlanguage{portuges}
22.5 \expandafter\endinput
22.6 \fi
```

`\atcatcode` This file, `portuges.ldf`, may have been read while  $\TeX$  is in the middle of processing a document, so we have to make sure the category code of `@` is ‘letter’ while this file is being read. We save the category code of the `@`-sign in `\atcatcode` and make it ‘letter’. Later the category code can be restored to whatever it was before.

```
22.7 \chardef\atcatcode=\catcode'\@
22.8 \catcode'\@=11\relax
```

Now we determine whether the the common macros from the file `babel.def` need to be read. We can be in one of two situations: either another language option has been read earlier on, in which case that other option has already read `babel.def`, or `portuges` is the first language option to be processed. In that case we need to read `babel.def` right here before we continue.

```
22.9 \ifx\undefined\babel@core@loaded\input babel.def\relax\fi
```

Another check that has to be made, is if another language definition file has been read already. In that case its definitions have been activated. This might

---

<sup>20</sup>The file described in this section has version number `?` and was last revised on `?`. Contributions were made by Jose Pedro Ramalhete (`JRAMALHE@CERNVM` or `Jose-Pedro_Ramalhete@MACMAIL`) and Arnaldo Viegas de Lima (`arnaldo@VNET.IBM.COM`).

interfere with definitions this file tries to make. Therefore we make sure that we cancel any special definitions. This can be done by checking the existence of the macro `\originalTeX`. If it exists we simply execute it, otherwise it is `\let` to `\empty`.

```
22.10 \ifx\undefined\originalTeX \let\originalTeX\empty \else\originalTeX\fi
```

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.

```
22.11 \ifx\undefined\l@portuges
```

```
22.12   \nopatterns{Portuges}
```

```
22.13   \adddialect\l@portuges0\fi
```

For the Brazilian version of these definitions we just add a “dialect”. Also, the macros `\captionsbrazil` and `\extrasbrazil` are `\let` to their Portuguese counterparts when these parts are defined.

```
22.14 \adddialect\l@brazil\l@portuges
```

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 L<sup>A</sup>T<sub>E</sub>X.

```
22.15 \addto\captionsportuges{%
```

```
22.16   \def\prefacename{Pref\’acio}%
```

```
22.17   \def\refname{Refer\^encias}%
```

```
22.18   \def\abstractname{Resumo}%
```

```
22.19   \def\bibName{Bibliografia}%
```

```
22.20   \def\chaptername{Cap\’{\i}tulo}%
```

```
22.21   \def\appendixname{Ap\^endice}%
```

```
22.22   \def\contentsname{\’Indice}%
```

```
22.23   \def\listfigurename{Lista de Figuras}%
```

```
22.24   \def\listtablename{Lista de Tabelas}%
```

```
22.25   \def\indexname{\’Indice Remissivo}%
```

```
22.26   \def\figurename{Figura}%
```

```
22.27   \def\tablename{Tabela}%
```

```
22.28   \def\partname{Parte}%
```

```
22.29   \def\enclname{Anexos}%
```

```
22.30   \def\ccname{C\’opia a}%
```

```
22.31   \def\headtoname{Para}%
```

```
22.32   \def\pagename{P\’agina}%
```

```
22.33   \def\seename{ver}%
```

```
22.34   \def\alsoname{ver tamb\’em}%
```

```
22.35   \def\proofname{Proof}% <-- needs translation
```

```
22.36 }%
```

`\captionsbrazil` The “captions” are different for both versions of the language, so we define the macro `\captionsbrazil` here.

```
22.37 \addto\captionsbrazil{%
```

```
22.38   \def\prefacename{Pref\’acio}%
```

```
22.39   \def\refname{Refer\^encias}%
```

```
22.40   \def\abstractname{Resumo}%
```

```
22.41   \def\bibName{Refer\^encias Bibliogr\’aficas}%
```

```

22.42 \def\chaptername{Cap\'}{\i}tulo}%
22.43 \def\appendixname{Ap\^endice}%
22.44 \def\contentsname{Sum\'}ario}%
22.45 \def\listfigurename{Lista de Figuras}%
22.46 \def\listtablename{Lista de Tabelas}%
22.47 \def\indexname{\'}Indice}%
22.48 \def\figurename{Figura}%
22.49 \def\tablename{Tabela}%
22.50 \def\partname{Parte}%
22.51 \def\enclname{Anexo}%
22.52 \def\ccname{C\'}opia para}%
22.53 \def\headtoname{Para}%
22.54 \def\pagename{P\'}agina}%
22.55 \def\seename{veja}%
22.56 \def\alsiname{veja tamb\'}em}%
22.57 }

```

`\dateportuges` The macro `\dateportuges` redefines the command `\today` to produce Portuguese dates.

```

22.58 \def\dateportuges{%
22.59 \def\today{\number\day\space de\space\ifcase\month\or
22.60 Janeiro\or Fevereiro\or Mar\c{c}o\or Abril\or Maio\or Junho\or
22.61 Julho\or Agosto\or Setembro\or Outubro\or Novembro\or Dezembro\fi
22.62 \space de\space\number\year}}

```

`\datebrazil` The macro `\datebrazil` redefines the command `\today` to produce Brazilian dates, for which the names of the months are not capitalized.

```

22.63 \def\datebrazil{%
22.64 \def\today{\number\day\space de\space\ifcase\month\or
22.65 janeiro\or fevereiro\or mar\c{c}o\or abril\or maio\or junho\or
22.66 julho\or agosto\or setembro\or outubro\or novembro\or dezembro\fi
22.67 \space de\space\number\year}}

```

`\portugeshyphenmins` Set correct values for `\lefthyphenmin` and `\righthyphenmin`.

```

\brasilhyphenmins22.68 \def\portugeshyphenmins{\tw@\tw@}
22.69 \def\brasilhyphenmins{\tw@\tw@}

```

`\extrasportuges` 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.

```

22.70 \initiate@active@char{"}
22.71 \addto\extrasportuges{\languageshorthands{portuges}}
22.72 \addto\extrasportuges{\bbl@activate{"}}
22.73 %\addto\noextrasportuges{\bbl@deactivate{"}}

```

First we define access to the guillemets for quotations,

```

22.74 \declare@shorthand{portuges}{"<"}{%
22.75 \textormath{\guillemotleft}}{\mbox{\guillemotleft}}
22.76 \declare@shorthand{portuges}{">"}{%
22.77 \textormath{\guillemotright}}{\mbox{\guillemotright}}

```

then we define two shorthands to be able to specify hyphenation breakpoints that behave a little different from `\-`.

```
22.78 \declare@shorthand{portuges}{-}{\allowhyphens-\allowhyphens}
22.79 \declare@shorthand{portuges}{"}{\hskip\z@skip}
```

And we want to have a shorthand for disabling a ligature.

```
22.80 \declare@shorthand{portuges}{"}{|}{%
22.81 \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.

```
22.82 \addto\extrasportuges{\babel@save\-\}
22.83 \addto\extrasportuges{\def\-\{-\allowhyphens
22.84 \discretionary{-}{-}{\allowhyphens}}
```

`\ord` We also provide an easy way to typeset ordinals, both in the male (`\ord` or `\ro`) `\ro` and the female (`orda` or `\ra`) form.

```
\orda22.85 \def\ord{${\rm o}$}
\ra22.86 \def\orda{${\rm a}$}
22.87 \let\ro\ord\let\ra\orda
```

`\extrasbrazil` Also for the “brazil” variant no extra definitions are needed at the moment.

```
\noextrasbrazil22.88 \let\extrasbrazil\extrasportuges
22.89 \let\noextrasbrazil\noextrasportuges
```

It is possible that a site might need to add some extra code to the `babel` macros. To enable this we load a local configuration file, `portuges.cfg` if it is found on  $\TeX$ ’ search path.

```
22.90 \loadlocalcfg{portuges}
```

Our last action is to make a note that the commands we have just defined, will be executed by calling the macro `\selectlanguage` at the beginning of the document.

```
22.91 \main@language{portuges}
```

Finally, the category code of `@` is reset to its original value. The macrospace used by `\atcatcode` is freed.

```
22.92 \catcode'\@=\atcatcode \let\atcatcode\relax
22.93 </code>
```

## 23 The Spanish language

The file `spanish.dtx`<sup>21</sup> defines all the language definition macro's for the Spanish<sup>22</sup> language.

This file<sup>23</sup> incorporates the result of discussions held in the Spanish- $\TeX$ <sup>24</sup> 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 <sup>a</sup> .
"o	masculine ordinal as in 1 <sup>o</sup> .
"<	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 existence 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  $\LaTeX 2_{\epsilon}$  or NFSS2. If T1 encoding is chosen, and provided that adequate hyphenation patterns<sup>25</sup> exist, it

---

<sup>21</sup>The file described in this section has version number ? and was last revised on ?. The original author is Julio Sánchez, ([jsanchez@gmv.es](mailto:jsanchez@gmv.es)).

<sup>22</sup>Catalan used to be part of this file but is now on its own file.

<sup>23</sup>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](mailto:jsanchez@gmv.es)) know. I humbly apologize for any omission.

<sup>24</sup>`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](mailto:jsanchez@gmv.es).

<sup>25</sup>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 L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> to load the package `t1enc` with `\usepackage`. This must be done before loading `babel`.

If the combination of keyboard and T<sub>E</sub>X version that the user has is able to produce the accented characters in the T1 encoding, 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.

As this file needs to be read only once, we check whether it was read before. If it was, the `\captionsspanish` is already defined, so we can stop processing. If this command is undefined we proceed with the various definitions and first show the current version of this file.

```
23.1 (*code)
23.2 \ifx\undefined\captionsspanish
23.3 \else
23.4 \selectlanguage{spanish}
23.5 \expandafter\endinput
23.6 \fi
```

`\atcatcode` This file, `spanish.ldf`, may have been read while T<sub>E</sub>X is in the middle of processing a document, so we have to make sure the category code of `@` is ‘letter’ while this file is being read. We save the category code of the `@`-sign in `\atcatcode` and make it ‘letter’. Later the category code can be restored to whatever it was before.

```
23.7 \chardef\atcatcode=\catcode'\@
23.8 \catcode'\@=11\relax
```

Now we determine whether the common macros from the file `babel.def` need to be read. We can be in one of two situations: either another language option has been read earlier on, in which case that other option has already read `babel.def`, or `spanish` is the first language option to be processed. In that case we need to read `babel.def` right here before we continue.

```
23.9 \ifx\undefined\babel@core@loaded\input babel.def\relax\fi
```

Another check that has to be made, is if another language definition file has been read already. In that case its definitions have been activated. This might interfere with definitions this file tries to make. Therefore we make sure that we cancel any special definitions. This can be done by checking the existence of the macro `\originalTeX`. If it exists we simply execute it, otherwise it is `\let` to `\empty`.

```
23.10 \ifx\undefined\originalTeX \let\originalTeX\empty \else\originalTeX\fi
```

---

tory `src/TeX/spanish`. The list of Frequently Asked Questions with Answers about T<sub>E</sub>X for Spanish is kept there as well. That list is meant to be a summary of the discussions held in the Spanish-T<sub>E</sub>X mail list. Warning: It is in Spanish.

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.

```
23.11 \ifx\undefined\l@spanish
23.12 \nopatterns{Spanish}
23.13 \adddialect\l@spanish0
23.14 \fi
```

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

`\captionsspanish` The macro `\captionsspanish` defines all strings<sup>26</sup> used in the four standard documentclasses provided with L<sup>A</sup>T<sub>E</sub>X.

```
23.15 \addto\captionsspanish{%
23.16 \def\prefacename{Prefacio}%
23.17 \def\refname{Referencias}%
23.18 \def\abstractname{Resumen}%
23.19 \def\bibName{Bibliograf\’\{i}a}%
23.20 \def\chaptername{Cap\’\{i}tulo}%
23.21 \def\appendixname{Ap\’endice}%
23.22 \def\contentsname{\’Indice General}%
23.23 \def\listfigurename{\’Indice de Figuras}%
23.24 \def\listtablename{\’Indice de Tablas}%
23.25 \def\indexname{\’Indice de Materias}%
23.26 \def\figurename{Figura}%
23.27 \def\tablename{Tabla}%
23.28 \def\partname{Parte}%
23.29 \def\enclname{Adjunto}%
23.30 \def\ccname{Copia a}%
23.31 \def\headtoname{A}%
23.32 \def\pagename{P\’agina}%
23.33 \def\seenname{v\’ease}%
23.34 \def\alsoname{v\’ease tambi\’en}%
23.35 \def\proofname{Proof}% <-- needs translation!
23.36 }%
```

`\datespanish` The macro `\datespanish` redefines the command `\today` to produce Spanish<sup>27</sup> dates.

```
23.37 \def\datespanish{%
23.38 \def\today{\number\day~de\space\ifcase\month\or
23.39 enero\or febrero\or marzo\or abril\or mayo\or junio\or
23.40 julio\or agosto\or septiembre\or octubre\or noviembre\or diciembre\fi
23.41 \space de~\number\year}}
```

`\extrasspanish` The macro `\extrasspanish` will perform all the extra definitions needed for the Spanish language. The macro `\noextrasspanish` is used to cancel the actions of

---

<sup>26</sup>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*.

<sup>27</sup>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.

`\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.

```

23.42 \addto\extrasspanish{\languageshorthands{spanish}}
23.43 \initiate@active@char{"}
23.44 \initiate@active@char{~}
23.45 \addto\extrasspanish{%
23.46   \bbl@activate{"}%
23.47   \bbl@activate{~}}
23.48 \@ifpackagewith{babel}{activeacute}{%
23.49   \initiate@active@char{'}}
23.50   \addto\extrasspanish{\bbl@activate{'}}{}
23.51 %\addto\noextrasspanish{
23.52 %   \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.

```

23.53 \addto\extrasspanish{%
23.54   \babel@save\"
23.55   \babel@save\~
23.56   \def\"{\protect\@umlaut}%
23.57   \def\~{\protect\@tilde}}
23.58 \@ifpackagewith{babel}{activeacute}{%
23.59   \babel@save\'
23.60   \addto\extrasspanish{\def\'{\protect\@acute}}
23.61   }{}

```

`\spanishhyphenmins` Spanish hyphenation uses `\lefthyphenmin` and `\righthyphenmin` both set to 2.

```

23.62 \def\spanishhyphenmins{\tw@\tw@}

```

`\dieresis` 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`.

```

23.63 \let\dieresis\"
23.64 \let\texttilde\~
23.65 \@ifpackagewith{babel}{activeacute}{\let\textacute\'}{}

```

`\@umlaut` 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 T<sub>E</sub>X at least finds the breaks that are not too close to the accent. The warning will only be printed to the log file.

```

23.66 \ifx\undefined\DeclareFontShape
23.67 \wlog{Warning: You are using an old LaTeX}
23.68 \wlog{Some word breaks will not be found.}
23.69 \def\@umlaut#1{\allowhyphens\dieresis{#1}\allowhyphens}
23.70 \def\@tilde#1{\allowhyphens\texttilde{#1}\allowhyphens}
23.71 \@ifpackagewith{babel}{activeacute}{%
23.72 \def\@acute#1{\allowhyphens\textacute{#1}\allowhyphens}}{}
23.73 \else
23.74 \edef\next{T1}
23.75 \ifx\f@encoding\next
23.76 \let\@umlaut\dieresis
23.77 \let\@tilde\texttilde
23.78 \@ifpackagewith{babel}{activeacute}{%
23.79 \let\@acute\textacute}{}
23.80 \else
23.81 \wlog{Warning: You are using encoding \f@encoding\space
23.82 instead of T1.}
23.83 \wlog{Some word breaks will not be found.}
23.84 \def\@umlaut#1{\allowhyphens\dieresis{#1}\allowhyphens}
23.85 \def\@tilde#1{\allowhyphens\texttilde{#1}\allowhyphens}
23.86 \@ifpackagewith{babel}{activeacute}{%
23.87 \def\@acute#1{\allowhyphens\textacute{#1}\allowhyphens}}{}
23.88 \fi
23.89 \fi

```

Now we can define our shorthands: the umlauts,

```

23.90 \declare@shorthand{spanish}{"u}{\@umlaut u}
23.91 \declare@shorthand{spanish}{"U}{\@umlaut U}

```

french quotes,

```

23.92 \declare@shorthand{spanish}{"<"}{%
23.93 \textormath{\guillemotleft}}{\mbox{\guillemotleft}}
23.94 \declare@shorthand{spanish}{">"}{%
23.95 \textormath{\guillemotright}}{\mbox{\guillemotright}}

```

ordinals<sup>28</sup>,

```

23.96 \declare@shorthand{spanish}{"o"}{%
23.97 \raise1ex\hbox{\underbar{\scriptsize o}}}
23.98 \declare@shorthand{spanish}{"a"}{%
23.99 \raise1ex\hbox{\underbar{\scriptsize a}}}

```

acute accents,

```

23.100 \@ifpackagewith{babel}{activeacute}{%
23.101 \declare@shorthand{spanish}{'a}{\textormath{\@acute a}{\prime} a}}
23.102 \declare@shorthand{spanish}{'e}{\textormath{\@acute e}{\prime} e}}
23.103 \declare@shorthand{spanish}{'i}{\textormath{\@acute i}{\prime} i}}
23.104 \declare@shorthand{spanish}{'o}{\textormath{\@acute o}{\prime} o}}
23.105 \declare@shorthand{spanish}{'u}{\textormath{\@acute u}{\prime} u}}
23.106 \declare@shorthand{spanish}{'A}{\textormath{\@acute A}{\prime} A}}
23.107 \declare@shorthand{spanish}{'E}{\textormath{\@acute E}{\prime} E}}

```

<sup>28</sup>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.

```

23.108 \declare@shorthand{spanish}{'I}{\textormath{\@acute I}{^{\prime} I}}
23.109 \declare@shorthand{spanish}{'O}{\textormath{\@acute O}{^{\prime} O}}
23.110 \declare@shorthand{spanish}{'U}{\textormath{\@acute U}{^{\prime} U}}

```

the acute accent,

```

23.111 \declare@shorthand{spanish}{' }{' }{%
23.112   \textormath{\textquotedblright}{\sp\bggroup\prim@s' }}

```

tildes,

```

23.113 \declare@shorthand{spanish}{'n}{\textormath{\~n}{^{\prime} n}}
23.114 \declare@shorthand{spanish}{'N}{\textormath{\~N}{^{\prime} N}}
23.115 }{}
23.116 \declare@shorthand{spanish}{~n}{\textormath{\~n}{\@tilde n}}
23.117 \declare@shorthand{spanish}{~N}{\textormath{\~N}{\@tilde N}}

```

and some additional commands:

```

23.118 \declare@shorthand{spanish}{"-}{\allowhyphens\-\allowhyphens}
23.119 \declare@shorthand{spanish}{"|}{%
23.120   \textormath{\penalty\@M\discretionary{-}{-}{\kern.03em}%
23.121     \allowhyphens}{-}}
23.122 \declare@shorthand{spanish}{""}{\hskip\z@skip}

```

It is possible that a site might need to add some extra code to the babel macros. To enable this we load a local configuration file, `spanish.cfg` if it is found on T<sub>E</sub>X' search path.

```

23.123 \loadlocalcfg{spanish}

```

Next the babel macro `\main@language` is used to activate the definitions for Spanish at the beginning of the document.

```

23.124 \main@language{spanish}

```

Finally, the category code of `@` is reset to its original value. The macrospace used by `\atcatcode` is freed.

```

23.125 \catcode'\@=\atcatcode \let\atcatcode\relax
23.126 </code>

```

## 24 The Catalan language

The file `catalan.dtx`<sup>29</sup> 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 become also available<sup>30</sup>.

<code>\lgem</code>	geminated-l digraph (similar to ll). <code>\Lgem</code> produces the uppercase version.
<code>\up</code>	Macro to help typing raised ordinals, like 1 <sup>er</sup> . Takes one argument.
<code>\-</code>	like the old <code>\-</code> , but allowing hyphenation in the rest of the word.
<code>"i</code>	i with diaeresis, allowing hyphenation in the rest of the word. Valid for the following vowels: i, u (both lowercase and uppercase).
<code>"c</code>	c-cedilla (ç). Valid for both uppercase and lowercase c.
<code>"l</code>	geminated-l digraph (similar to ll). Valid for both uppercase and lowercase l.
<code>"&lt;</code>	French left double quotes (similar to <<).
<code>"&gt;</code>	French right double quotes (similar to >>).
<code>"-</code>	explicit hyphen sign, allowing hyphenation in the rest of the word.
<code>" </code>	disable ligature at this position.

Table 8: Extra definitions made by file `catalan.ldf` (activated by default)

<code>'e</code>	acute accented a, allowing hyphenation in the rest of the word. Valid for the following vowels: e, i, o, u (both lowercase and uppercase).
<code>`a</code>	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.

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

<sup>29</sup>The file described in this section has version number ? and was last revised on ?.

<sup>30</sup>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, for example, to type `l'{}astre` instead of `l'astre`.

processing. If this command is undefined we proceed with the various definitions and first show the current version of this file.

```
24.1 (*code)
24.2 \ifx\undefined\captionscatalan
24.3 \else
24.4 \selectlanguage{catalan}
24.5 \expandafter\endinput
24.6 \fi
```

`\atcatcode` This file, `catalan.ldf`, may have been read while  $\text{T}_{\text{E}}\text{X}$  is in the middle of processing a document, so we have to make sure the category code of `@` is ‘letter’ while this file is being read. We save the category code of the `@`-sign in `\atcatcode` and make it ‘letter’. Later the category code can be restored to whatever it was before.

```
24.7 \chardef\atcatcode=\catcode'\@
24.8 \catcode'\@=11\relax
```

Now we determine whether the common macros from the file `babel.def` need to be read. We can be in one of two situations: either another language option has been read earlier on, in which case that other option has already read `babel.def`, or `catalan` is the first language option to be processed. In that case we need to read `babel.def` right here before we continue.

```
24.9 \ifx\undefined\babel@core@loaded\input babel.def\fi
```

Another check that has to be made, is if another language definition file has been read already. In that case its definitions have been activated. This might interfere with definitions this file tries to make. Therefore we make sure that we cancel any special definitions. This can be done by checking the existence of the macro `\originalTeX`. If it exists we simply execute it, otherwise it is `\let` to `\empty`.

```
24.10 \ifx\undefined\originalTeX
24.11 \let\originalTeX\empty
24.12 \fi
24.13 \originalTeX
```

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.

```
24.14 \ifx\undefined\l@catalan
24.15 \nopatterns{Catalan}
24.16 \adddialect\l@catalan0
24.17 \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  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ .

```
24.18 \addto\captionscatalan{%
24.19 \def\prefacename{Prefaci}%
24.20 \def\refname{Refer`encies}%
```

```

24.21 \def\abstractname{Resum}%
24.22 \def\bibName{Bibliografia}%
24.23 \def\chaptername{Cap\'}{\i}tol}%
24.24 \def\appendixname{Ap\'}endix}%
24.25 \def\contentsname{\'}Index}%
24.26 \def\listfigurename{\'}Index de figures}%
24.27 \def\listtablename{\'}Index de taules}%
24.28 \def\indexname{\'}Index de mat\'}eries}%
24.29 \def\figurename{Figura}%
24.30 \def\tablename{Taula}%
24.31 \def\partname{Part}%
24.32 \def\enclname{Adjunt}%
24.33 \def\ccname{C\'}opia a}%
24.34 \def\headtoname{A}%
24.35 \def\pagename{P\'}agina}%
24.36 \def\seenname{veure}%
24.37 \def\alsoname{veure tamb\'}e}%
24.38 \def\proofname{Demostraci\'}o}%
24.39 }

```

`\datecatalan` The macro `\datecatalan` redefines the command `\today` to produce Catalan dates. Months are written in lowercase<sup>31</sup>.

```

24.40 \def\datecatalan{%
24.41 \def\today{\number\day~\ifcase\month\or
24.42 de gener\or de febrer\or de mar\c{c}\or d'abril\or de maig\or
24.43 de juny\or de juliol\or d'agost\or de setembre\or d'octubre\or
24.44 de novembre\or de desembre\fi
24.45 \space de~\number\year}}

```

`\extrascatalan` 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`.

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.

```

24.46 \addto\extrascatalan{\languageshorthands{catalan}}
24.47 \initiate@active@char{"}
24.48 \addto\extrascatalan{\bbl@activate{''}}
24.49 \@ifpackagewith{babel}{activegrave}{%
24.50 \initiate@active@char{'}
24.51 \addto\extrascatalan{\bbl@activate{'}}{}}
24.52 \@ifpackagewith{babel}{activeacute}{%
24.53 \initiate@active@char{'}
24.54 \addto\extrascatalan{\bbl@activate{'}}{}}
24.55 %\addto\noextrascatalan{%
24.56 % \bbl@deactivate{"}
24.57 % \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.

<sup>31</sup>This seems to be the common practice. See for example: E. Coromina, *El 9 Nou: Manual de redacció i estil*, Ed. Eumo, Vic, 1993

We provide new definitions for the accent macros when one or both of the options `activegrave` or `activeacute` were specified.

```

24.58 \addto\extrascatalan{%
24.59   \babel@save\"
24.60   \def\"{\protect\@umlaut}}%
24.61 \@ifpackagewith{babel}{activegrave}{%
24.62   \babel@save\'
24.63   \addto\extrascatalan{\def\'{\protect\@grave}}
24.64   }{}
24.65 \@ifpackagewith{babel}{activeacute}{%
24.66   \babel@save\'
24.67   \addto\extrascatalan{\def\'{\protect\@acute}}
24.68   }{}

```

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` The original definition of `\"` is stored as `\dieresis`, because the definition of `\textacute` `\"` might not be the default plain T<sub>E</sub>X 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 `\'`.

```

24.69 \let\dieresis\"
24.70 \@ifpackagewith{babel}{activegrave}{\let\textgrave\'}{}
24.71 \@ifpackagewith{babel}{activeacute}{\let\textacute\'}{}

```

`\@umlaut` 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 T<sub>E</sub>X. 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 T<sub>E</sub>X at least finds the breaks that are not too close to the accent. The warning will only be printed to the log file.

```

24.72 \ifx\DeclareFontShape\undefined
24.73   \wlog{Warning: You are using an old LaTeX}
24.74   \wlog{Some word breaks will not be found.}
24.75   \def\@umlaut#1{\allowhyphens\dieresis{#1}\allowhyphens}
24.76   \@ifpackagewith{babel}{activeacute}{%
24.77     \def\@acute#1{\allowhyphens\textacute{#1}\allowhyphens}}{}
24.78   \@ifpackagewith{babel}{activegrave}{%
24.79     \def\@grave#1{\allowhyphens\textgrave{#1}\allowhyphens}}{}
24.80 \else
24.81   \edef\next{T1}
24.82   \ifx\f@encoding\next
24.83     \let\@umlaut\dieresis
24.84     \@ifpackagewith{babel}{activeacute}{%
24.85       \let\@acute\textacute}{}
24.86     \@ifpackagewith{babel}{activegrave}{%

```

```

24.87     \let\@grave\textgrave}{-}
24.88 \else
24.89     \wlog{Warning: You are using encoding \f@encoding\space
24.90         instead of T1.}
24.91     \wlog{Some word breaks will not be found.}
24.92     \def\@umlaut#1{\allowhyphens\dieresis{#1}\allowhyphens}
24.93     \@ifpackagewith{babel}{activeacute}{%-
24.94         \def\@acute#1{\allowhyphens\textacute{#1}\allowhyphens}}{-}
24.95     \@ifpackagewith{babel}{activegrave}{%-
24.96         \def\@grave#1{\allowhyphens\textgrave{#1}\allowhyphens}}{-}
24.97 \fi
24.98 \fi

```

If the user setup has extended fonts, the Ferguson macros are required to be defined. We check for their existence and, if defined, expand to whatever they are defined to. For instance, `\'a` would check for the existence 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.<sup>32</sup>

Now we can define our shorthands: the diaeresis and “*ela geminada*” support,

```

24.99 \declare@shorthand{catalan}{\i}{\textormath{\@umlaut i}{\ddot i\math}}
24.100 \declare@shorthand{catalan}{\l}{\lgem{}}
24.101 \declare@shorthand{catalan}{\u}{\textormath{\@umlaut u}{\ddot u}}
24.102 \declare@shorthand{catalan}{\I}{\textormath{\@umlaut I}{\ddot I}}
24.103 \declare@shorthand{catalan}{\L}{\Lgem{}}
24.104 \declare@shorthand{catalan}{\U}{\textormath{\@umlaut U}{\ddot U}}
    cedille,
24.105 \declare@shorthand{catalan}{\c}{\textormath{\c c}{\prime c}}
24.106 \declare@shorthand{catalan}{\C}{\textormath{\c C}{\prime C}}
    ‘french’ quote characters,
24.107 \declare@shorthand{catalan}{\<}{%-
24.108     \textormath{\guillemotleft}}{\mbox{\guillemotleft}}
24.109 \declare@shorthand{catalan}{\>}{%-
24.110     \textormath{\guillemotright}}{\mbox{\guillemotright}}
    grave accents,
24.111 \@ifpackagewith{babel}{activegrave}{%-
24.112     \declare@shorthand{catalan}{\a}{\textormath{\@grave a}{\grave a}}
24.113     \declare@shorthand{catalan}{\e}{\textormath{\@grave e}{\grave e}}
24.114     \declare@shorthand{catalan}{\o}{\textormath{\@grave o}{\grave o}}
24.115     \declare@shorthand{catalan}{\A}{\textormath{\@grave A}{\grave A}}
24.116     \declare@shorthand{catalan}{\E}{\textormath{\@grave E}{\grave E}}

```

---

<sup>32</sup>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.

```

24.117 \declare@shorthand{catalan}{'0}{\textormath{\@grave 0}{\grave 0}}
24.118 }{}

acute accents,

24.119 \@ifpackagewith{babel}{activeacute}{%
24.120 \declare@shorthand{catalan}{'a}{\textormath{\@acute a}{^{\prime} a}}
24.121 \declare@shorthand{catalan}{'e}{\textormath{\@acute e}{^{\prime} e}}
24.122 \declare@shorthand{catalan}{'i}{\textormath{\@acute i}{^{\prime} i}}
24.123 \declare@shorthand{catalan}{'o}{\textormath{\@acute o}{^{\prime} o}}
24.124 \declare@shorthand{catalan}{'u}{\textormath{\@acute u}{^{\prime} u}}
24.125 \declare@shorthand{catalan}{'A}{\textormath{\@acute A}{^{\prime} A}}
24.126 \declare@shorthand{catalan}{'E}{\textormath{\@acute E}{^{\prime} E}}
24.127 \declare@shorthand{catalan}{'I}{\textormath{\@acute I}{^{\prime} I}}
24.128 \declare@shorthand{catalan}{'O}{\textormath{\@acute O}{^{\prime} O}}
24.129 \declare@shorthand{catalan}{'U}{\textormath{\@acute U}{^{\prime} U}}

the acute accent,

24.130 \declare@shorthand{catalan}{''}{%
24.131 \textormath{\textquotedblright}{\sp\bgroup\prim@s'}}
24.132 }{}

and finally, some support definitions

24.133 \declare@shorthand{catalan}{"-}{\allowhyphens-\allowhyphens}
24.134 \declare@shorthand{catalan}{"|}{%
24.135 \textormath{\penalty\@M\discretionary{-}{-}{\kern.03em}%
24.136 \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 T<sub>E</sub>X 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 T<sub>E</sub>X can generate from the hyphenation patterns. However, the average length of words in Catalan makes this desirable and so it is kept here.

```

24.137 \addto\extrascatalan{%
24.138 \babel@save{\-}%
24.139 \def\-\{\allowhyphens\discretionary{-}{-}{\allowhyphens}}

```

**\lgem** Here we define a macro for typing the catalan “ela geminada” (geminated l).  
**\Lgem** The macros **\lgem** and **\Lgem** have been chosen for its lowercase and uppercase representation, respectively<sup>33</sup>.

The code used in the actual macro used is a combination of the one proposed by Feruglio and Fuster<sup>34</sup> and the proposal from Valiente to appear in the next T<sub>E</sub>X Users Group Annual Meeting (1995). This last proposal has not been fully implemented due to its limitation to CM fonts.

```

24.140 \newskip\zzz
24.141 \newdimen\leftllkern \newdimen\rightllkern \newdimen\raiselldim
24.142 \def\lgem{\relax\ifmmode\orig@ll
24.143 \else

```

<sup>33</sup>The macro names \ll and \LL were not taken because of the fact that \ll is already used in mathematical mode.

<sup>34</sup>G.V. Valiente and R. Fuster, Typesetting Catalan Texts with T<sub>E</sub>X, *TUGboat* 14(3), 1993.

```

24.144 \leftllkern=0pt\rightllkern=0pt\raiselldim=0pt%
24.145 \setbox0\hbox{1}\setbox1\hbox{1\}/\setbox2\hbox{.}%
24.146 \advance\raiselldim by \the\fontdimen5\the\font
24.147 \advance\raiselldim by -\ht2%
24.148 \leftllkern=-.25\wd0%
24.149 \advance\leftllkern by \wd1%
24.150 \advance\leftllkern by -\wd0%
24.151 \rightllkern=-.25\wd0%
24.152 \advance\rightllkern by -\wd1%
24.153 \advance\rightllkern by \wd0%
24.154 \allowhyphens\discretionary{1-}{1}%
24.155 {\hbox{1}\kern\leftllkern\raise\raiselldim\hbox{.}%
24.156 \kern\rightllkern\hbox{1}}\allowhyphens
24.157 }
24.158 \def\Lgem{\leftllkern=0pt\rightllkern=0pt\raiselldim=0pt%
24.159 \setbox0\hbox{L}\setbox1\hbox{L\}/\setbox2\hbox{.}%
24.160 \advance\raiselldim by .5\ht0%
24.161 \advance\raiselldim by -.5\ht2%
24.162 \leftllkern=-.125\wd0%
24.163 \advance\leftllkern by \wd1%
24.164 \advance\leftllkern by -\wd0%
24.165 \rightllkern=-\wd0%
24.166 \divide\rightllkern by 6%
24.167 \advance\rightllkern by -\wd1%
24.168 \advance\rightllkern by \wd0%
24.169 \allowhyphens\discretionary{L-}{L}%
24.170 {\hbox{L}\kern\leftllkern\raise\raiselldim\hbox{.}%
24.171 \kern\rightllkern\hbox{L}}\allowhyphens}

```

\up A macro for typesetting things like 1<sup>er</sup> as proposed by Raymon Seroul<sup>35</sup>.

```
24.172 \def\up#1{\raise 1ex\hbox{\small#1}}
```

It is possible that a site might need to add some extra code to the babel macros. To enable this we load a local configuration file, `catalan.cfg` if it is found on T<sub>E</sub>X' search path.

```
24.173 \loadlocalcfg{catalan}
```

Our last action is to make a note that the commands we have just defined, will be executed by calling the macro `\selectlanguage` at the beginning of the document.

```
24.174 \main@language{catalan}
```

Finally, the category code of `@` is reset to its original value. The macrospace used by `\atcatcode` is freed.

```
24.175 \catcode'\@=\atcatcode \let\atcatcode\relax
24.176 \</code>
```

---

<sup>35</sup>This macro has been borrowed from `francais.dtx`

## 25 The Galician language

The file `galician.dtx`<sup>36</sup> 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

<code>" </code>	disable ligature at this position.
<code>"-</code>	an explicit hyphen sign, allowing hyphenation in the rest of the word.
<code>\-</code>	like the old <code>\-</code> , but allowing hyphenation in the rest of the word.
<code>'a</code>	an accent that allows hyphenation. Valid for all vowels uppercase and lowercase.
<code>'n</code>	a n with a tilde. This is included to improve compatibility with FTC. Works for uppercase too.
<code>"u</code>	a u with dieresis allowing hyphenation.
<code>"a</code>	feminine ordinal as in 1 <sup>a</sup> .
<code>"o</code>	masculine ordinal as in 1 <sup>o</sup> .
<code>~n</code>	a n with tilde. Works for uppercase too.

Table 10: The extra definitions made by `galician.1df`

to their original definitions if not followed by one of the characters indicated in that table.

As this file needs to be read only once, we check whether it was read before. If it was, the command `\captionsgalician` is already defined, so we can stop processing. If this command is undefined we proceed with the various definitions and first show the current version of this file.

```
25.1 (*code)
25.2 \ifx\undefined\captionsgalician
25.3 \else
25.4 \selectlanguage{galician}
25.5 \expandafter\endinput
25.6 \fi
```

`\atcatcode` This file, `galician.1df`, may have been read while  $\TeX$  is in the middle of processing a document, so we have to make sure the category code of `@` is ‘letter’ while this file is being read. We save the category code of the `@`-sign in `\atcatcode` and make it ‘letter’. Later the category code can be restored to whatever it was before.

```
25.7 \chardef\atcatcode=\catcode'\@
25.8 \catcode'\@=11\relax
```

Now we determine whether the the common macros from the file `babel.def` need to be read. We can be in one of two situations: either another language option has been read earlier on, in which case that other option has already read `babel.def`, or `galician` is the first language option to be processed. In that case we need to read `babel.def` right here before we continue.

```
25.9 \ifx\undefined\babel@core@loaded\input babel.def\relax\fi
```

<sup>36</sup>The file described in this section has version number ? and was last revised on ?.

Another check that has to be made, is if another language definition file has been read already. In that case its definitions have been activated. This might interfere with definitions this file tries to make. Therefore we make sure that we cancel any special definitions. This can be done by checking the existence of the macro `\originalTeX`. If it exists we simply execute it.

```
25.10 \ifx\undefined\originalTeX
25.11 \let\originalTeX\empty
25.12 \fi
25.13 \originalTeX
```

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.

```
25.14 \ifx\undefined\l@galician
25.15 \@nopatterns{Galician}
25.16 \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 L<sup>A</sup>T<sub>E</sub>X.

```
25.17 \addto\captionsgalician{%
25.18 \def\prefacename{Prefacio}%
25.19 \def\refname{Referencias}%
25.20 \def\abstractname{Resumo}%
25.21 \def\bibname{Bibliograf\’{\i}a}%
25.22 \def\chaptername{Cap\’{\i}tulo}%
25.23 \def\appendixname{Ap\’endice}%
25.24 \def\contentsname{\’Indice Xeral}%
25.25 \def\listfigurename{\’Indice de Figuras}%
25.26 \def\listtablename{\’Indice de T\’aboas}%
25.27 \def\indexname{\’Indice de Materias}%
25.28 \def\figurename{Figura}%
25.29 \def\tablename{T\’aboa}%
25.30 \def\partname{Parte}%
25.31 \def\enclname{Adxunto}%
25.32 \def\ccname{Copia a}%
25.33 \def\headtoname{A}%
25.34 \def\pagename{P\’axina}%
25.35 \def\seename{v\’exase}%
25.36 \def\alsoname{v\’exase tam\’en}%
25.37 \def\proofname{Proof}% <-- Needs Translation!
25.38 }
```

`\dategalician` The macro `\dategalician` redefines the command `\today` to produce Galician dates.

```
25.39 \def\dategalician{%
25.40 \def\today{\number\day~de\space\ifcase\month\or
25.41 xaneiro\or febreiro\or marzo\or abril\or maio\or xu\~no\or
25.42 xullo\or agosto\or setembro\or outubro\or novembro\or decembro\fi
25.43 \space de~\number\year}}
```

`\extragalician` The macro `\extragalician` will perform all the extra definitions needed for the Galician language. The macro `\noextragalician` is used to cancel the actions of `\extragalician`.

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.

```
25.44 \addto\extragalician{\languageshorthands{galician}}
25.45 \initiate@active@char{"}
25.46 \initiate@active@char{~}
25.47 \addto\extragalician{%
25.48   \bbl@activate{"}\bbl@activate{~}}
25.49 \@ifpackagewith{babel}{activeacute}{%
25.50   \initiate@active@char{'}
25.51   \addto\extragalician{\bbl@activate{'}}}{%
25.52 %\addto\noextragalician{%
25.53 %   \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.

```
25.54 \addto\extragalician{%
25.55   \babel@save"\\babel@save\~
25.56   \def"\{\protect\@umlaut}%
25.57   \def\~{\protect\@tilde}}
25.58 \@ifpackagewith{babel}{activeacute}{%
25.59   \babel@save\'
25.60   \addto\extragalician{\def\'\{\protect\@acute}}
25.61   }
```

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` 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 `\~`.

```

25.62 \let\dieresis\"
25.63 \let\texttilde\~
25.64 \@ifpackagewith{babel}{activeacute}{\let\textacute\'}{\}

```

`\@umlaut` If the user setup has extended fonts, the Ferguson macros are required to be defined. We check for their existence and, if defined, expand to whatever they are defined to. For instance, `\'a` would check for the existence 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- $\TeX$ .<sup>37</sup>

```

25.65 \def\@umlaut#1{\allowhyphens\dieresis{#1}\allowhyphens}
25.66 \def\@tilde#1{\allowhyphens\texttilde{#1}\allowhyphens}
25.67 \@ifpackagewith{babel}{activeacute}{\%
25.68 \def\@acute#1{\allowhyphens\textacute{#1}\allowhyphens}}{\}

```

Now we can define our shorthands: the umlauts,

```

25.69 \declare@shorthand{galician}{"-}{\allowhyphens-\allowhyphens}
25.70 \declare@shorthand{galician}{"|}{\discretionary{-}{\kern.03em}}
25.71 \declare@shorthand{galician}{"u}{\@umlaut{u}}
25.72 \declare@shorthand{galician}{"U}{\@umlaut{U}}

```

ordinals<sup>38</sup>,

```

25.73 \declare@shorthand{galician}{"o}{\%
25.74 \raise1ex\hbox{\underbar{\scriptsize o}}}
25.75 \declare@shorthand{galician}{"a}{\%
25.76 \raise1ex\hbox{\underbar{\scriptsize a}}}

```

acute accents,

```

25.77 \@ifpackagewith{babel}{activeacute}{\%
25.78 \declare@shorthand{galician}{'a}{\textormath{\@acute a}{\prime} a}}
25.79 \declare@shorthand{galician}{'e}{\textormath{\@acute e}{\prime} e}}
25.80 \declare@shorthand{galician}{'i}{\textormath{\@acute i}{\prime} i}}
25.81 \declare@shorthand{galician}{'o}{\textormath{\@acute o}{\prime} o}}
25.82 \declare@shorthand{galician}{'u}{\textormath{\@acute u}{\prime} u}}
25.83 \declare@shorthand{galician}{'A}{\textormath{\@acute A}{\prime} A}}
25.84 \declare@shorthand{galician}{'E}{\textormath{\@acute E}{\prime} E}}
25.85 \declare@shorthand{galician}{'I}{\textormath{\@acute I}{\prime} I}}
25.86 \declare@shorthand{galician}{'O}{\textormath{\@acute O}{\prime} O}}
25.87 \declare@shorthand{galician}{'U}{\textormath{\@acute U}{\prime} U}}

```

tildes,

```

25.88 \declare@shorthand{galician}{'n}{\textormath{\~n}{\prime} n}}
25.89 \declare@shorthand{galician}{'N}{\textormath{\~N}{\prime} N}}

```

<sup>37</sup>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.

<sup>38</sup>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`.

```

25.90 }{}
25.91 \declare@shorthand{galician}{~n}{\textormath{~n}{\@tilde n}}
25.92 \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 T<sub>E</sub>X 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 T<sub>E</sub>X can generate from the hyphenation patterns. However, the average length of words in Galician makes this desirable and so it is kept here.

```

25.93 \addto\extragalician{%
25.94 \babel@save{\-}%
25.95 \def\-\{\allowhyphens\discretionary{-}{\}\allowhyphens}}

```

It is possible that a site might need to add some extra code to the babel macros. To enable this we load a local configuration file, `galician.cfg` if it is found on T<sub>E</sub>X' search path.

```
25.96 \loadlocalcfg{galician}
```

Our last action is to make a note that the commands we have just defined, will be executed by calling the macro `\selectlanguage` at the beginning of the document.

```
25.97 \main@language{galician}
```

Finally, the category code of @ is reset to its original value. The macrospace used by `\atcatcode` is freed.

```

25.98 \catcode'\@\atcatcode \let\atcatcode\relax
25.99 </code>

```

## 26 The Romanian language

The file `romanian.dtx`<sup>39</sup> defines all the language-specific macros for the Romanian language.

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

As this file needs to be read only once, we check whether it was read before. If it was, the command `\captionromanian` is already defined, so we can stop processing. If this command is undefined we proceed with the various definitions and first show the current version of this file.

```
26.1 <*code>
26.2 \ifx\undefined\captionromanian
26.3 \else
26.4   \selectlanguage{romanian}
26.5   \expandafter\endinput
26.6 \fi
```

`\atcatcode` This file, `romanian.sty`, may have been read while TeX is in the middle of processing a document, so we have to make sure the category code of `@` is ‘letter’ while this file is being read. We save the category code of the `@`-sign in `\atcatcode` and make it ‘letter’. Later the category code can be restored to whatever it was before.

```
26.7 \chardef\atcatcode=\catcode'\@
26.8 \catcode'\@=11\relax
```

Now we determine whether the the common macros from the file `babel.def` need to be read. We can be in one of two situations: either another language option has been read earlier on, in which case that other option has already read `babel.def`, or `romanian` is the first language option to be processed. In that case we need to read `babel.def` right here before we continue.

```
26.9 \ifx\undefined\babel@core@loaded\input babel.def\relax\fi
```

Another check that has to be made, is if another language definition file has been read already. In that case its definitions have been activated. This might interfere with definitions this file tries to make. Therefore we make sure that we cancel any special definitions. This can be done by checking the existence of the macro `\originalTeX`. If it exists we simply execute it, otherwise it is `\let` to `\empty`.

```
26.10 \ifx\undefined\originalTeX \let\originalTeX\empty \else\originalTeX\fi
```

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.

```
26.11 \ifx\undefined\l@romanian
26.12   \nopatterns{Romanian}
26.13   \adddialect\l@romanian0\fi
```

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

---

<sup>39</sup>The file described in this section has version number ? and was last revised on ?. A contribution was made by Umstatter Horst (`hhu@cernvm.cern.ch`).

`\captionsromanian` The macro `\captionsromanian` defines all strings used in the four standard documentclasses provided with L<sup>A</sup>T<sub>E</sub>X.

```
26.14 \addto\captionsromanian{%
26.15   \def\prefacename{Prefa\c{t}\u{a}}%
26.16   \def\refname{Bibliografie}%
26.17   \def\abstractname{Rezumat}%
26.18   \def\bibName{Bibliografie}%
26.19   \def\chaptername{Capitolul}%
26.20   \def\appendixname{Anexa}%
26.21   \def\contentsname{Cuprins}%
26.22   \def\listfigurename{List\u{a} de figuri}%
26.23   \def\listtablename{List\u{a} de tabele}%
26.24   \def\indexname{Glosar}%
26.25   \def\figurename{Figura}%      % sau Plan\c{s}a
26.26   \def\tablename{Tabela}%
26.27   \def\partname{Partea}%
26.28   \def\enclname{Anex\u{a}}%    % sau Anexe
26.29   \def\ccname{Copie}%
26.30   \def\headtoname{Pentru}%
26.31   \def\pagename{Pagina}%
26.32   \def\seename{Vezi}%
26.33   \def\alsoname{Vezi de asemenea}%
26.34   \def\proofname{Proof}%      <-- needs translation
26.35 }%
```

`\dateromanian` The macro `\dateromanian` redefines the command `\today` to produce Romanian dates.

```
26.36 \def\dateromanian{%
26.37 \def\today{\number\day~\ifcase\month\or
26.38   ianuarie\or februarie\or martie\or aprilie\or mai\or
26.39   iunie\or iulie\or august\or septembrie\or octombrie\or
26.40   noiembrie\or decembrie\fi
26.41   \space \number\year}}
```

`\extrasromanian` 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.

```
26.42 \addto\extrasromanian{}
26.43 \addto\noextrasromanian{}
```

It is possible that a site might need to add some extra code to the babel macros. To enable this we load a local configuration file, `romanian.cfg` if it is found on T<sub>E</sub>X' search path.

```
26.44 \loadlocalcfg{romanian}
```

Our last action is to make a note that the commands we have just defined, will be executed by calling the macro `\selectlanguage` at the beginning of the document.

```
26.45 \main@language{romanian}
```

Finally, the category code of `@` is reset to its original value. The macrospace used by `\atcatcode` is freed.

```
26.46 \catcode'\@=\atcatcode \let\atcatcode\relax
26.47 \end{code}
```

## 27 The Danish language

The file `danish.dtx`<sup>40</sup> 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`

As this file needs to be read only once, we check whether it was read before. If it was, the command `\captiondanish` is already defined, so we can stop processing. If this command is undefined we proceed with the various definitions and first show the current version of this file.

```
27.1 {*code}
27.2 \ifx\undefined\captiondanish
27.3 \else
27.4 \selectlanguage{danish}
27.5 \expandafter\endinput
27.6 \fi
```

`\atcatcode` This file, `danish.sty`, may have been read while  $\TeX$  is in the middle of processing a document, so we have to make sure the category code of `@` is 'letter' while this file is being read. We save the category code of the `@`-sign in `\atcatcode` and make it 'letter'. Later the category code can be restored to whatever it was before.

```
27.7 \chardef\atcatcode=\catcode'\@
27.8 \catcode'\@=11\relax
```

Now we determine whether the the common macros from the file `babel.def` need to be read. We can be in one of two situations: either another language option has been read earlier on, in which case that other option has already read `babel.def`, or `danish` is the first language option to be processed. In that case we need to read `babel.def` right here before we continue.

```
27.9 \ifx\undefined\babel@core@loaded\input babel.def\relax\fi
```

Another check that has to be made, is if another language definition file has been read already. In that case its definitions have been activated. This might interfere with definitions this file tries to make. Therefore we make sure that we cancel any special definitions. This can be done by checking the existence of the

---

<sup>40</sup>The file described in this section has version number ? and was last revised on ?. A contribution was made by Henning Larsen (`larsen@cernvm.cern.ch`)

macro `\originalTeX`. If it exists we simply execute it, otherwise it is `\let` to `\empty`.

```
27.10 \ifx\undefined\originalTeX \let\originalTeX\empty \else\originalTeX\fi
```

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.

```
27.11 \ifx\undefined\l@danish
```

```
27.12     \nopatterns{Danish}
```

```
27.13     \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  $\LaTeX$ .

```
27.14 \addto\captionsdanish{%
```

```
27.15     \def\prefacename{Forord}%
```

```
27.16     \def\refname{Litteratur}%
```

```
27.17     \def\abstractname{Resum'e}%
```

```
27.18     \def\bibname{Litteratur}%
```

```
27.19     \def\chaptername{Kapitel}%
```

```
27.20     \def\appendixname{Bilag}%
```

```
27.21     \def\contentsname{Indhold}%
```

```
27.22     \def\listfigurename{Figurer}%
```

```
27.23     \def\listtablename{Tabeller}%
```

```
27.24     \def\indexname{Indeks}%
```

```
27.25     \def\figurename{Figur}%
```

```
27.26     \def\tablename{Tabel}%
```

```
27.27     \def\partname{Del}%
```

```
27.28     \def\enclname{Vedlagt}%
```

```
27.29     \def\ccname{Kopi til}% or Kopi sendt til
```

```
27.30     \def\headtoname{Til}% in letter
```

```
27.31     \def\pagename{Side}%
```

```
27.32     \def\seename{Se}%
```

```
27.33     \def\alsoname{Se ogs{\aa}}%
```

```
27.34     \def\proofname{Proof}% <-- needs translation!
```

```
27.35     }%
```

`\datedanish` The macro `\datedanish` redefines the command `\today` to produce Danish dates.

```
27.36 \def\datedanish{%
```

```
27.37 \def\today{\number\day.\~\ifcase\month\or
```

```
27.38 januar\or februar\or marts\or april\or maj\or juni\or
```

```
27.39 juli\or august\or september\or oktober\or november\or december\fi
```

```
27.40 \space\number\year}}
```

`\extrasdanish` 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 `\frenchspacing` to be in effect.

```
27.41 \addto\extrasdanish{\bbl@frenchspacing}
```

```
27.42 \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.

```
27.43 \initiate@active@char{"}
27.44 \addto\extrasdanish{\languageshorthands{danish}}
27.45 \addto\extrasdanish{\bbl@activate{"}}
27.46 %\addto\noextrasdanish{\bbl@deactivate{"}}
```

First we define access to the low opening double quote and guillemets for quotations,

```
27.47 \declare@shorthand{danish}{"'}{%
27.48 \textormath{\quotedblbase}}{\mbox{\quotedblbase}}}
27.49 \declare@shorthand{danish}{"'}{%
27.50 \textormath{\textquotedblright}}{\mbox{\textquotedblright}}}
27.51 \declare@shorthand{danish}{"<}{%
27.52 \textormath{\guillemotleft}}{\mbox{\guillemotleft}}}
27.53 \declare@shorthand{danish}{">}{%
27.54 \textormath{\guillemotright}}{\mbox{\guillemotright}}}
```

then we define two shorthands to be able to specify hyphenation breakpoints that behave a little different from \-.

```
27.55 \declare@shorthand{danish}{"-}{\allowhyphens-\allowhyphens}
27.56 \declare@shorthand{danish}{"}{\hskip\z@skip}
```

And we want to have a shorthand for disabling a ligature.

```
27.57 \declare@shorthand{danish}{"|}{%
27.58 \textormath{\discretionary{-}}{\kern.03em}}{}}
```

It is possible that a site might need to add some extra code to the babel macros. To enable this we load a local configuration file, `danish.cfg` if it is found on T<sub>E</sub>X' search path.

```
27.59 \loadlocalcfg{danish}
```

Our last action is to make a note that the commands we have just defined, will be executed by calling the macro `\selectlanguage` at the beginning of the document.

```
27.60 \main@language{danish}
```

Finally, the category code of @ is reset to its original value. The macrospace used by `\atcatcode` is freed.

```
27.61 \catcode'\@=\atcatcode \let\atcatcode\relax
27.62 </code>
```

## 28 The Norwegian language

The file `norsk.dtx`<sup>41</sup> 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.

As this file needs to be read only once, we check whether it was read before. If it was, the command `\captionnorsk` is already defined, so we can stop processing. If this command is undefined we proceed with the various definitions and first show the current version of this file.

```
28.1 (*code)
28.2 \ifx\undefined\captionnorsk
28.3 \else
28.4 \selectlanguage{norsk}
28.5 \expandafter\endinput
28.6 \fi
```

`\atcatcode` This file, `norsk.sty`, may have been read while  $\TeX$  is in the middle of processing a document, so we have to make sure the category code of `@` is ‘letter’ while this file is being read. We save the category code of the `@`-sign in `\atcatcode` and make it ‘letter’. Later the category code can be restored to whatever it was before.

```
28.7 \chardef\atcatcode=\catcode'\@
28.8 \catcode'\@=11\relax
```

Now we determine whether the the common macros from the file `babel.def` need to be read. We can be in one of two situations: either another language option has been read earlier on, in which case that other option has already read `babel.def`, or `norsk` is the first language option to be processed. In that case we need to read `babel.def` right here before we continue.

```
28.9 \ifx\undefined\babel@core@loaded\input babel.def\relax\fi
```

Another check that has to be made, is if another language definition file has been read already. In that case its definitions have been activated. This might interfere with definitions this file tries to make. Therefore we make sure that we cancel any special definitions. This can be done by checking the existence of the macro `\originalTeX`. If it exists we simply execute it, otherwise it is `\let` to `\empty`.

```
28.10 \ifx\undefined\originalTeX \let\originalTeX\empty \else\originalTeX\fi
```

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.

```
28.11 \ifx\undefined\l@norsk
28.12 \nopatterns{Norsk}
28.13 \adddialect\l@norsk0\fi
```

For the ‘nynorsk’ version of these definitions we just add a “dialect”. Also, the macros `\datenynorsk` and `\extrasnynorsk` are `\let` to their ‘norsk’ counterparts when these parts are defined.

```
28.14 \adddialect\l@nynorsk\l@norsk
```

---

<sup>41</sup>The file described in this section has version number ? and was last revised on ?. 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`).

`\norskhyphenmins` The Norwegian hyphenation patterns can be used with `\lefthyphenmin` set to 1 and `\nynorskhyphenmins` and `\righthyphenmin` set to 2. This is true for both ‘versions’ of the language.

```
28.15 \def\norskhyphenmins{\@ne\tw@}
28.16 \let\nynorskhyphenmins\norskhyphenmins
```

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 L<sup>A</sup>T<sub>E</sub>X.

```
28.17 \addto\captionsnorsk{%
28.18   \def\prefacename{Forord}%
28.19   \def\refname{Referanser}%
28.20   \def\abstractname{Sammendrag}%
28.21   \def\bibname{Bibliografi}%       or Litteraturoversikt
28.22 %                               or Litteratur or Referanser
28.23   \def\chaptername{Kapittel}%
28.24   \def\appendixname{Tillegg}%     or Appendiks
28.25   \def\contentsname{Innhold}%
28.26   \def\listfigurename{Figurer}%  or Figurliste
28.27   \def\listtablename{Tabeller}%  or Tabelliste
28.28   \def\indexname{Register}%
28.29   \def\figurename{Figur}%
28.30   \def\tablename{Tabell}%
28.31   \def\partname{Del}%
28.32   \def\enclname{Vedlegg}%
28.33   \def\ccname{Kopi sendt}%
28.34   \def\headtoname{Til}% in letter
28.35   \def\pagename{Side}%
28.36   \def\seenname{Se}%
28.37   \def\alsiname{Se ogs\aa{}}%
28.38   \def\proofname{Proof}%
28.39 }
```

`\captionsnynorsk` The macro `\captionsnynorsk` defines all strings used in the four standard documentclasses provided with L<sup>A</sup>T<sub>E</sub>X, but using a different spelling than in the command `\captionsnorsk`.

```
28.40 \addto\captionsnynorsk{%
28.41   \def\prefacename{Forord}%
28.42   \def\refname{Referansar}%
28.43   \def\abstractname{Samandrag}%
28.44   \def\bibname{Litteratur}%       or Litteraturoversyn
28.45   %                               % or Referansar
28.46   \def\chaptername{Kapittel}%
28.47   \def\appendixname{Tillegg}%     or Appendiks
28.48   \def\contentsname{Innhald}%
28.49   \def\listfigurename{Figurar}%  or Figurliste
28.50   \def\listtablename{Tabellar}%  or Tabelliste
28.51   \def\indexname{Register}%
28.52   \def\figurename{Figur}%
28.53   \def\tablename{Tabell}%
28.54   \def\partname{Del}%
28.55   \def\enclname{Vedlegg}%
```

```

28.56 \def\ccname{Kopi sendt}%
28.57 \def\headtoname{Til}% in letter
28.58 \def\pagename{Side}%
28.59 \def\seenname{Sj\aa{}}%
28.60 \def\alsoname{Sj\aa{} ogs\aa{}}%
28.61 \def\proofname{Proof}%
28.62 }

```

`\datenorsk` The macro `\datenorsk` redefines the command `\today` to produce Norwegian dates.

```

28.63 \def\datenorsk{%
28.64 \def\today{\number\day.\~\ifcase\month\or
28.65 januar\or februar\or mars\or april\or mai\or juni\or
28.66 juli\or august\or september\or oktober\or november\or desember\fi
28.67 \space\number\year}}

```

`\datenynorsk` The spelling of the names of the months is the same for both versions of the “Norsk” language, so we simply `\let` the macro `\datenynorsk` be equal to `\datenorsk`

```
28.68 \let\datenynorsk\datenorsk
```

`\extrasnorsk` 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 `\frenchspacing` to be in effect.

```

28.69 \addto\extrasnorsk{\bbl@frenchspacing}
28.70 \addto\noextrasnorsk{\bbl@nonfrenchspacing}

```

`\extrasynorsk` Also for the “nynorsk” variant no extra definitions are needed at the moment.

```

\noextrasynorsk 28.71 \let\extrasynorsk\extrasnorsk
28.72 \let\noextrasynorsk\noextrasnorsk

```

It is possible that a site might need to add some extra code to the babel macros. To enable this we load a local configuration file, `norsk.cfg` if it is found on  $\TeX$  search path.

```
28.73 \loadlocalcfg{norsk}
```

Our last action is to make a note that the commands we have just defined, will be executed by calling the macro `\selectlanguage` at the beginning of the document.

```
28.74 \main@language{norsk}
```

Finally, the category code of `@` is reset to its original value. The macrospace used by `\atcatcode` is freed.

```

28.75 \catcode'\@=\atcatcode \let\atcatcode\relax
28.76 </code>

```

## 29 The Swedish language

The file `swedish.dtx`<sup>42</sup> defines all the language-specific macros for the Swedish language.

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

As this file needs to be read only once, we check whether it was read before. If it was, the command `\captionsswedish` is already defined, so we can stop processing. If this command is undefined we proceed with the various definitions and first show the current version of this file.

```
29.1 (*code)
29.2 \ifx\undefined\captionsswedish
29.3 \else
29.4   \selectlanguage{swedish}
29.5   \expandafter\endinput
29.6 \fi
```

`\atcatcode` This file, `swedish.sty`, may have been read while  $\TeX$  is in the middle of processing a document, so we have to make sure the category code of `@` is ‘letter’ while this file is being read. We save the category code of the `@`-sign in `\atcatcode` and make it ‘letter’. Later the category code can be restored to whatever it was before.

```
29.7 \chardef\atcatcode=\catcode'\@
29.8 \catcode'\@=11\relax
```

Now we determine whether the the common macros from the file `babel.def` need to be read. We can be in one of two situations: either another language option has been read earlier on, in which case that other option has already read `babel.def`, or `swedish` is the first language option to be processed. In that case we need to read `babel.def` right here before we continue.

```
29.9 \ifx\undefined\babel@core@loaded\input babel.def\relax\fi
```

Another check that has to be made, is if another language definition file has been read already. In that case its definitions have been activated. This might interfere with definitions this file tries to make. Therefore we make sure that we cancel any special definitions. This can be done by checking the existence of the macro `\originalTeX`. If it exists we simply execute it, otherwise it is `\let` to `\empty`.

```
29.10 \ifx\undefined\originalTeX \let\originalTeX\empty \else\originalTeX\fi
```

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.

```
29.11 \ifx\undefined\l@swedish
29.12   \nopatterns{Swedish}
29.13   \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.

---

<sup>42</sup>The file described in this section has version number ? and was last revised on ?. A contribution was made by Sten Hellman (`HELLMAN@CERNVM.CERN.CH`).

`\captionsswedish` The macro `\captionsswedish` defines all strings used in the four standard documentclasses provided with L<sup>A</sup>T<sub>E</sub>X.

```
29.14 \addto\captionsswedish{%
29.15   \def\prefacename{F\"orord}%
29.16   \def\refname{Referenser}%
29.17   \def\abstractname{Sammanfattning}%
29.18   \def\bibName{Litteraturf\"orteckning}%
29.19   \def\chaptername{Kapitel}%
29.20   \def\appendixname{Bilaga}%
29.21   \def\contentsname{Inneh\"osname aa\endcsname ll}%
29.22   \def\listfigurename{Figurer}%
29.23   \def\listtablename{Tabeller}%
29.24   \def\indexname{Sakregister}%
29.25   \def\figurename{Figur}%
29.26   \def\tablename{Tabell}%
29.27   \def\partname{Del}%
29.28   \def\enclname{Bil}%
29.29   \def\ccname{Kopia f\"or k\"annedom}%
29.30   \def\headtoname{Till}% in letter
29.31   \def\pagename{Sida}%
29.32   \def\seename{se}%

29.33   \def\alsoname{se \"aven}%
29.34   \def\proofname{Proof}% <-- needs translation
29.35 }
```

`\dateswedish` The macro `\dateswedish` redefines the command `\today` to produce Swedish dates.

```
29.36 \def\dateswedish{%
29.37 \def\today{\number\day~\ifcase\month\or
29.38 januari\or februari\or mars\or april\or maj\or juni\or
29.39 juli\or augusti\or september\or oktober\or november\or december\fi
29.40 \space\number\year}}
```

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

```
29.41 \def\swedishhyphenmins{\tw@\tw@}
```

`\extrasswedish` 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 the moment this macro is empty but it is defined for compatibility with the other language definition files.

For Swedish texts `\frenchspacing` should be in effect. We make sure this is the case and reset it if necessary.

```
29.42 \addto\extrasswedish{\bbl@frenchspacing}
29.43 \addto\noextrasswedish{\bbl@nonfrenchspacing}
```

It is possible that a site might need to add some extra code to the babel macros. To enable this we load a local configuration file, `swedish.cfg` if it is found on T<sub>E</sub>X' search path.

```
29.44 \loadlocalcfg{swedish}
```

Our last action is to make a note that the commands we have just defined, will be executed by calling the macro `\selectlanguage` at the beginning of the document.

```
29.45 \main@language{swedish}
```

Finally, the category code of `@` is reset to its original value. The macrospace used by `\atcatcode` is freed.

```
29.46 \catcode'\@=\atcatcode \let\atcatcode\relax
```

```
29.47 </code>
```

## 30 The Finnish language

The file `finnish.dtx`<sup>43</sup> defines all the language definition macros for the Finnish language.

For this language the character " is made active. In table 12 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 12: The extra definitions made by `finnish.lda`

As this file needs to be read only once, we check whether it was read before. If it was, the command `\captionfinnish` is already defined, so we can stop processing. If this command is undefined we proceed with the various definitions and first show the current version of this file.

```
30.1 <*code>
30.2 \ifx\undefined\captionfinnish
30.3 \else
30.4 \selectlanguage{finnish}
30.5 \expandafter\endinput
30.6 \fi
```

`\atcatcode` This file, `finnish.lda`, may have been read while  $\TeX$  is in the middle of processing a document, so we have to make sure the category code of @ is 'letter' while this file is being read. We save the category code of the @-sign in `\atcatcode` and make it 'letter'. Later the category code can be restored to whatever it was before.

```
30.7 \chardef\atcatcode=\catcode'\@
30.8 \catcode'\@=11\relax
```

Now we determine whether the the common macros from the file `babel.def` need to be read. We can be in one of two situations: either another language option has been read earlier on, in which case that other option has already read `babel.def`, or `finnish` is the first language option to be processed. In that case we need to read `babel.def` right here before we continue.

```
30.9 \ifx\undefined\babel@core@loaded\input babel.def\relax\fi
```

---

<sup>43</sup>The file described in this section has version number ? and was last revised on ?. A contribution was made by Mikko KANERVA (`KANERVA@CERNVM`) and Keranen Reino (`KERANEN@CERNVM`).

Another check that has to be made, is if another language definition file has been read already. In that case its definitions have been activated. This might interfere with definitions this file tries to make. Therefore we make sure that we cancel any special definitions. This can be done by checking the existence of the macro `\originalTeX`. If it exists we simply execute it, otherwise it is `\let` to `\empty`.

```
30.10 \ifx\undefined\originalTeX \let\originalTeX\empty \else\originalTeX\fi
```

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.

```
30.11 \ifx\undefined\l@finnish
30.12   \nopatterns{Finnish}
30.13   \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  $\LaTeX$ .

```
30.14 \addto\captionsfinnish{%
30.15   \def\prefacename{Esipuhe}%
30.16   \def\refname{Viitteet}%
30.17   \def\abstractname{Tiivistelmä}
30.18   \def\bibname{Kirjallisuutta}%
30.19   \def\chaptername{Luku}%
30.20   \def\appendixname{Liite}%
30.21   \def\contentsname{Sisältö} /* Could be "Sisällykset" as well */
30.22   \def\listfigurename{Kuvat}%
30.23   \def\listtablename{Taulukot}%
30.24   \def\indexname{Hakemistö}%
30.25   \def\figurename{Kuva}%
30.26   \def\tablename{Taulukko}%
30.27   \def\partname{Osa}%
30.28   \def\enclname{Liitteet}%
30.29   \def\ccname{Jakelu}%
30.30   \def\headtoname{Vastaaottaja}%
30.31   \def\pagename{Sivu}%
30.32   \def\seenname{katso}%
30.33   \def\alsosome{katso myös}%
30.34   \def\proofname{Proof} <-- needs translation!
30.35 }
```

`\datefinnish` The macro `\datefinnish` redefines the command `\today` to produce Finnish dates.

```
30.36 \def\datefinnish{%
30.37 \def\today{\number\day.\ifcase\month\or
30.38   tammikuuta\or helmikuuta\or maaliskuuta\or huhtikuuta\or
30.39   toukokuuta\or kesäkuuta\or heinäkuuta\or elokuuta\or
30.40   syyskuuta\or lokakuuta\or marraskuuta\or joulukuuta\fi
30.41   \space\number\year}}
```

`\extrasfinnish` 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.

```
30.42 \addto\extrasfinnish{%
30.43   \babel@savevariable\hyphenpenalty\hyphenpenalty=30%
30.44   \babel@savevariable\exhyphenpenalty\exhyphenpenalty=30%
30.45   \babel@savevariable\doublehyphendemerits\doublehyphendemerits=5000%
30.46   \babel@savevariable\finalhyphendemerits\finalhyphendemerits=5000%
30.47 }
30.48 \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.

```
30.49 \addto\extrasfinnish{\bbl@frenchspacing}
30.50 \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.

```
30.51 \initiate@active@char{"}
30.52 \addto\extrasfinnish{\languageshorthands{finnish}}
30.53 \addto\extrasfinnish{\bbl@activate{"}}
30.54 %\addto\noextrasfinnish{\bbl@deactivate{"}}
```

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

```
30.55 \addto\extrasfinnish{\umlautlow\umlautelow}
30.56 \addto\noextrasfinnish{\umlauthigh}
```

First we define access to the low opening double quote and guillemets for quotations,

```
30.57 \declare@shorthand{finnish}{"´}{%
30.58   \textormath{\quotedblbase}}{\mbox{\quotedblbase}}}
30.59 \declare@shorthand{finnish}{"’}{%
30.60   \textormath{\textquotedblright}}{\mbox{\textquotedblright}}}
30.61 \declare@shorthand{finnish}{"‹}{%
30.62   \textormath{\guillemotleft}}{\mbox{\guillemotleft}}}
30.63 \declare@shorthand{finnish}{"›}{%
30.64   \textormath{\guillemotright}}{\mbox{\guillemotright}}}
```

then we define two shorthands to be able to specify hyphenation breakpoints that behave a little different from `\-`.

```
30.65 \declare@shorthand{finnish}{"-}{\allowhyphens-\allowhyphens}
30.66 \declare@shorthand{finnish}{"z}{\hskip\z@skip}
30.67 \declare@shorthand{finnish}{"=}{\hbox{-}\allowhyphens}
```

And we want to have a shorthand for disabling a ligature.

```
30.68 \declare@shorthand{finnish}{"}{|}{%
30.69 \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.

```
30.70 \addto\extrasfinnish{\babel@save\-\}
30.71 \addto\extrasfinnish{\def\-\{\allowhyphens
30.72 \discretionary{-}{}{\allowhyphens}}
```

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

```
30.73 \def\finnishhyphenmins{\tw@\tw@}
```

It is possible that a site might need to add some extra code to the babel macros. To enable this we load a local configuration file, `finnish.cfg` if it is found on  $\TeX$ ' search path.

```
30.74 \loadlocalcfg{finnish}
```

Our last action is to make a note that the commands we have just defined, will be executed by calling the macro `\selectlanguage` at the beginning of the document.

```
30.75 \main@language{finnish}
```

Finally, the category code of `@` is reset to its original value. The macrospace used by `\atcatcode` is freed.

```
30.76 \catcode'\@=\atcatcode \let\atcatcode\relax
30.77 </code>
```

## 31 The Hungarian language

The file option `magyar.dtx`<sup>44</sup> 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 such as ‘on february 10th’.

As this file needs to be read only once, we check whether it was read before. If it was, the command `\captionsmagyar` is already defined, so we can stop processing. If this command is undefined we proceed with the various definitions and first show the current version of this file.

```
31.1 (*code)
31.2 \ifx\undefined\captionsmagyar
31.3 \else
31.4   \selectlanguage{magyar}
31.5   \expandafter\endinput
31.6 \fi
```

`\atcatcode` This file, `magyar.ldf`, may have been read while TeX is in the middle of processing a document, so we have to make sure the category code of `@` is ‘letter’ while this file is being read. We save the category code of the `@`-sign in `\atcatcode` and make it ‘letter’. Later the category code can be restored to whatever it was before.

```
31.7 \chardef\atcatcode=\catcode'\@
31.8 \catcode'\@=11\relax
```

Now we determine whether the the common macros from the file `babel.def` need to be read. We can be in one of two situations: either another language option has been read earlier on, in which case that other option has already read `babel.def`, or `magyar` is the first language option to be processed. In that case we need to read `babel.def` right here before we continue.

```
31.9 \ifx\undefined\babel@core@loaded\input babel.def\relax\fi
```

Another check that has to be made, is if another language definition file has been read already. In that case its definitions have been activated. This might interfere with definitions this file tries to make. Therefore we make sure that we cancel any special definitions. This can be done by checking the existence of the macro `\originalTeX`. If it exists we simply execute it, otherwise it is `\let` to `\empty`.

```
31.10 \ifx\undefined\originalTeX \let\originalTeX\empty \else\originalTeX\fi
```

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` to see whether we have to do something here.

```
31.11 \ifx\undefined\l@magyar
31.12   \nopatterns{Magyar}
31.13   \addialect\l@magyar0\fi
```

---

<sup>44</sup>The file described in this section has version number ? and was last revised on ?. A contribution was made by Attila Koppanyi ([attila@cernvm.cern.ch](mailto:attila@cernvm.cern.ch)). Later updates and suggestions by Árpád Bíró ([JZP1104@HUSZEG11.bitnet](mailto:JZP1104@HUSZEG11.bitnet)), Istvan Hamecz ([hami@ursus.bke.hu](mailto:hami@ursus.bke.hu)) and Horvath Dezsó ([horvath@pisa.infn.it](mailto:horvath@pisa.infn.it)).

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 accomodate 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 L<sup>A</sup>T<sub>E</sub>X.

```

31.14 \addto\captionsmagyar{%
31.15   \def\prefacename{El\H osz\'}o}%
31.16   \def\refname{Referenci\'}ak}%
31.17   \def\abstractname{Kivonat}%
31.18   \def\bibName{Bibliogr\'}afia}%
31.19   \def\chaptername{fejezet}%
31.20   \def\appendixname{f"uggel\'}ek}%
31.21   \def\contentsname{Tartalom}%
31.22   \def\listfigurename{\'}Abr\'}ak jegyz\'}eke}%
31.23   \def\listtablename{T\'}abl\'}azatok jegyz\'}eke}%
31.24   \def\indexname{T\'}argymutat\'}o}%
31.25   \def\figurename{\'}abra}%
31.26   \def\tablename{t\'}abl\'}azat}%
31.27   \def\partname{r\'}esz}%
31.28   \def\enclname{Mell\'}eklet}%
31.29   \def\ccname{K\'}orlev\'}el--c\'}\i mzettek}%
31.30   \def\headtoname{C\'}\i mzett}%
31.31   \def\pagename{oldal}%
31.32   \def\seename{L\'}asd}%
31.33   \def\alsoname{L\'}asd m\'}eg}%
31.34   \def\proofname{Proof}%    <-- needs translation
31.35 }%
```

`\datemagyar` The macro `\datemagyar` redefines the command `\today` to produce Hungarian dates.

```

31.36 \def\datemagyar{%
31.37   \def\today{\number\year.\~\ifcase\month\or
31.38   janu\'}ar\or febru\'}ar\or m\'}arcius\or
31.39   \'}aprilis\or m\'}ajus\or j\'}unius\or
31.40   j\'}ulius\or augusztus\or szeptember\or
31.41   okt\'}ober\or november\or december\fi
31.42   \space\ifcase\day\or
31.43   1.\or 2.\or 3.\or 4.\or 5.\or
31.44   6.\or 7.\or 8.\or 9.\or 10.\or
31.45   11.\or 12.\or 13.\or 14.\or 15.\or
31.46   16.\or 17.\or 18.\or 19.\or 20.\or
31.47   21.\or 22.\or 23.\or 24.\or 25.\or
31.48   26.\or 27.\or 28.\or 29.\or 30.\or
31.49   31.\fi}}
```

`\ondate` The macro `\ondate` produces Hungarian dates which have the meaning ‘*on this day*’. It does not redefine the command `\today`.

```
31.50 \def\ondate#1{%
31.51   \number\year.\ifcase\month\or
31.52   janu\ar\or febru\ar\or m\arcus\or
31.53   \aprilis\or m\ajus\or j\unius\or
31.54   j\ulius\or augusztus\or szeptember\or
31.55   okt\ober\or november\or december\fi
31.56   \space\ifcase\day\or
31.57   1-j\en\or 2-\an\or 3-\an\or 4-\en\or 5-\en\or
31.58   6-\an\or 7-\en\or 8-\an\or 9-\en\or 10-\en\or
31.59   11-\en\or 12-\en\or 13-\an\or 14-\en\or 15-\en\or
31.60   16-\an\or 17-\en\or 18-\an\or 19-\en\or 20-\an\or
31.61   21-\en\or 22-\en\or 23-\an\or 24-\en\or 25-\en\or
31.62   26-\an\or 27-\en\or 28-\an\or 29-\en\or 30-\an\or
31.63   31-\en\fi}
```

`\extrasmagyar` 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 `\ondate` is defined.

```
31.64 \addto\extrasmagyar{\let\ontoday\ondate}
31.65 \addto\noextrasmagyar{\let\ontoday\undefined}
```

It is possible that a site might need to add some extra code to the babel macros. To enable this we load a local configuration file, `magyar.cfg` if it is found on T<sub>E</sub>X search path.

```
31.66 \loadlocalcfg{magyar}
```

Our last action is to make a note that the commands we have just defined, will be executed by calling the macro `\selectlanguage` at the beginning of the document.

```
31.67 \main@language{magyar}
```

Finally, the category code of `@` is reset to its original value. The macrospace used by `\atcatcode` is freed.

```
31.68 \catcode'\@=\atcatcode \let\atcatcode\relax
31.69 </code>
```

## 32 The Estonian language

The file `estonian.dtx`<sup>45</sup> 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 13. 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 13: 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`.

### 32.1 Implementation

Check whether the file has been read already.

32.1 `<*`

32.2 `\ifx\undefined\captionsestonian`

---

<sup>45</sup>The file described in this section has version number ? and was last revised on ?. The original author is Enn Saar, (`saar@aai.ee`).

```

32.3 \else
32.4 \selectlanguage{estonian}
32.5 \expandafter\endinput
32.6 \fi

```

Change the category code of @, as usual.

```

32.7 \chardef\atcatcode=\catcode'\@
32.8 \catcode'\@=11\relax

```

Check whether we need the common macros from the file babel.def.

```

32.9 \ifx\undefined\babel@core@loaded\input babel.def\relax\fi

```

We execute the macro `\originalTeX` to get rid of side effects that could be caused by language options used before.

```

32.10 \ifx\undefined\originalTeX \let\originalTeX\empty \else\originalTeX\fi

```

If Estonian is not included in the format file (does not have hyphenation patterns), we shall use English hyphenation.

```

32.11 \ifx\undefined\l@estonian
32.12 \@nopatterns{Estonian}
32.13 \adddialect\l@estonian0
32.14 \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 L<sup>A</sup>T<sub>E</sub>X.

```

32.15 \addto\captionsestonian{%
32.16 \def\prefacename{Sissejuhatus}%
32.17 \def\refname{Viited}%
32.18 \def\bibname{Kirjandus}%
32.19 \def\appendixname{Lisa}%
32.20 \def\contentsname{Sisukord}%
32.21 \def\listfigurename{Joonised}%
32.22 \def\listtablename{Tabelid}%
32.23 \def\indexname{Indeks}%
32.24 \def\figurename{Joonis}%
32.25 \def\tablename{Tabel}%
32.26 \def\partname{Osa}%
32.27 \def\enclname{Lisa(d)}%
32.28 \def\ccname{Koopia(d)}%
32.29 \def\headtoname{}%
32.30 \def\pagename{Lk.}%
32.31 \def\seename{vt.}%
32.32 \def\alsoname{vt. ka}
32.33 \def\proofname{Proof}% <-- needs translation
32.34 }

```

These captions contain accented characters.

```

32.35 \begingroup \catcode'\\"active
32.36 \def\x{\endgroup
32.37 \addto\captionsestonian{%
32.38 \def\abstractname{Kokkuvõte}%
32.39 \def\chaptername{Peat"ukk}}
32.40 \x

```

`\dateestonian` The macro `\dateestonian` redefines the command `\today` to produce Estonian dates.

```
32.41 \begingroup \catcode'\\"active
32.42 \def\x{\endgroup
32.43   \def\month@estonian{\ifcase\month\or
32.44     jaanuar\or veebruar\or m"arts\or aprill\or mai\or juuni\or
32.45     juuli\or august\or september\or oktoober\or november\or
32.46     detsember\fi}}
32.47 \x
32.48 \def\dateestonian{\def\today{\number\day.\space\month@estonian
32.49 \space\number\year.\space a.}}
```

`\extrasestonian` 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).

```
32.50 \initiate@active@char{"}
32.51 \initiate@active@char{~}
32.52 \addto\extrasestonian{\languageshorthands{estonian}}
32.53 \addto\extrasestonian{\bbl@activate{"}\bbl@activate{~}}
```

Store the original macros, and redefine accents.

```
32.54 \addto\extrasestonian{\babel@save"\umlautlow\babel@save"\~\tilde\low}
```

If we are using the T1 output encoding, we should allow for the T1 input encoding, too (this has been chosen as the preliminary archiving standard by TWGML).

```
32.55 \edef\next{T1}
32.56 \ifx\f@encoding\next
32.57   \addto\extrasestonian{%
32.58     \catcode245=11 \catcode228=11 \catcode246=11 \catcode252=11
32.59     \catcode178=11 \catcode186=11 \catcode213=11 \catcode196=11
32.60     \catcode214=11 \catcode220=11 \catcode146=11 \catcode154=11
32.61     \lccode245=245 \lccode228=228 \lccode246=246 \lccode252=252
32.62     \lccode178=178 \lccode186=186 \lccode213=245 \lccode196=228
32.63     \lccode214=246 \lccode220=252 \lccode146=178 \lccode154=186
32.64     \uccode245=213 \uccode228=196 \uccode246=214 \uccode252=220
32.65     \uccode178=146 \uccode186=154 \uccode213=213 \uccode196=196
32.66     \uccode214=214 \uccode220=220 \uccode146=146 \uccode154=154
32.67     \sfcode245=1000 \sfcode228=1000 \sfcode246=1000 \sfcode252=1000
32.68     \sfcode178=1000 \sfcode186=1000 \sfcode213=999 \sfcode196=999
32.69     \sfcode214=999 \sfcode220=999 \sfcode146=999 \sfcode154=999}
32.70 \fi
```

Estonian does not use extra spaces after sentences.

```
32.71 \addto\extrasestonian{\bbl@frenchspacing}
32.72 \addto\noextrasestonian{\bbl@nonfrenchspacing}
```

`\estonianhyphenmins` For Estonian, `\lefthyphenmin` and `\righthyphenmin` are both 2.

```
32.73 \def\estonianhyphenmins{\tw@\tw@}
```

`\tildelow` The standard T<sub>E</sub>X accents are too high for Estonian typography, we have to lower  
`\gentilde` them (following the babel German style). For a detailed explanation see the file  
`\newtilde` `glyphs.dtx`.

```

\newcheck 32.74 \def\tildelow{\def\~{\protect\gentilde}}
          32.75 \def\gentilde#1{\if#1o\newtilde{#1}\else\if#1O\newtilde{#1}%
          32.76   \else\newcheck{#1}%
          32.77   \fi\fi}
          32.78 \def\newtilde#1{\leavevmode\allowhyphens
          32.79   {\U@D 1ex%
          32.80   {\setbox\z@\hbox{\char126}\dimen@ -.45ex\advance\dimen@\ht\z@
          32.81   \ifdim 1ex<\dimen@ \fontdimen5\font\dimen@ \fi}%
          32.82   \accent126\fontdimen5\font\U@D #1}\allowhyphens}
          32.83 \def\newcheck#1{\leavevmode\allowhyphens
          32.84   {\U@D 1ex%
          32.85   {\setbox\z@\hbox{\char20}\dimen@ -.45ex\advance\dimen@\ht\z@
          32.86   \ifdim 1ex<\dimen@ \fontdimen5\font\dimen@ \fi}%
          32.87   \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`.

```

32.88 \begingroup \catcode'\ "12
32.89 \edef\x{\endgroup
32.90   \def\noexpand\dq{"}
32.91   \def\noexpand\til{~}}
32.92 \x
32.93 \let\dieresis\"
32.94 \let\texttilde\~

```

This part follows closely `spanish.ldf`. We check the encoding and if it is T1, we have to tell T<sub>E</sub>X about our redefined accents.

```

32.95 \edef\next{T1}
32.96 \ifx\fontencoding\next
32.97   \let\@umlaut\dieresis
32.98   \let\@tilde\texttilde
32.99   \DeclareTextComposite{\~}{T1}{s}{178}
32.100  \DeclareTextComposite{\~}{T1}{S}{146}
32.101  \DeclareTextComposite{\~}{T1}{z}{186}
32.102  \DeclareTextComposite{\~}{T1}{Z}{154}
32.103  \DeclareTextComposite{\"}{T1}{'}{17}
32.104  \DeclareTextComposite{\"}{T1}{'}{18}
32.105  \DeclareTextComposite{\"}{T1}{<}{19}
32.106  \DeclareTextComposite{\"}{T1}{>}{20}

```

If the encoding differs from T1, we expand the accents, enabling hyphenation beyond the accent. In this case T<sub>E</sub>X will not find all possible breaks, and we have to warn people.

```

32.107 \else
32.108   \wlog{Warning: Hyphenation would work better for the T1 encoding.}
32.109   \let\@umlaut\newumlaut
32.110   \let\@tilde\gentilde
32.111 \fi

```

Now we define the shorthands.

```

32.112 \declare@shorthand{estonian}{\textormath{"{a}}{\ddot a}}
32.113 \declare@shorthand{estonian}{\textormath{"{A}}{\ddot A}}
32.114 \declare@shorthand{estonian}{\textormath{"{o}}{\ddot o}}
32.115 \declare@shorthand{estonian}{\textormath{"{O}}{\ddot O}}
32.116 \declare@shorthand{estonian}{\textormath{"{u}}{\ddot u}}
32.117 \declare@shorthand{estonian}{\textormath{"{U}}{\ddot U}}

    german and french quotes,
32.118 \declare@shorthand{estonian}{" '}{%
32.119 \textormath{\quotedblbase}}{\mbox{\quotedblbase}}
32.120 \declare@shorthand{estonian}{" '}{%
32.121 \textormath{\textquotedblleft}}{\mbox{\textquotedblleft}}
32.122 \declare@shorthand{estonian}{"<}{%
32.123 \textormath{\guillemotleft}}{\mbox{\guillemotleft}}
32.124 \declare@shorthand{estonian}{">}{%
32.125 \textormath{\guillemotright}}{\mbox{\guillemotright}}

32.126 \declare@shorthand{estonian}{~o}{\textormath{@tilde o}{\tilde o}}
32.127 \declare@shorthand{estonian}{~O}{\textormath{@tilde O}{\tilde O}}
32.128 \declare@shorthand{estonian}{~s}{\textormath{@tilde s}{\check s}}
32.129 \declare@shorthand{estonian}{~S}{\textormath{@tilde S}{\check S}}
32.130 \declare@shorthand{estonian}{~z}{\textormath{@tilde z}{\check z}}
32.131 \declare@shorthand{estonian}{~Z}{\textormath{@tilde Z}{\check Z}}

    and some additional commands:
32.132 \declare@shorthand{estonian}{-}{\allowhyphens\-\allowhyphens}
32.133 \declare@shorthand{estonian}{"|}{%
32.134 \textormath{\penalty\M\discretionary{-}{\kern.03em}%
32.135 \allowhyphens}{}}
32.136 \declare@shorthand{estonian}{""}{\dq}
32.137 \declare@shorthand{estonian}{~~}{\til}

    It is possible that a site might need to add some extra code to the babel macros.
    To enable this we load a local configuration file, estonian.cfg if it is found on
    TeX' search path.
32.138 \loadlocalcfg{estonian}

    Select, finally, Estonian, and restore the category code of @. We are done.
32.139 \main@language{estonian}
32.140 \catcode'\@=\atcatcode \let\atcatcode\relax
32.141 \code>

```

## 33 The Croatian language

The file `croatian.dtx`<sup>46</sup> defines all the language definition macros for the Croatian language.

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

As this file needs to be read only once, we check whether it was read before. If it was, the command `\captionscroatian` is already defined, so we can stop processing. If this command is undefined we proceed with the various definitions and first show the current version of this file.

```
33.1 <*code>
33.2 \ifx\undefined\captionscroatian
33.3 \else
33.4   \selectlanguage{croatian}
33.5   \expandafter\endinput
33.6 \fi
```

`\atcatcode` This file, `croatian.sty`, may have been read while TeX is in the middle of processing a document, so we have to make sure the category code of `@` is ‘letter’ while this file is being read. We save the category code of the `@`-sign in `\atcatcode` and make it ‘letter’. Later the category code can be restored to whatever it was before.

```
33.7 \chardef\atcatcode=\catcode'\@
33.8 \catcode'\@=11\relax
```

Now we determine whether the the common macros from the file `babel.def` need to be read. We can be in one of two situations: either another language option has been read earlier on, in which case that other option has already read `babel.def`, or `croatian` is the first language option to be processed. In that case we need to read `babel.def` right here before we continue.

```
33.9 \ifx\undefined\babel@core@loaded\input babel.def\relax\fi
```

Another check that has to be made, is if another language definition file has been read already. In that case its definitions have been activated. This might interfere with definitions this file tries to make. Therefore we make sure that we cancel any special definitions. This can be done by checking the existence of the macro `\originalTeX`. If it exists we simply execute it, otherwise it is `\let` to `\empty`.

```
33.10 \ifx\undefined\originalTeX \let\originalTeX\empty \else\originalTeX\fi
```

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.

```
33.11 \ifx\undefined\l@croatian
33.12   \nopatterns{Croatian}
33.13   \adddialect\l@croatian0\fi
```

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

---

<sup>46</sup>The file described in this section has version number ? and was last revised on ?. A contribution was made by Alan Paić (`paica@cernvm.cern.ch`).

`\captionscroatian` The macro `\captionscroatian` defines all strings used in the four standard documentclasses provided with L<sup>A</sup>T<sub>E</sub>X.

```
33.14 \addto\captionscroatian{%
33.15   \def\prefacename{Predgovor}%
33.16   \def\refname{Literatura}%
33.17   \def\abstractname{Sa\v{z}etak}%
33.18   \def\bibname{Bibliografija}%
33.19   \def\chaptername{Glava}%
33.20   \def\appendixname{Dodatak}%
33.21   \def\contentsname{Sadr\v{z}aj}%
33.22   \def\listfigurename{Slike}%
33.23   \def\listtablename{Tablice}%
33.24   \def\indexname{Indeks}%
33.25   \def\figurename{Slika}%
33.26   \def\tablename{Tablica}%
33.27   \def\partname{Dio}%
33.28   \def\enclname{Prilozi}%
33.29   \def\ccname{Kopije}%
33.30   \def\headtoname{Prima}%
33.31   \def\pagename{Strana}%
33.32   \def\seename{Vidi}%
33.33   \def\alsoname{Vidi tako\dj er}%
33.34   \def\proofname{Proof}% <-- needs translation
33.35 }%
```

`\datecroatian` The macro `\datecroatian` redefines the command `\today` to produce Croatian dates.

```
33.36 \def\datecroatian{%
33.37 \def\today{\number\day .~\ifcase\month\or
33.38   sije\v{c}anj\or velja\v{c}a\or o\v{z}ujak\or travanj\or svibanj\or
33.39   lipanj\or srpanj\or kolovoz\or rujan\or listopad\or studeni\or
33.40   prosinac\fi
33.41   \space \number\year}}
```

`\extrascroatian` 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.

```
33.42 \addto\extrascroatian{}
33.43 \addto\noextrascroatian{}
```

It is possible that a site might need to add some extra code to the babel macros. To enable this we load a local configuration file, `croatian.cfg` if it is found on T<sub>E</sub>X' search path.

```
33.44 \loadlocalcfg{croatian}
```

Our last action is to make a note that the commands we have just defined, will be executed by calling the macro `\selectlanguage` at the beginning of the document.

```
33.45 \main@language{croatian}
```

Finally, the category code of `@` is reset to its original value. The macrospace used by `\atcatcode` is freed.

```
33.46 \catcode'\@=\atcatcode \let\atcatcode\relax
33.47 \end{code}
```

## 34 The Czech language

The file `czech.dtx`<sup>47</sup> 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 ř. The command `\w` is used to put the ring-accent which appears in ångström over the letters u and U.

As this file needs to be read only once, we check whether it was read before. If it was, the command `\captionsczech` is already defined, so we can stop processing. If this command is undefined we proceed with the various definitions and first show the current version of this file.

```
34.1 (*code)
34.2 \ifx\undefined\captionsczech
34.3 \else
34.4   \selectlanguage{czech}
34.5   \expandafter\endinput
34.6 \fi
```

`\atcatcode` This file, `czech.sty`, may have been read while  $\TeX$  is in the middle of processing a document, so we have to make sure the category code of `@` is 'letter' while this file is being read. We save the category code of the `@`-sign in `\atcatcode` and make it 'letter'. Later the category code can be restored to whatever it was before.

```
34.7 \chardef\atcatcode=\catcode'\@
34.8 \catcode'\@=11\relax
```

Now we determine whether the the common macros from the file `babel.def` need to be read. We can be in one of two situations: either another language option has been read earlier on, in which case that other option has already read `babel.def`, or `czech` is the first language option to be processed. In that case we need to read `babel.def` right here before we continue.

```
34.9 \ifx\undefined\babel@core@loaded\input babel.def\relax\fi
```

Another check that has to be made, is if another language definition file has been read already. In that case its definitions have been activated. This might interfere with definitions this file tries to make. Therefore we make sure that we cancel any special definitions. This can be done by checking the existence of the macro `\originalTeX`. If it exists we simply execute it, otherwise it is `\let` to `\empty`.

```
34.10 \ifx\undefined\originalTeX \let\originalTeX\empty \else\originalTeX\fi
```

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.

```
34.11 \ifx\undefined\l@czech
34.12   \nopatterns{Czech}
34.13   \adddialect\l@czech0\fi
```

---

<sup>47</sup>The file described in this section has version number ? and was last revised on ?. Contributions were made by Milos Lokajicek (LOKAJICK@CERNVM).

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 L<sup>A</sup>T<sub>E</sub>X.

```

34.14 \addto\captionsczech{%
34.15   \def\prefacename{P\v redmluva}%
34.16   \def\refname{Reference}%
34.17   \def\abstractname{Abstrakt}%
34.18   \def\bibname{Literatura}%
34.19   \def\chaptername{Kapitola}%
34.20   \def\appendixname{Dodatek}%
34.21   \def\contentsname{Obsah}%
34.22   \def\listfigurename{Seznam obr\'azk\u}%
34.23   \def\listtablename{Seznam tabulek}%
34.24   \def\indexname{Index}%
34.25   \def\figurename{Obr\'azek}%
34.26   \def\tablename{Tabulka}%
34.27   \def\partname{\v{C}\'}ast}%
34.28   \def\enclname{P\v{r}\'}{i}loha}%
34.29   \def\ccname{cc}%
34.30   \def\headtoname{Komu}%
34.31   \def\pagename{Strana}%
34.32   \def\seename{viz}%
34.33   \def\alsiname{viz tek\'e}%
34.34   \def\proofname{Proof}%    <-- needs translation
34.35 }%
```

`\dateczech` The macro `\dateczech` redefines the command `\today` to produce Czech dates.

```

34.36 \def\dateczech{%
34.37   \def\today{\number\day.~\ifcase\month\or
34.38     ledna\or \\'unora\or b\v{r}ezna\or dubna\or kv\v{e}tna\or \v{c}ervna\or
34.39     \v{c}ervence\or srpna\or z\'a\v{r}\'}{i}\or \v{r}\'}{i}jna\or
34.40     listopadu\or prosince\fi
34.41   \space \number\year}}
```

`\extrasczech` 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.

```

34.42 \addto\extrasczech{\babel@save\q\let\q\v}
34.43 \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.

```

34.44 \addto\extrasczech{\bbl@frenchspacing}
34.45 \addto\noextrasczech{\bbl@nonfrenchspacing}
```

`\v` L<sup>A</sup>T<sub>E</sub>X's normal `\v` accent places a caron over the letter that follows it (ö). This is not what we want for the letters d, t, l and L; for those the accent should change shape. This is achieved by the following.

```

34.46 \AtBeginDocument{%
34.47   \DeclareTextCompositeCommand{\v}{OT1}{t}{t}{%
```

```

34.48   t\kern-.23em\raise.24ex\hbox{'}}
34.49   \DeclareTextCompositeCommand{\v}{OT1}{d}{%
34.50     d\kern-.13em\raise.24ex\hbox{'}}
34.51   \DeclareTextCompositeCommand{\v}{OT1}{l}{\lcaron{}}
34.52   \DeclareTextCompositeCommand{\v}{OT1}{L}{\Lcaron{}}

```

`\lcaron` For the letters l and L we want to distinguish between normal fonts and monospaced fonts.

```

34.53 \def\lcaron{%
34.54   \setbox0\hbox{M}\setbox\tw@\hbox{i}%
34.55   \ifdim\wd0>\wd\tw@\relax
34.56     l\kern-.13em\raise.24ex\hbox{'}\kern-.11em%
34.57   \else
34.58     l\raise.45ex\hbox to\z@{\kern-.35em '\hss}%
34.59   \fi}
34.60 \def\Lcaron{%
34.61   \setbox0\hbox{M}\setbox\tw@\hbox{i}%
34.62   \ifdim\wd0>\wd\tw@\relax
34.63     L\raise.24ex\hbox to\z@{\kern-.28em'\hss}%
34.64   \else
34.65     L\raise.45ex\hbox to\z@{\kern-.40em '\hss}%
34.66   \fi}

```

It is possible that a site might need to add some extra code to the babel macros. To enable this we load a local configuration file, `czech.cfg` if it is found on T<sub>E</sub>X' search path.

```
34.67 \loadlocalcfg{czech}
```

Our last action is to make a note that the commands we have just defined, will be executed by calling the macro `\selectlanguage` at the beginning of the document.

```
34.68 \main@language{czech}
```

Finally, the category code of `@` is reset to its original value. The macrospace used by `\atcatcode` is freed.

```

34.69 \catcode'\@=\atcatcode \let\atcatcode\relax
34.70 </code>

```

## 35 The Polish language

The file `polish.dtx`<sup>48</sup> defines all the language-specific macros for the Polish language.

For this language the character " is made active. In table 14 an overview is given of its purpose.

"a	or <code>\aob</code> , for tailed-a (like <i>ą</i> )
"A	or <code>\Aob</code> , for tailed-A (like <i>Ą</i> )
"e	or <code>\eob</code> , for tailed-e (like <i>ę</i> )
"E	or <code>\Eob</code> , for tailed-E (like <i>Ę</i> )
"c	or <code>\'c</code> , for accented c (like <i>ć</i> ), same with uppercase letters and n,o,s
"l	or <code>\lpb{}</code> , for l with stroke (like <i>ł</i> )
"L	or <code>\Lpb{}</code> , for L with stroke (like <i>Ł</i> )
"r	or <code>\zkb{}</code> , for pointed z (like <i>ź</i> ), cf. pronunciation
"R	or <code>\Zkb{}</code> , for pointed Z (like <i>Ź</i> )
"z	or <code>\'z</code> , for accented z
"Z	or <code>\'Z</code> , 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 compound words with hyphen, e.g. <i>x-"y</i> ).
"‘	for German left double quotes (looks like <i>„</i> ).
"’	for German right double quotes.
"<	for French left double quotes (similar to <i>&lt;&lt;</i> ).
">	for French right double quotes (similar to <i>&gt;&gt;</i> ).

Table 14: The extra definitions made by `polish.sty`

As this file needs to be read only once, we check whether it was read before. If it was, the command `\captionspolish` is already defined, so we can stop processing. If this command is undefined we proceed with the various definitions and first show the current version of this file.

```
35.1 \ifx\undefined\captionspolish
35.2 \else
35.3   \selectlanguage{polish}
35.4   \expandafter\endinput
35.5 \fi
```

`\atcatcode` This file, `polish.sty`, may have been read while  $\TeX$  is in the middle of processing a document, so we have to make sure the category code of `@` is 'letter' while this file is being read. We save the category code of the `@`-sign in `\atcatcode` and make it 'letter'. Later the category code can be restored to whatever it was before.

```
35.6 (*code)
35.7 \chardef\atcatcode=\catcode'\@
35.8 \catcode'\@=11\relax
```

---

<sup>48</sup>The file described in this section has version number ? and was last revised on ?.

Now we determine whether the the common macros from the file `babel.def` need to be read. We can be in one of two situations: either another language option has been read earlier on, in which case that other option has already read `babel.def`, or `polish` is the first language option to be processed. In that case we need to read `babel.def` right here before we continue.

```
35.9 \ifx\undefined\babel@core@loaded\input babel.def\relax\fi
```

Another check that has to be made, is if another language definition file has been read already. In that case its definitions have been activated. This might interfere with definitions this file tries to make. Therefore we make sure that we cancel any special definitions. This can be done by checking the existence of the macro `\originalTeX`. If it exists we simply execute it.

```
35.10 \ifx\undefined\originalTeX
```

```
35.11 \let\originalTeX\empty
```

```
35.12 \fi
```

```
35.13 \originalTeX
```

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.

```
35.14 \ifx\undefined\l@polish
```

```
35.15 \nopatterns{Polish}
```

```
35.16 \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 L<sup>A</sup>T<sub>E</sub>X.

```
35.17 \addto\captionspolish{%
35.18 \def\prefacename{Przedmowa}%
35.19 \def\refname{Bibliografia}%
35.20 \def\abstractname{Streszczenie}%
35.21 \def\bibName{Literatura}%
35.22 \def\chaptername{Rozdzia\l}%
35.23 \def\appendixname{Dodatek}%
35.24 \def\contentsname{Spis rzeczy}%
35.25 \def\listfigurename{Spis rysunk\ow}%
35.26 \def\listtablename{Spis tablic}%
35.27 \def\indexname{Indeks}%
35.28 \def\figurename{Rysunek}%
35.29 \def\tablename{Tablica}%
35.30 \def\partname{Cz\eob}\s\c}%
35.31 \def\enclname{Za\l\ aob\cznik}%
35.32 \def\ccname{Kopie:}%
35.33 \def\headtoname{Do}%
35.34 \def\pagename{Strona}%
35.35 \def\seename{Por\ownaj}%
35.36 \def\alsoname{Por\ownaj tak\ze}%
35.37 \def\proofname{Proof}% <-- needs translation
35.38 }
```

`\datepolish` The macro `\datepolish` redefines the command `\today` to produce Polish dates.

```

35.39 \def\datepolish{%
35.40   \def\today{\number\day~\ifcase\month\or
35.41   stycznia\or lutego\or marca\or kwietnia\or maja\or czerwca\or lipca\or
35.42   sierpnia\or wrze\'snia\or pa\'zdziernika\or listopada\or grudnia\fi
35.43   \space\number\year}
35.44 }

```

`\extrapolish` The macro `\extrapolish` will perform all the extra definitions needed for the Polish language. The macro `\noextrapolish` is used to cancel the actions of `\extrapolish`.

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.

```

35.45 \initiate@active@char{"}
35.46 \addto\extrapolish{\languageshorthands{polish}}
35.47 \addto\extrapolish{\bbl@activate{"}}
35.48 %\addto\noextrapolish{\bbl@deactivate{"}}

```

The code above is necessary because we need an extra active character. This character is then used as indicated in table 14.

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 forecasting 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.

```

35.49 \newdimen\pl@left
35.50 \newdimen\pl@down
35.51 \newdimen\pl@right
35.52 \newdimen\pl@temp

```

`\sob` The macro `\sob` is used to put the ‘ogonek’ in the right place.

```

35.53 \def\sob#1#2#3#4#5{%parameters: letter and fractions hl,ho,vl,vo
35.54   \setbox0\hbox{#1}\setbox1\hbox{$_\mathchar'454$}\setbox2\hbox{p}%
35.55   \pl@right=#2\wd0 \advance\pl@right by-#3\wd1
35.56   \pl@down=#5\ht1 \advance\pl@down by-#4\ht0
35.57   \pl@left=\pl@right \advance\pl@left by\wd1
35.58   \pl@temp=-\pl@down \advance\pl@temp by\dp2 \dp1=\pl@temp
35.59   \kern\pl@right\lower\pl@down\box1\kern-\pl@left #1}

```

`\aob` The ogonek is placed with the letters ‘a’, ‘A’, ‘e’, and ‘E’.

```

\Aob35.60 \def\aob{\sob a{.66}{.20}{0}{.90}}
\eob35.61 \def\Aob{\sob A{.80}{.50}{0}{.90}}
\Eob35.62 \def\eob{\sob e{.50}{.35}{0}{.93}}
35.63 \def\Eob{\sob E{.60}{.35}{0}{.90}}

```

`\spb` The macro `\spb` is used to put the ‘poprzeczka’ in the right place.

```

35.64 \def\spb#1#2#3#4#5{%
35.65   \setbox0\hbox{#1}\setbox1\hbox{\char'023}%

```

```

35.66 \pl@right=#2\wd0 \advance\pl@right by-#3\wd1
35.67 \pl@down=#5\ht1 \advance\pl@down by-#4\ht0
35.68 \pl@left=\pl@right \advance\pl@left by\wd1
35.69 \ht1=\pl@down \dp1=-\pl@down
35.70 \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.

```

35.71 \def\skb#1#2#3#4#5{%
35.72 \setbox0\hbox{#1}\setbox1\hbox{\char'056}%
35.73 \pl@right=#2\wd0 \advance\pl@right by-#3\wd1
35.74 \pl@down=#5\ht1 \advance\pl@down by-#4\ht0
35.75 \pl@left=\pl@right \advance\pl@left by\wd1
35.76 \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.

```

35.77 \def\textpl{%
35.78 \def\lpb{\p111}%
35.79 \def\Lpb{\pLLL}%
35.80 \def\zkb{\.z}%
35.81 \def\Zkb{\.Z}}

```

Initially we assume that typesetting is done with text fonts.

```

35.82 \textpl
35.83 \let\111=\1 \let\LLL=\L
35.84 \def\p111{\111}
35.85 \def\pLLL{\LLL}

```

`\telepl` But for the ‘teletype’ font in ‘OT1’ encoding we have to take some special actions, involving the macros defined above.

```

35.86 \def\telepl{%
35.87 \def\lpb{\spb 1{.45}{.5}{.4}{.8}}%
35.88 \def\Lpb{\spb L{.23}{.5}{.4}{.8}}%
35.89 \def\zkb{\skb z{.5}{.5}{1.2}{0}}%
35.90 \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 supposed to be either plain (based) or  $\LaTeX$  2.09.

```

35.91 \ifx\selectfont\undefined
35.92 \ifx\prm\undefined \addto\rm{\textpl}\else \addto\prm{\textpl}\fi
35.93 \ifx\pit\undefined \addto\it{\textpl}\else \addto\pit{\textpl}\fi
35.94 \ifx\pbf\undefined \addto\bf{\textpl}\else \addto\pbf{\textpl}\fi
35.95 \ifx\psl\undefined \addto\sl{\textpl}\else \addto\psl{\textpl}\fi
35.96 \ifx\psf\undefined \else \addto\psf{\textpl}\fi
35.97 \ifx\psc\undefined \else \addto\psc{\textpl}\fi
35.98 \ifx\ptt\undefined \addto\tt{\telepl}\else \addto\ptt{\telepl}\fi
35.99 \else

```

When `\selectfont` exists we assume  $\LaTeX$  2 $\epsilon$ .

```

35.100 \expandafter\addto\csname selectfont \endcsname{%
35.101 \csname\fontencoding @pl\endcsname}
35.102 \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.

```
35.103 \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.

```
35.104 \expandafter\def\csname OT1@pl\endcsname{%
35.105   \long\edef\curr@family{\f@family}%
35.106   \ifx\curr@family\ttdefault
35.107     \telepl
35.108   \else
35.109     \textpl
35.110   \fi}
```

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

```
35.111 \begingroup \catcode'\`"12
35.112 \def\x{\endgroup
35.113   \def\dq{"}}
35.114 \x
```

Now we can define the doublequote macros for diacritics,

```
35.115 \declare@shorthand{polish}{"a}{\textormath{\aob}{\ddot a}}
35.116 \declare@shorthand{polish}{"A}{\textormath{\Aob}{\ddot A}}
35.117 \declare@shorthand{polish}{"c}{\textormath{\`c}{\acute c}}
35.118 \declare@shorthand{polish}{"C}{\textormath{\`C}{\acute C}}
35.119 \declare@shorthand{polish}{"e}{\textormath{\eob}{\ddot e}}
35.120 \declare@shorthand{polish}{"E}{\textormath{\Eob}{\ddot E}}
35.121 \declare@shorthand{polish}{"l}{\textormath{\lpb}{\ddot l}}
35.122 \declare@shorthand{polish}{"L}{\textormath{\Lpb}{\ddot L}}
35.123 \declare@shorthand{polish}{"n}{\textormath{\`n}{\acute n}}
35.124 \declare@shorthand{polish}{"N}{\textormath{\`N}{\acute N}}
35.125 \declare@shorthand{polish}{"o}{\textormath{\`o}{\acute o}}
35.126 \declare@shorthand{polish}{"O}{\textormath{\`O}{\acute O}}
35.127 \declare@shorthand{polish}{"r}{\textormath{\zkb}{\ddot r}}
35.128 \declare@shorthand{polish}{"R}{\textormath{\Zkb}{\ddot R}}
35.129 \declare@shorthand{polish}{"s}{\textormath{\`s}{\acute s}}
35.130 \declare@shorthand{polish}{"S}{\textormath{\`S}{\acute S}}
35.131 \declare@shorthand{polish}{"z}{\textormath{\`z}{\acute z}}
35.132 \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.

```
35.133 \declare@shorthand{polish}{"' }{%
35.134   \textormath{\quotedblbase}}{\mbox{\quotedblbase}}
35.135 \declare@shorthand{polish}{"' }{%
35.136   \textormath{\textquotedblleft}}{\mbox{\textquotedblleft}}
35.137 \declare@shorthand{polish}{"< }{%
35.138   \textormath{\guillemotleft}}{\mbox{\guillemotleft}}
35.139 \declare@shorthand{polish}{"> }{%
35.140   \textormath{\guillemotright}}{\mbox{\guillemotright}}
```

then we define two shorthands to be able to specify hyphenation breakpoints that behave a little different from `\-`.

```

35.141 \declare@shorthand{polish}{-}{\allowhyphens-\allowhyphens}
35.142 \declare@shorthand{polish}{"}{\hskip\z@skip}

```

And we want to have a shorthand for disabling a ligature.

```

35.143 \declare@shorthand{polish}{|}{%
35.144 \textormath{\discretionary{-}{-}{\kern.03em}}{}}

```

`\mdqon` All that's left to do now is to define a couple of commands for reasons of compatibility with `polish.tex`.

```

35.145 \def\mdqon{\bbl@activate{}}
35.146 \def\mdqoff{\bbl@deactivate{}}

```

It is possible that a site might need to add some extra code to the babel macros. To enable this we load a local configuration file, `polish.cfg` if it is found on T<sub>E</sub>X' search path.

```

35.147 \loadlocalcfg{polish}

```

Our last action is to make a note that activate the commands we have just defined, will be executed by calling the macro `\selectlanguage` at the beginning of the document.

```

35.148 \main@language{polish}

```

Finally, the category code of `@` is reset to its original value. The macrospace used by `\atcatcode` is freed.

```

35.149 \catcode'\@=\atcatcode \let\atcatcode\relax
35.150 </code>

```

## 36 The Slovak language

The file `slovak.dtx`<sup>49</sup> 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`.

As this file needs to be read only once, we check whether it was read before. If it was, the command `\captionsslovak` is already defined, so we can stop processing. If this command is undefined we proceed with the various definitions and first show the current version of this file.

```
36.1 (*code)
36.2 \ifx\undefined\captionsslovak
36.3 \else
36.4   \selectlanguage{slovak}
36.5   \expandafter\endinput
36.6 \fi
```

`\atcatcode` This file, `slovak.sty`, may have been read while  $\text{T}_{\text{E}}\text{X}$  is in the middle of processing a document, so we have to make sure the category code of `@` is ‘letter’ while this file is being read. We save the category code of the `@`-sign in `\atcatcode` and make it ‘letter’. Later the category code can be restored to whatever it was before.

```
36.7 \chardef\atcatcode=\catcode'\@
36.8 \catcode'\@=11\relax
```

Now we determine whether the the common macros from the file `babel.def` need to be read. We can be in one of two situations: either another language option has been read earlier on, in which case that other option has already read `babel.def`, or `slovak` is the first language option to be processed. In that case we need to read `babel.def` right here before we continue.

```
36.9 \ifx\undefined\babel@core@loaded\input babel.def\relax\fi
```

Another check that has to be made, is if another language definition file has been read already. In that case its definitions have been activated. This might interfere with definitions this file tries to make. Therefore we make sure that we cancel any special definitions. This can be done by checking the existence of the macro `\originalTeX`. If it exists we simply execute it, otherwise it is `\let` to `\empty`.

```
36.10 \ifx\undefined\originalTeX \let\originalTeX\empty \else\originalTeX\fi
```

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.

```
36.11 \ifx\undefined\l@slovak
36.12   \nopatterns{Slovak}
36.13   \addialect\l@slovak0\fi
```

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

---

<sup>49</sup>The file described in this section has version number ? and was last revised on ?. It was written by Jana Chlebková (`chlebk@euromath.dk`).

`\captionsslovak` The macro `\captionsslovak` defines all strings used in the four standard documentclasses provided with L<sup>A</sup>T<sub>E</sub>X.

```
36.14 \addto\captionsslovak{%
36.15   \def\prefacename{\'Uvod}%
36.16   \def\refname{Referencia}%
36.17   \def\abstractname{Abstrakt}%
36.18   \def\bibName{Literat\'ura}%
36.19   \def\chaptername{Kapitola}%
36.20   \def\appendixname{Dodatok}%
36.21   \def\contentsname{Obsah}%
36.22   \def\listfigurename{Zoznam obr\'azkov}%
36.23   \def\listtablename{Zoznam tabuliek}%
36.24   \def\indexname{Index}%
36.25   \def\figurename{Obr\'azok}%
36.26   \def\tablename{Tabu\lka}%% special letter l with hook
36.27   \def\partname{\v{C}as\ t}%% special letter t with hook
36.28   \def\enclname{Pr\'{i}loha}%
36.29   \def\ccname{CC}%
36.30   \def\headtoname{Komu}%
36.31   \def\pagename{Strana}%
36.32   \def\seenname{vi\ d}%% Special letter d with hook
36.33   \def\alsoname{vi\ d tie\ v z}%% Special letter d with hook
36.34   \def\proofname{Proof}% <-- needs translation
36.35 }
```

`\dateslovak` The macro `\dateslovak` redefines the command `\today` to produce Slovak dates.

```
36.36 \def\dateslovak{%
36.37 \def\today{\number\day.\~\ifcase\month\or
36.38 janu\'ara\or febru\'ara\or marca\or apr\'{i}la\or m\'aja\or j\'una\or
36.39 j\'ula\or august\or septembra\or okt\'obra\or
36.40 novembra\or decembra\fi
36.41   \space \number\year}}
```

`\extrasslovak` 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.

```
36.42 \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.

```
36.43 \def\slovakhyphenmins{\tw@\tw@}
```

It is possible that a site might need to add some extra code to the babel macros. To enable this we load a local configuration file, `slovak.cfg` if it is found on T<sub>E</sub>X' search path.

```
36.44 \loadlocalcfg{slovak}
```

Our last action is to make a note that the commands we have just defined, will be executed by calling the macro `\selectlanguage` at the beginning of the document.

```
36.45 \main@language{slovak}
```

Finally, the category code of @ is reset to its original value. The macrospace used by `\atcatcode` is freed.

```
36.46 \catcode'\@=\atcatcode \let\atcatcode\relax
36.47 </code>
```

## 37 The Slovenian language

The file `slovene.dtx`<sup>50</sup> defines all the language-specific macros for the Slovenian language.

For this language the character " is made active. In table 15 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 compound 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 15: The extra definitions made by `slovene.ldf`

As this file needs to be read only once, we check whether it was read before. If it was, the command `\captionsslovene` is already defined, so we can stop processing. If this command is undefined we proceed with the various definitions and first show the current version of this file.

```
37.1 (*code)
37.2 \ifx\undefined\captionsslovene
37.3 \else
37.4 \selectlanguage{slovene}
37.5 \expandafter\endinput
37.6 \fi
```

`\atcatcode` This file, `slovene.ldf`, may have been read while  $\TeX$  is in the middle of processing a document, so we have to make sure the category code of `@` is ‘letter’ while this file is being read. We save the category code of the `@`-sign in `\atcatcode` and make it ‘letter’. Later the category code can be restored to whatever it was before.

```
37.7 \chardef\atcatcode=\catcode'\@
37.8 \catcode'\@=11\relax
```

Now we determine whether the the common macros from the file `babel.def` need to be read. We can be in one of two situations: either another language option has been read earlier on, in which case that other option has already read `babel.def`, or `slovene` is the first language option to be processed. In that case we need to read `babel.def` right here before we continue.

```
37.9 \ifx\undefined\babel@core@loaded\input babel.def\relax\fi
```

---

<sup>50</sup>The file described in this section has version number ? and was last revised on ?. Contributions were made by Danilo Zavrtanik, University of Ljubljana (YU) and Leon Žlajpah (`leon.zlajpah@ijs.si`).

Another check that has to be made, is if another language definition file has been read already. In that case its definitions have been activated. This might interfere with definitions this file tries to make. Therefore we make sure that we cancel any special definitions. This can be done by checking the existence of the macro `\originalTeX`. If it exists we simply execute it, otherwise it is `\let` to `\empty`.

```
37.10 \ifx\undefined\originalTeX \let\originalTeX\empty \else\originalTeX\fi
```

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.

```
37.11 \ifx\undefined\l@slovene
37.12   \nopatterns{Slovene}
37.13   \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 L<sup>A</sup>T<sub>E</sub>X.

```
37.14 \addto\captionsslovene{%
37.15   \def\prefacename{Predgovor}%
37.16   \def\refname{Literatura}%
37.17   \def\abstractname{Povzetek}%
37.18   \def\bibname{Literatura}%
37.19   \def\chaptername{Poglavje}%
37.20   \def\appendixname{Dodatek}%
37.21   \def\contentsname{Kazalo}%
37.22   \def\listfigurename{Slike}%
37.23   \def\listtablename{Tabele}%
37.24   \def\indexname{Indeks}%
37.25   \def\figurename{Slika}%
37.26   \def\tablename{Tabela}%
37.27   \def\partname{Del}%
37.28   \def\enclname{Priloge}%
37.29   \def\ccname{Kopije}%
37.30   \def\headtoname{Prejme}%
37.31   \def\pagename{Stran}%
37.32   \def\seenname{glej}%
37.33   \def\alsoname{glej tudi}%
37.34   \def\proofname{Proof}% <-- needs translation
37.35   }%
```

`\dateslovene` The macro `\dateslovene` redefines the command `\today` to produce Slovenian dates.

```
37.36 \def\dateslovene{%
37.37 \def\today{\number\day.\~\ifcase\month\or
37.38 januar\or februar\or marec\or april\or maj\or junij\or
37.39 julij\or avgust\or september\or oktober\or november\or december\fi
37.40 \space \number\year}}
```

`\extrasslovene` 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 slovanian group of shorthands should be used.

```
37.41 \initiate@active@char{"}
37.42 \addto\extrasslovene{\languageshorthands{slovene}}
37.43 \addto\extrasslovene{\bbl@activate{}}
37.44 %\addto\noextrasslovene{\bbl@deactivate{}}
```

First we define shorthands to facilitate the occurrence of letters such as č.

```
37.45 \declare@shorthand{slovene}{c}{\textormath{\v c}{\check c}}
37.46 \declare@shorthand{slovene}{s}{\textormath{\v s}{\check s}}
37.47 \declare@shorthand{slovene}{z}{\textormath{\v z}{\check z}}
37.48 \declare@shorthand{slovene}{C}{\textormath{\v C}{\check C}}
37.49 \declare@shorthand{slovene}{L}{\textormath{\v L}{\check L}}
37.50 \declare@shorthand{slovene}{S}{\textormath{\v S}{\check S}}
37.51 \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.

```
37.52 \declare@shorthand{slovene}{' }{%
37.53 \textormath{\quotedblbase}}{\mbox{\quotedblbase}}
37.54 \declare@shorthand{slovene}{' ' }{%
37.55 \textormath{\textquotedblleft}}{\mbox{\textquotedblleft}}
37.56 \declare@shorthand{slovene}{" < }{%
37.57 \textormath{\guillemotleft}}{\mbox{\guillemotleft}}
37.58 \declare@shorthand{slovene}{" > }{%
37.59 \textormath{\guillemotright}}{\mbox{\guillemotright}}
```

then we define two shorthands to be able to specify hyphenation breakpoints that behave a little different from \-.

```
37.60 \declare@shorthand{slovene}{"-}{\allowhyphens-\allowhyphens}
37.61 \declare@shorthand{slovene}{""}{\hskip\z@skip}
```

And we want to have a shorthand for disabling a ligature.

```
37.62 \declare@shorthand{slovene}{" | }{%
37.63 \textormath{\discretionary{-}{ }{\kern.03em}}{}}
```

It is possible that a site might need to add some extra code to the babel macros. To enable this we load a local configuration file, `slovene.cfg` if it is found on T<sub>E</sub>X' search path.

```
37.64 \loadlocalcfg{slovene}
```

Our last action is to make a note that the commands we have just defined, will be executed by calling the macro `\selectlanguage` at the beginning of the document.

```
37.65 \main@language{slovene}
```

Finally, the category code of @ is reset to its original value. The macrospace used by `\atcatcode` is freed.

```
37.66 \catcode'\@=\atcatcode \let\atcatcode\relax
37.67 </code>
```

## 38 The Lower Sorbian language

The file `lsorbian.dtx`<sup>51</sup> It defines all the language-specific macros for Lower Sorbian.

As this file needs to be read only once, we check whether it was read before. If it was, the command `\captionlsorbian` is already defined, so we can stop processing. If this command is undefined we proceed with the various definitions and first show the current version of this file.

```
38.1 <*code>
38.2 \ifx\undefined\captionlsorbian
38.3 \else
38.4   \selectlanguage{lsorbian}
38.5   \expandafter\endinput
38.6 \fi
```

`\atcatcode` This file, `lsorbian.ldf`, may have been read while  $\TeX$  is in the middle of processing a document, so we have to make sure the category code of `@` is ‘letter’ while this file is being read. We save the category code of the `@`-sign in `\atcatcode` and make it ‘letter’. Later the category code can be restored to whatever it was before.

```
38.7 \chardef\atcatcode=\catcode'\@
38.8 \catcode'\@=11\relax
```

Now we determine whether the the common macros from the file `babel.def` need to be read. We can be in one of two situations: either another language option has been read earlier on, in which case that other option has already read `babel.def`, or `lsorbian` is the first language option to be processed. In that case we need to read `babel.def` right here before we continue.

```
38.9 \ifx\undefined\babel@core@loaded\input babel.def\relax\fi
```

Another check that has to be made, is if another language definition file has been read already. In that case its definitions have been activated. This might interfere with definitions this file tries to make. Therefore we make sure that we cancel any special definitions. This can be done by checking the existence of the macro `\originalTeX`. If it exists we simply execute it, otherwise it is `\let` to `\empty`.

```
38.10 \ifx\undefined\originalTeX \let\originalTeX\empty \else\originalTeX\fi
```

When this file is read as an option, i.e. by the `\usepackage` command, `lsorbian` will be an ‘unknown’ language in 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.

```
38.11 \ifx\undefined\l@lsorbian
38.12   \nopatterns{Lsorbian}
38.13   \addialect\l@lsorbian\l@usorbian\fi
```

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

---

<sup>51</sup>The file described in this section has version number ? and was last revised on ?. It was written by Eduard Werner (`edi@kaihh.hanse.de`).

`\captionlsorbian` The macro `\captionlsorbian` defines all strings used in the four standard documentclasses provided with L<sup>A</sup>T<sub>E</sub>X.

```
38.14 \addto\captionlsorbian{%
38.15   \def\prefacename{Zawod}%
38.16   \def\refname{Referency}%
38.17   \def\abstractname{Abstrakt}%
38.18   \def\bibName{Literatura}%
38.19   \def\chaptername{Kapitl}%
38.20   \def\appendixname{Dodawki}%
38.21   \def\contentsname{Wop\'simje\'se}%
38.22   \def\listfigurename{Zapis wobrazow}%
38.23   \def\listtablename{Zapis tabulkow}%
38.24   \def\indexname{Indeks}%
38.25   \def\figurename{Wobraz}%
38.26   \def\tablename{Tabulka}%
38.27   \def\partname{\'Z\'v el}%
38.28   \def\enclname{P\'si\'l oga}%
38.29   \def\ccname{CC}%
38.30   \def\headtoname{Komu}%
38.31   \def\pagename{Strona}%
38.32   \def\seename{gl.}%
38.33   \def\alsoname{gl.~teke}%
38.34   \def\proofname{Proof}% <-- needs translation
38.35 }
```

`\newdatelsorbian` The macro `\newdatelsorbian` redefines the command `\today` to produce Lower Sorbian dates.

```
38.36 \def\newdatelsorbian{%
38.37   \def\today{\number\day.\~\ifcase\month\or
38.38     januara\or februara\or m\vrca\or apryla\or maja\or junija\or
38.39     julija\or awgusta\or septembra\or oktobra\or
38.40     nowembra\or decembra\fi
38.41     \space \number\year}}
```

`\olddatelsorbian` The macro `\olddatelsorbian` redefines the command `\today` to produce old-style Lower Sorbian dates.

```
38.42 \def\olddatelsorbian{%
38.43   \def\today{\number\day.\~\ifcase\month\or
38.44     wjelikego ro\vrzka\or
38.45     ma\l ego ro\vrzka\or
38.46     na\l v etnika\or
38.47     ja\l v sownika\or
38.48     ro\l v zownika\or
38.49     sma\l v znika\or
38.50     pra\l v znika\or
38.51     \l v znje\'nca\or
38.52     po\l v znje\'nca\or
38.53     winowca\or
38.54     nazymnika\or
38.55     godownika\fi \space \number\year}}
```

The default will be the new-style dates.

```
38.56 \let\datelsorbian\newdatelsorbian
```

`\extraslsorbian` The macro `\extraslsorbian` will perform all the extra definitions needed for the  
`\noextraslsorbian` 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.

```
38.57 \addto\extraslsorbian{}  
38.58 \addto\noextraslsorbian{}
```

It is possible that a site might need to add some extra code to the babel macros.  
To enable this we load a local configuration file, `lsorbian.cfg` if it is found on  
TeX' search path.

```
38.59 \loadlocalcfg{lsorbian}
```

Our last action is to make a note that the commands we have just defined,  
will be executed by calling the macro `\selectlanguage` at the beginning of the  
document.

```
38.60 \main@language{lsorbian}
```

Finally, the category code of `@` is reset to its original value. The macrospace  
used by `\atcatcode` is freed.

```
38.61 \catcode'\@=\atcatcode \let\atcatcode\relax  
38.62 \code)
```

## 39 The Upper Sorbian language

The file `usorbian.dtx`<sup>52</sup> It defines all the language-specific macros for Upper Sorbian.

As this file needs to be read only once, we check whether it was read before. If it was, the command `\captionusorbian` is already defined, so we can stop processing. If this command is undefined we proceed with the various definitions and first show the current version of this file.

```
39.1 \ifx\undefined\captionusorbian
39.2 \else
39.3 \selectlanguage{usorbian}
39.4 \expandafter\endinput
39.5 \fi
```

`\atcatcode` This file, `usorbian.sty`, may have been read while  $\TeX$  is in the middle of processing a document, so we have to make sure the category code of `@` is ‘letter’ while this file is being read. We save the category code of the `@`-sign in `\atcatcode` and make it ‘letter’. Later the category code can be restored to whatever it was before.

```
39.6 \chardef\atcatcode=\catcode'\@
39.7 \catcode'\@=11\relax
```

Now we determine whether the the common macros from the file `babel.def` need to be read. We can be in one of two situations: either another language option has been read earlier on, in which case that other option has already read `babel.def`, or `usorbian` is the first language option to be processed. In that case we need to read `babel.def` right here before we continue.

```
39.8 \ifx\undefined\babel@core@loaded\input babel.def\relax\fi
```

Another check that has to be made, is if another language definition file has been read already. In that case its definitions have been activated. This might interfere with definitions this file tries to make. Therefore we make sure that we cancel any special definitions. This can be done by checking the existence of the macro `\originalTeX`. If it exists we simply execute it, otherwise it is `\let` to `\empty`.

```
39.9 \ifx\undefined\originalTeX \let\originalTeX\empty \else\originalTeX\fi
```

When this file is read as an option, i.e. by the `\usepackage` command, `usorbian` will be an ‘unknown’ language in 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.

```
39.10 \ifx\undefined\l@usorbian
39.11 \nopatterns{Usorbian}
39.12 \adddialect\l@usorbian0\fi
```

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

---

<sup>52</sup>The file described in this section has version number ? and was last revised on ?. It was written by Eduard Werner (`edi@kaihh.hanse.de`).

`\captionusorbian` The macro `\captionusorbian` defines all strings used in the four standard documentclasses provided with L<sup>A</sup>T<sub>E</sub>X.

```
39.13 \addto\captionusorbian{%
39.14   \def\prefacename{Zawod}%
39.15   \def\refname{Referency}%
39.16   \def\abstractname{Abstrakt}%
39.17   \def\bibName{Literatura}%
39.18   \def\chaptername{Kapitl}%
39.19   \def\appendixname{Dodawki}%
39.20   \def\contentsname{Wobsah}%
39.21   \def\listfigurename{Zapis wobrazow}%
39.22   \def\listtablename{Zapis tabulkow}%
39.23   \def\indexname{Indeks}%
39.24   \def\figurename{Wobraz}%
39.25   \def\tablename{Tabulka}%
39.26   \def\partname{D\'z\v el}%
39.27   \def\enclname{P\v r\l oha}%
39.28   \def\ccname{CC}%
39.29   \def\headtoname{Komu}%
39.30   \def\pagename{Strona}%
39.31   \def\seename{hl.}%
39.32   \def\alsoname{hl.~te\v z}
39.33   \def\proofname{Proof}% <-- needs translation
39.34 }
```

`\newdateusorbian` The macro `\newdateusorbian` redefines the command `\today` to produce Upper Sorbian dates.

```
39.35 \def\newdateusorbian{%
39.36   \def\today{\number\day.~\ifcase\month\or
39.37   januara\or februara\or m\v erca\or apryla\or meje\or junija\or
39.38   julija\or awgusta\or septembra\or oktobra\or
39.39   nowembra\or decembra\fi
39.40   \space \number\year}}
```

`\olddateusorbian` The macro `\olddateusorbian` redefines the command `\today` to produce old-style Upper Sorbian dates.

```
39.41 \def\olddateusorbian{%
39.42   \def\today{\number\day.~\ifcase\month\or
39.43   wulkeho r\'o\v zka\or ma\l eho r\'o\v zka\or nal\v etnika\or
39.44   jutrownika\or r\'o\v zownika\or sma\v znika\or pra\v znika\or
39.45   \v znjenca\or po\v znjenca\or winowca\or nazymnika\or
39.46   hodownika\fi \space \number\year}}
```

The default will be the new-style dates.

```
39.47 \let\dateusorbian\newdateusorbian
```

`\extrasorbian` The macro `\extrasorbian` will perform all the extra definitions needed for the Upper Sorbian language. It's pirated from `germanb.sty`. The macro `\noextrasorbian` is used to cancel the actions of `\extrasorbian`.

Because for Upper Sorbian (as well as for Dutch) the " character is made active. This is done once, later on its definition may vary.

```
39.48 \initiate@active@char{"}
39.49 \addto\extrasorbian{\languageshorthands{usorbian}}
```

```

39.50 \addto\extrasusorbian{\bbl@activate{}}
39.51 %\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  $\TeX$ book).

```

39.52 \addto\extrasusorbian{\babel@savevariable{\lccode‘\^^Y}%
39.53 \lccode‘\^^Y‘\^^Y}

```

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

```

39.54 \addto\extrasusorbian{\babel@save“\umlautlow}
39.55 \addto\noextrasusorbian{\umlauthigh}

```

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

```

39.56 \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 `ˆ`. Also we store the original meaning of the command `\` for future use.

```

39.57 \begingroup \catcode‘\ "12
39.58 \def\x{\endgroup
39.59 \def\@SS{\mathchar"7019 }
39.60 \def\dq{"}
39.61 \x

```

Now we can define the doublequote macros: the umlauts,

```

39.62 \declare@shorthand{usorbian}{a}{\textormath{\{a}\{\ddot a}}
39.63 \declare@shorthand{usorbian}{o}{\textormath{\{o}\{\ddot o}}
39.64 \declare@shorthand{usorbian}{u}{\textormath{\{u}\{\ddot u}}
39.65 \declare@shorthand{usorbian}{A}{\textormath{\{A}\{\ddot A}}
39.66 \declare@shorthand{usorbian}{O}{\textormath{\{O}\{\ddot O}}
39.67 \declare@shorthand{usorbian}{U}{\textormath{\{U}\{\ddot U}}

```

tremas,

```

39.68 \declare@shorthand{usorbian}{e}{\textormath{\{e}\{\ddot e}}
39.69 \declare@shorthand{usorbian}{E}{\textormath{\{E}\{\ddot E}}
39.70 \declare@shorthand{usorbian}{i}{\textormath{\{i}\{\ddot i}\{\imath}}}
39.71 \declare@shorthand{usorbian}{I}{\textormath{\{I}\{\ddot I}}

```

usorbian es-zet (sharp s),

```

39.72 \declare@shorthand{usorbian}{s}{\textormath{\{s}\{\@SS}}}
39.73 \declare@shorthand{usorbian}{S}{SS}

```

german and french quotes,

```

39.74 \declare@shorthandusorbian{‘}{‘}{%
39.75 \textormath{\quotedblbase}{\mbox{\quotedblbase}}
39.76 \declare@shorthand{usorbian}{’}{’}{%
39.77 \textormath{\textquotedblleft}{\mbox{\textquotedblleft}}
39.78 \declare@shorthand{usorbian}{<}{<}{%
39.79 \textormath{\guillemotleft}{\mbox{\guillemotleft}}
39.80 \declare@shorthand{usorbian}{>}{>}{%
39.81 \textormath{\guillemotright}{\mbox{\guillemotright}}

```

discretionary commands

```

39.82 \declare@shorthand{usorbian}{"c"}{%
39.83 \textormath{\usorbian@dq@disc ck}{c}}
39.84 \declare@shorthand{usorbian}{"C"}{%
39.85 \textormath{\usorbian@dq@disc CK}{C}}
39.86 \declare@shorthand{usorbian}{"f"}{%
39.87 \textormath{\usorbian@dq@disc f{ff}}{f}}
39.88 \declare@shorthand{usorbian}{"F"}{%
39.89 \textormath{\usorbian@dq@disc F{FF}}{F}}
39.90 \declare@shorthand{usorbian}{"l"}{%
39.91 \textormath{\usorbian@dq@disc l{ll}}{l}}
39.92 \declare@shorthand{usorbian}{"L"}{%
39.93 \textormath{\usorbian@dq@disc L{LL}}{L}}
39.94 \declare@shorthand{usorbian}{"m"}{%
39.95 \textormath{\usorbian@dq@disc m{mm}}{m}}
39.96 \declare@shorthand{usorbian}{"M"}{%
39.97 \textormath{\usorbian@dq@disc M{MM}}{M}}
39.98 \declare@shorthand{usorbian}{"n"}{%
39.99 \textormath{\usorbian@dq@disc n{nn}}{n}}
39.100 \declare@shorthand{usorbian}{"N"}{%
39.101 \textormath{\usorbian@dq@disc N{NN}}{N}}
39.102 \declare@shorthand{usorbian}{"p"}{%
39.103 \textormath{\usorbian@dq@disc p{pp}}{p}}
39.104 \declare@shorthand{usorbian}{"P"}{%
39.105 \textormath{\usorbian@dq@disc P{PP}}{P}}
39.106 \declare@shorthand{usorbian}{"t"}{%
39.107 \textormath{\usorbian@dq@disc t{tt}}{t}}
39.108 \declare@shorthand{usorbian}{"T"}{%
39.109 \textormath{\usorbian@dq@disc T{TT}}{T}}

```

and some additional commands:

```

39.110 \declare@shorthand{usorbian}{"-"}{\penalty\@M\-\allowhyphens}
39.111 \declare@shorthand{usorbian}{"|"}{%
39.112 \textormath{\penalty\@M\discretionary{-}{-}{\kern.03em}%
39.113 \allowhyphens}{}}
39.114 \declare@shorthand{usorbian}{""}{\hskip\z@skip}

```

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

```

\c#39.115 \def\mdqon{\bbl@activate{}}
39.116 \def\mdqoff{\bbl@deactivate{}}
39.117 \def\ck{\allowhyphens\discretionary{k-}{k}{ck}\allowhyphens}

```

It is possible that a site might need to add some extra code to the babel macros. To enable this we load a local configuration file, `usorbian.cfg` if it is found on T<sub>E</sub>X' search path.

```
39.118 \loadlocalcfg{usorbian}
```

Our last action is to make a note that the commands we have just defined, will be executed by calling the macro `\selectlanguage` at the beginning of the document.

```
39.119 \main@language{usorbian}
```

Finally, the category code of `@` is reset to its original value.

```
39.120 \catcode'\@=\atcatcode
```

## 40 The Turkish language

The file `turkish.dtx`<sup>53</sup> defines all the language definition macros for the Turkish language<sup>54</sup>.

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 `\frenhspace` is set.

As this file needs to be read only once, we check whether it was read before. If it was, the command `\captionsturkish` is already defined, so we can stop processing. If this command is undefined we proceed with the various definitions and first show the current version of this file.

```
40.1 (*code)
40.2 \ifx\undefined\captionsturkish
40.3 \else
40.4   \selectlanguage{turkish}
40.5   \expandafter\endinput
40.6 \fi
```

`\atcatcode` This file, `turkish.sty`, may have been read while  $\TeX$  is in the middle of processing a document, so we have to make sure the category code of `@` is ‘letter’ while this file is being read. We save the category code of the `@`-sign in `\atcatcode` and make it ‘letter’. Later the category code can be restored to whatever it was before.

```
40.7 \chardef\atcatcode=\catcode'\@
40.8 \catcode'\@=11\relax
```

Now we determine whether the the common macros from the file `babel.def` need to be read. We can be in one of two situations: either another language option has been read earlier on, in which case that other option has already read `babel.def`, or `turkish` is the first language option to be processed. In that case we need to read `babel.def` right here before we continue.

```
40.9 \ifx\undefined\babel@core@loaded\input babel.def\relax\fi
```

Another check that has to be made, is if another language definition file has been read already. In that case its definitions have been activated. This might interfere with definitions this file tries to make. Therefore we make sure that we cancel any special definitions. This can be done by checking the existence of the macro `\originalTeX`. If it exists we simply execute it.

```
40.10 \ifx\undefined\originalTeX \let\originalTeX\empty\fi
40.11 \originalTeX
```

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.

```
40.12 \ifx\undefined\l@turkish
40.13   \@nopatterns{Turkish}
40.14   \adddialect\l@turkish0\fi
```

<sup>53</sup>The file described in this section has version number ? and was last revised on ?.

<sup>54</sup>Mustafa Burc, `z6001@rziris01.rrz.uni-hamburg.de` provided the code for this file. It is based on the work by Pierre Mackay

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 L<sup>A</sup>T<sub>E</sub>X.

```
40.15 \addto\captionsturkish{%
40.16   \def\prefacename{Preface}% <-- This needs translation!!
40.17   \def\refname{Ba\c svurulan Kitaplar}%
40.18   \def\abstractname{Konu}%
40.19   \def\bibName{Bibliografi}%
40.20   \def\chaptername{Anab\ "ol\ "um}%
40.21   \def\appendixname{Appendix}%
40.22   \def\contentsname{\.I\c cindekiler}%
40.23   \def\listfigurename{\c Sekiller Listesi}%
40.24   \def\listtablename{Tablolar\i{n} Listesi}%
40.25   \def\indexname{\.Index}%
40.26   \def\figurename{\c Sekiller}%
40.27   \def\tablename{Tablo}%
40.28   \def\partname{B\ "ol\ "um}%
40.29   \def\enclname{Ekler}%
40.30   \def\ccname{G\ "onderen}%
40.31   \def\headtoname{A1\i{c}\i}%
40.32   \def\pagename{Sayfa}%
40.33   \def\subjectname{To}% <-- This needs translation!!
40.34   \def\seename{see}% <-- This needs translation!!
40.35   \def\alsoname{see also}% <-- This needs translation!!
40.36   \def\proofname{Proof}% <-- This needs translation!!
40.37 }%
```

`\dateturkish` The macro `\dateturkish` redefines the command `\today` to produce Turkish dates.

```
40.38 \def\dateturkish{%
40.39   \def\today{\number\day.\~\ifcase\month\or
40.40     Ocak\or \c Subat\or Mart\or Nisan\or May\i{s}\or Haziran\or
40.41     Temmuz\or A\u gustos\or Eyl\ "ul\or Ekim\or Kas\i{m}\or
40.42     Aral\i{k}\fi
40.43   \space\number\year}}
```

`\extrasturkish` 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.

```
40.44 \initiate@active@char{:}
40.45 \initiate@active@char{!}
40.46 \initiate@active@char{=}
```

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

```
40.47 \addto\extrasturkish{\languageshorthands{turkish}}
```

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

```
40.48 \addto\extrasturkish{%
```

```
40.49 \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.

```
40.50 \addto\extrasturkish{\bbl@frenchspacing}
```

```
40.51 \addto\noextrasturkish{\bbl@nonfrenchspacing}
```

```
\turkish@sh@:  The definitions for the three active characters were made using intermediate
\turksih@sh@! macros. These are defined now. The insertion of extra ‘white space’ should only
\turkish@sh@=  happen outside math mode, hence the check \ifmmode in the macros.
```

```
40.52 \declare@shorthand{turkish}{:}{%}
```

```
40.53 \ifmmode
```

```
40.54 \string:%
```

```
40.55 \else\relax
```

```
40.56 \ifhmode
```

```
40.57 \ifdim\lastskip>\z@
```

```
40.58 \unskip\penalty\M\thinspace
```

```
40.59 \fi
```

```
40.60 \fi
```

```
40.61 \string:%
```

```
40.62 \fi}
```

```
40.63 \declare@shorthand{turkish}{!}{%}
```

```
40.64 \ifmmode
```

```
40.65 \string!%
```

```
40.66 \else\relax
```

```
40.67 \ifhmode
```

```
40.68 \ifdim\lastskip>\z@
```

```
40.69 \unskip\penalty\M\thinspace
```

```
40.70 \fi
```

```
40.71 \fi
```

```
40.72 \string!%
```

```
40.73 \fi}
```

```
40.74 \declare@shorthand{turkish}{=}{%}
```

```
40.75 \ifmmode
```

```
40.76 \string=%
```

```
40.77 \else\relax
```

```
40.78 \ifhmode
```

```
40.79 \ifdim\lastskip>\z@
```

```
40.80 \unskip\kern\fontdimen2\font
```

```
40.81 \kern-1.4\fontdimen3\font
```

```
40.82 \fi
```

```
40.83 \fi
```

```
40.84 \string=%
```

```
40.85 \fi}
```

It is possible that a site might need to add some extra code to the babel macros.

To enable this we load a local configuration file, `turkish.cfg` if it is found on T<sub>E</sub>X’ search path.

```
40.86 \loadlocalcfg{turkish}
```

Our last action is to make a note that the commands we have just defined, will be executed by calling the macro `\selectlanguage` at the beginning of the document.

```
40.87 \main@language{turkish}
```

Finally, the category code of @ is reset to its original value. The macrospace used by `\atcatcode` is freed.

```
40.88 \catcode'\@atcatcode \let\atcatcode\relax
40.89 </code>
```

## 41 The Bahasa language

The file `bahasa.dtx`<sup>55</sup> 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 pronunciation, this file can be used for both languages.

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

As this file needs to be read only once, we check whether it was read before. If it was, the command `\captionbahasa` is already defined, so we can stop processing. If this command is undefined we proceed with the various definitions and first show the current version of this file.

```
41.1 (*code)
41.2 \ifx\undefined\captionbahasa
41.3 \else
41.4   \selectlanguage{bahasa}
41.5   \expandafter\endinput
41.6 \fi
```

`\atcatcode` This file, `bahasa.sty`, may have been read while `TeX` is in the middle of processing a document, so we have to make sure the category code of `@` is ‘letter’ while this file is being read. We save the category code of the `@`-sign in `\atcatcode` and make it ‘letter’. Later the category code can be restored to whatever it was before.

```
41.7 \chardef\atcatcode=\catcode‘\@
41.8 \catcode‘\@=11\relax
```

Now we determine whether the common macros from the file `babel.def` need to be read. We can be in one of two situations: either another language option has been read earlier on, in which case that other option has already read `babel.def`, or `bahasa` is the first language option to be processed. In that case we need to read `babel.def` right here before we continue.

```
41.9 \ifx\undefined\babel@core@loaded\input babel.def\relax\fi
```

Another check that has to be made, is if another language definition file has been read already. In that case its definitions have been activated. This might interfere with definitions this file tries to make. Therefore we make sure that we cancel any special definitions. This can be done by checking the existence of the macro `\originalTeX`. If it exists we simply execute it, otherwise it is `\let` to `\empty`.

```
41.10 \ifx\undefined\originalTeX \let\originalTeX\empty\fi
41.11 \originalTeX
```

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.

```
41.12 \ifx\undefined\l@bahasa
41.13   \nopatterns{Bahasa}
41.14   \adddialect\l@bahasa0\fi
```

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

---

<sup>55</sup>The file described in this section has version number ? and was last revised on ?.

`\captionsbahasa` The macro `\captionsbahasa` defines all strings used in the four standard documentclasses provided with L<sup>A</sup>T<sub>E</sub>X.

```
41.15 \addto\captionsbahasa{%
41.16   \def\prefacename{Pendahuluan}%
41.17   \def\refname{Pustaka}%
41.18   \def\abstractname{Ringkasan}% (sometime it's called 'intisari'
41.19                                     % or 'ikhtisar')
41.20   \def\bibName{Bibliografi}%
41.21   \def\chaptername{Bab}%
41.22   \def\appendixname{Lampiran}%
41.23   \def\contentsname{Daftar Isi}%
41.24   \def\listfigurename{Daftar Gambar}%
41.25   \def\listtablename{Daftar Tabel}%
41.26 % Glossary: Daftar Istilah
41.27   \def\indexname{Indeks}%
41.28   \def\figurename{Gambar}%
41.29   \def\tablename{Tabel}%
41.30   \def\partname{Bagian}%
41.31 % Subject: Subyek
41.32 % From: Dari
41.33   \def\enclname{Lampiran}%
41.34   \def\ccname{cc}%
41.35   \def\headtoname{Kepada}%
41.36   \def\pagename{Halaman}%
41.37 % Notes (Endnotes): Catatan
41.38   \def\seenname{lihat}%
41.39   \def\alsoname{lihat juga}%
41.40   \def\proofname{Proof}% <-- needs translation
41.41 }
```

`\datebahasa` The macro `\datebahasa` redefines the command `\today` to produce Bahasa dates.

```
41.42 \def\datebahasa{%
41.43   \def\today{\number\day~\ifcase\month\or
41.44     Januari\or Februari\or Maret\or April\or Mei\or Juni\or
41.45     Juli\or Agustus\or September\or Oktober\or Nopember\or Desember\fi
41.46     \space \number\year}}
```

`\extrasbahasa` The macro `\extrasbahasa` will perform all the extra definitions needed for the Bahasa language. The macro `\noextrasbahasa` 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.

```
41.47 \addto\extrasbahasa{}
41.48 \addto\noextrasbahasa{}
```

It is possible that a site might need to add some extra code to the babel macros. To enable this we load a local configuration file, `bahasa.cfg` if it is found on T<sub>E</sub>X' search path.

```
41.49 \loadlocalcfg{bahasa}
```

Our last action is to make a note that the commands we have just defined, will be executed by calling the macro `\selectlanguage` at the beginning of the document.

```
41.50 \main@language{bahasa}
```

Finally, the category code of @ is reset to its original value. The macrospace used by `\atcatcode` is freed.

```
41.51 \catcode'\@=\atcatcode \let\atcatcode\relax
41.52 </code>
```

## Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

<b>Symbols</b>	<code>\bbl@eq@</code> . . . . .	<u>8.150</u>	<code>\captionsslovene</code>	<u>37.14</u>
<code>\- . . . .</code>	<code>\bbl@firstcs</code> . . . .	<u>8.356</u>	<code>\captionsspanish</code>	<u>23.15</u>
<u>24.137, 25.93, 30.70</u>	<code>\bbl@frenchspacing</code> .	<u>6, 8.429</u>	<code>\captionsswedish</code>	<u>29.14</u>
<code>\@acute</code> <u>23.66, 24.72, 25.65</u>	. . . . .	<u>6, 8.429</u>	<code>\captionsturkish</code>	<u>40.15</u>
<code>\@bibitem</code> . . . . .	<code>\bbl@main@language</code>	<u>8.113</u>	<code>\captionsusorbian</code>	<u>39.13</u>
<u>8.800</u>	<code>\bbl@nonfrenchspacing</code>	<u>6, 8.429</u>	<code>\ck . . . . .</code>	<u>16.111, 39.115</u>
<code>\@citex</code> . . . . .	. . . . .	<u>6, 8.429</u>		
<u>8.794</u>	<code>\bbl@pr@m@s</code> . . . . .	<u>8.386</u>	<b>D</b>	
<code>\@grave</code> . . . . .	<code>\bbl@remove@special</code>	<u>6, 8.236</u>	<code>\dateamerican</code> . . .	<u>15.58</u>
<u>24.72</u>	. . . . .	<u>6, 8.236</u>	<code>\dateaustrian</code> . . .	<u>16.45</u>
<code>\@lbibitem</code> . . . . .	<code>\bbl@scndcs</code> . . . . .	<u>8.356</u>	<code>\datebahasa</code> . . . . .	<u>41.42</u>
<u>8.802</u>	<code>\bibcite</code> . . . . .	<u>8.798</u>	<code>\datebrazil</code> . . . . .	<u>22.63</u>
<code>\@nolanerr</code> . . . . .	<code>\brasilhyphenmins</code>	<u>22.68</u>	<code>\datebreton</code> . . . . .	<u>17.37</u>
<u>8.122</u>	<code>\breton@sh@:@</code> . . .	<u>17.67</u>	<code>\datecatalan</code> . . . .	<u>24.40</u>
<code>\@nopatterns</code> . . . .	<code>\breton@sh@;@</code> . . .	<u>17.56</u>	<code>\datecroatian</code> . . .	<u>33.36</u>
<u>8.122</u>	<code>\breton@sh@?@</code> . . .	<u>17.83</u>	<code>\dateczech</code> . . . . .	<u>34.36</u>
<code>\@testdef</code> . . . . .	<code>\breton@sh@@</code> . . . .	<u>17.67</u>	<code>\datedanish</code> . . . . .	<u>27.36</u>
<u>8.781</u>			<code>\datedutch</code> . . . . .	<u>14.42</u>
<code>\@tilde</code> . . . . .			<code>\dateenglish</code> . . . .	<u>15.46</u>
<u>23.66, 25.65</u>			<code>\dateesperanto</code> . . .	<u>13.37</u>
<code>\@trema</code> . . . . .			<code>\dateestonian</code> . . .	<u>32.41</u>
<u>14.54</u>			<code>\datefinnish</code> . . . .	<u>30.36</u>
<code>\@umlaut</code> . . . . .			<code>\datefrancais</code> . . .	<u>20.43</u>
<u>23.66, 24.72, 25.65</u>			<code>\dategalician</code> . . .	<u>25.39</u>
			<code>\dategerman</code> . . . . .	<u>16.40</u>
<b>A</b>			<code>\dateirish</code> . . . . .	<u>18.39</u>
<code>\active@prefix</code> . . .			<code>\dateitalian</code> . . . .	<u>21.37</u>
<u>8.337</u>			<code>\datelang</code> . . . . .	<u>5</u>
<code>\adddialect</code> . . . . .	<b>C</b>		<code>\datemagyar</code> . . . . .	<u>31.36</u>
<u>4, 8.36</u>	<code>\captionsamerican</code>	<u>15.45</u>	<code>\datenorsk</code> . . . . .	<u>28.63</u>
<code>\addlanguage</code> . . . .	<code>\captionsbahasa</code> . .	<u>41.15</u>	<code>\datenynorsk</code> . . . .	<u>28.68</u>
<u>4, 8.24</u>	<code>\captionsbrazil</code> . .	<u>22.37</u>	<code>\datepolish</code> . . . . .	<u>35.39</u>
<code>\addto</code> . . . . .	<code>\captionsbreton</code> . .	<u>17.15</u>	<code>\dateportuges</code> . . .	<u>22.58</u>
<u>6, 8.437</u>	<code>\captionscatalan</code>	<u>24.18</u>	<code>\dateromanian</code> . . .	<u>26.36</u>
<code>\allowhyphens</code> . . .	<code>\captionscroatian</code>	<u>33.14</u>	<code>\datescottish</code> . . .	<u>19.39</u>
<u>6, 8.445</u>	<code>\captionsczech</code> . . .	<u>34.14</u>	<code>\dateslovak</code> . . . . .	<u>36.36</u>
<code>\Aob</code> . . . . .	<code>\captionsdanish</code> . .	<u>27.14</u>	<code>\dateslovene</code> . . . .	<u>37.36</u>
<u>35.60</u>	<code>\captionsdanish</code> . .	<u>27.14</u>	<code>\datespanish</code> . . . .	<u>23.37</u>
<code>\aob</code> . . . . .	<code>\captionsdutch</code> . . .	<u>14.16</u>	<code>\dateswedish</code> . . . .	<u>29.36</u>
<u>35.60</u>	<code>\captionsenglish</code>	<u>15.23</u>	<code>\dateturkish</code> . . . .	<u>40.38</u>
<code>\atcatcode</code> . . . . .	<code>\captionseesperanto</code>	<u>13.14</u>	<code>\declare@shorthand</code> .	<u>5, 8.358</u>
<u>13.7, 14.7, 15.7,</u>	<code>\captionsestonian</code>	<u>32.15</u>	. . . . .	<u>5, 8.358</u>
<u>16.7, 17.7, 18.7,</u>	<code>\captionsestonian</code>	<u>32.15</u>	<code>\defineshorthand</code> <u>3, 8.380</u>	
<u>19.7, 20.7, 21.7,</u>	<code>\captionsestonian</code>	<u>30.14</u>	<code>\dieresis</code> . . . . .	
<u>22.7, 23.7, 24.7,</u>	<code>\captionsestonian</code>	<u>20.21</u>	. . . . .	
<u>25.7, 26.7, 27.7,</u>	<code>\captionsgalician</code>	<u>25.17</u>	<u>23.63, 24.69, 25.62</u>	
<u>28.7, 29.7, 30.7,</u>	<code>\captionsgerman</code> . . .	<u>16.17, 16.39</u>	<code>\DJ</code> . . . . .	<u>8.508</u>
<u>31.7, 33.7, 34.7,</u>	. . . . .	<u>16.17, 16.39</u>	<code>\dj</code> . . . . .	<u>8.508</u>
<u>35.6, 36.7, 37.7,</u>	<code>\captionsirish</code> . . .	<u>18.17</u>	<code>\doc@style</code> . . . . .	<u>8.608</u>
<u>38.7, 39.6, 40.7, 41.7</u>	<code>\captionsitalian</code>	<u>21.15</u>		
	<code>\captionslang</code> . . . . .	<u>5</u>		
<b>B</b>	<code>\captionslsorbian</code>	<u>38.14</u>		
<code>\babel@beginsave</code> <u>8.412</u>	<code>\captionsmagyar</code> . .	<u>31.14</u>		
<code>\babel@sanitize@arg</code>	<code>\captionsnorsk</code> . . .	<u>28.17</u>		
. . . . .	<code>\captionssynorsk</code>	<u>28.40</u>		
<u>8.772</u>	<code>\captionsspanish</code> . .	<u>35.17</u>		
<code>\babel@save</code> . . . . .	<code>\captionssportuges</code>	<u>22.15</u>		
<u>6, 8.415</u>	<code>\captionssromanian</code>	<u>26.14</u>		
<code>\babel@savecnt</code> . . .	<code>\captionsscottish</code>	<u>19.17</u>		
<u>8.412</u>	<code>\captionsslovak</code> . .	<u>36.14</u>		
<code>\babel@savevariable</code>				
. . . . .				
<u>6, 8.424</u>				
<code>\bbl@activate</code> . . . .				
<u>5, 8.344</u>				
<code>\bbl@add@special</code> <u>6, 8.226</u>				
<code>\bbl@afterelse</code> . . .				
<u>8.247</u>				
<code>\bbl@afterfi</code> . . . .				
<u>8.247</u>				
<code>\bbl@deactivate</code> <u>5, 8.350</u>				

<code>\dq</code> .	<a href="#">16.59</a> , <a href="#">35.111</a> , <a href="#">39.57</a>	<code>\french@sh@:</code> . . .	<a href="#">20.74</a>	<code>\mdqon</code> . . . . .	<a href="#">16.111</a> ,	
<code>\dutchhyphenmins</code>	<a href="#">14.53</a>	<code>\french@sh@;</code> . . .	<a href="#">20.63</a>		<a href="#">35.145</a> , <a href="#">39.115</a>	
<b>E</b>			<code>\french@sh@?</code> . . .	<b>N</b>		
environments:		<code>\french@sh@@</code> . . . . .	<a href="#">20.74</a>	<code>\newcheck</code> . . . . .	<a href="#">32.74</a>	
language . . . . .	<a href="#">2</a>	<code>\frq</code> . . . . .	<a href="#">8.556</a>	<code>\newdatelsorbian</code>	<a href="#">38.36</a>	
otherlanguage . . .	<a href="#">15</a>	<code>\frqq</code> . . . . .	<a href="#">8.560</a>	<code>\newdateusorbian</code>	<a href="#">39.35</a>	
<code>\Eob</code> . . . . .	<a href="#">35.60</a>	<b>G</b>			<code>\newlabel</code> . . . . .	<a href="#">8.777</a>
<code>\eob</code> . . . . .	<a href="#">35.60</a>	<code>\gentilde</code> . . . . .	<a href="#">32.74</a>	<code>\newtilde</code> . . . . .	<a href="#">32.74</a>	
<code>\Esper</code> . . . . .	<a href="#">13.51</a>	<code>\german@dq@disc</code> .	<a href="#">16.64</a>	<code>\nocite</code> . . . . .	<a href="#">8.796</a>	
<code>\esper</code> . . . . .	<a href="#">13.51</a>	<code>\glq</code> . . . . .	<a href="#">8.548</a>	<code>\noextrasamerican</code>	<a href="#">15.65</a>	
<code>\estonianhyphenmins</code>		<code>\glqq</code> . . . . .	<a href="#">8.552</a>	<code>\noextrasaustrian</code>	<a href="#">16.57</a>	
. . . . .	<a href="#">32.73</a>	<code>\grq</code> . . . . .	<a href="#">8.548</a>	<code>\noextrasbahasa</code> .	<a href="#">41.47</a>	
<code>\extrasamerican</code> .	<a href="#">15.65</a>	<code>\grqq</code> . . . . .	<a href="#">8.552</a>	<code>\noextrasbrazil</code> .	<a href="#">22.88</a>	
<code>\extrasaustrian</code> .	<a href="#">16.57</a>	<code>\guillemotleft</code> . .	<a href="#">8.462</a>	<code>\noextrasbreton</code> .	<a href="#">17.43</a>	
<code>\extrasbahasa</code> . . .	<a href="#">41.47</a>	<code>\guillemotright</code> .	<a href="#">8.462</a>	<code>\noextrascatalan</code>	<a href="#">24.46</a>	
<code>\extrasbrazil</code> . . .	<a href="#">22.88</a>	<code>\guilsinglleft</code> . .	<a href="#">8.480</a>	<code>\noextrascroatian</code>	<a href="#">33.42</a>	
<code>\extrasbreton</code> . . .	<a href="#">17.43</a>	<code>\guilsinglright</code> .	<a href="#">8.480</a>	<code>\noextrasczech</code> . .	<a href="#">34.42</a>	
<code>\extrascatalan</code> . .	<a href="#">24.46</a>	<b>H</b>			<code>\noextrasdanish</code> .	<a href="#">27.41</a>
<code>\extrascroatian</code> .	<a href="#">33.42</a>	<code>\hodiau</code> . . . . .	<a href="#">13.61</a>	<code>\noextrasdutch</code> . .	<a href="#">14.47</a>	
<code>\extrasczech</code> . . . .	<a href="#">34.42</a>	<code>\hodiaun</code> . . . . .	<a href="#">13.61</a>	<code>\noextrasenglish</code>	<a href="#">15.63</a>	
<code>\extrasdanish</code> . . .	<a href="#">27.41</a>	<b>I</b>			<code>\noextrasesperanto</code>	<a href="#">13.42</a>
<code>\extrasdutch</code> . . . .	<a href="#">14.47</a>	<code>\if@safe@actives</code>	<a href="#">8.342</a>	<code>\noextrasesonian</code> .	<a href="#">32.50</a>	
<code>\extrasenglish</code> . .	<a href="#">15.63</a>	<code>\iflanguage</code> . . . .	<a href="#">3</a> , <a href="#">8.39</a>	<code>\noextrasfinnish</code> .	<a href="#">30.42</a>	
<code>\extrasesperanto</code>	<a href="#">13.42</a>	<code>\IJ</code> . . . . .	<a href="#">8.498</a>	<code>\noextrasfrancais</code>	<a href="#">20.50</a>	
<code>\extrasestonian</code> .	<a href="#">32.50</a>	<code>\ij</code> . . . . .	<a href="#">8.498</a>	<code>\noextragalician</code>	<a href="#">25.44</a>	
<code>\extrasfinnish</code> . .	<a href="#">30.42</a>	<code>\initiate@active@char</code>		<code>\noextrasgerman</code> .	<a href="#">16.47</a>	
<code>\extrasfrancais</code> . .	<a href="#">20.50</a>	. . . . .	<a href="#">5</a> , <a href="#">8.247</a>	<code>\noextrasirish</code> . .	<a href="#">18.46</a>	
<code>\extrasgalician</code> . .	<a href="#">25.44</a>	<code>\italianhyphenmins</code>	<a href="#">21.42</a>	<code>\noextrasitalian</code>	<a href="#">21.43</a>	
<code>\extrasgerman</code> . . .	<a href="#">16.47</a>	<b>L</b>			<code>\noextraslang</code> . . . . .	<a href="#">5</a>
<code>\extrasirish</code> . . . .	<a href="#">18.46</a>	<code>\label</code> . . . . .	<a href="#">8.775</a>	<code>\noextraslsorbian</code>	<a href="#">38.57</a>	
<code>\extrasitalian</code> . .	<a href="#">21.43</a>	<code>\langhyphenmin</code> . . . .	<a href="#">5</a>	<code>\noextrasmagyar</code> .	<a href="#">31.64</a>	
<code>\extraslang</code> . . . . .	<a href="#">5</a>	<code>\language</code> . . . . .	<a href="#">8.16</a>	<code>\noextrasnynorsk</code>	<a href="#">28.71</a>	
<code>\extraslsorbian</code> .	<a href="#">38.57</a>	language (environ-		<code>\noextraspolsish</code> .	<a href="#">35.45</a>	
<code>\extrasmagyar</code> . . .	<a href="#">31.64</a>	ment) . . . . .	<a href="#">2</a>	<code>\noextrasportuges</code>	<a href="#">22.70</a>	
<code>\extrasnorsk</code> . . . .	<a href="#">28.69</a>	<code>\language@group</code> .	<a href="#">8.373</a>	<code>\noextrasromanian</code>	<a href="#">26.42</a>	
<code>\extrasnynorsk</code> . . . .	<a href="#">28.69</a> , <a href="#">28.71</a>	<code>\language@name</code> . . . . .	<a href="#">3</a>	<code>\noextrasscottish</code>	<a href="#">19.46</a>	
<code>\extraspolsish</code> . . .	<a href="#">35.45</a>	<code>\languageshorthands</code>		<code>\noextrasslovak</code> .	<a href="#">36.42</a>	
<code>\extrasportuges</code> .	<a href="#">22.70</a>	. . . . .	<a href="#">3</a> , <a href="#">8.381</a>	<code>\noextrasslovene</code>	<a href="#">37.41</a>	
<code>\extrasromanian</code> .	<a href="#">26.42</a>	<code>\last@language</code> . . .	<a href="#">8.19</a>	<code>\noextrasspanish</code>	<a href="#">23.42</a>	
<code>\extrasscottish</code> .	<a href="#">19.46</a>	<code>\Lcaron</code> . . . . .	<a href="#">34.53</a>	<code>\noextrasswedish</code>	<a href="#">29.42</a>	
<code>\extrasslovak</code> . . .	<a href="#">36.42</a>	<code>\lcaron</code> . . . . .	<a href="#">34.53</a>	<code>\noextrasturkish</code>	<a href="#">40.44</a>	
<code>\extrasslovene</code> . .	<a href="#">37.41</a>	<code>\Lgem</code> . . . . .	<a href="#">24.140</a>	<code>\norskhyphenmins</code>	<a href="#">28.15</a>	
<code>\extrasspanish</code> . .	<a href="#">23.42</a>	<code>\lgem</code> . . . . .	<a href="#">24.140</a>	<code>\nynorskhyphenmins</code>	<a href="#">28.15</a>	
<code>\extrasswedish</code> . .	<a href="#">29.42</a>	<code>\loadlocalcfg</code> . . .	<a href="#">5</a> , <a href="#">9.1</a>	<b>O</b>		
<code>\extrasturkish</code> . .	<a href="#">40.44</a>	<code>\lower@umlaut</code> . . .	<a href="#">8.574</a>	<code>\olddatelsorbian</code>	<a href="#">38.42</a>	
<code>\extrasusorbian</code> .	<a href="#">39.48</a>	<b>M</b>			<code>\olddateusorbian</code>	<a href="#">39.41</a>
<b>F</b>			<code>\main@language</code>	<a href="#">5</a> , <a href="#">8.113</a>	<code>\ondatemagyar</code> . . .	<a href="#">31.50</a>
<code>\finishhyphenmins</code>	<a href="#">30.73</a>	<code>\mdqoff</code> . . . . .	<a href="#">16.111</a> ,	<code>\ontoday</code> . . . . .	<a href="#">120</a>	
<code>\flq</code> . . . . .	<a href="#">8.556</a>		<a href="#">35.145</a> ,	<code>\ord</code> . . . . .	<a href="#">22.85</a>	
<code>\flqq</code> . . . . .	<a href="#">8.560</a>		<a href="#">39.115</a>	<code>\orda</code> . . . . .	<a href="#">22.85</a>	
<code>\foreignlanguage</code>	<a href="#">2</a> , <a href="#">8.92</a>	<b>O</b>			<code>\OT1dqpos</code> . . . . .	<a href="#">8.407</a>
				otherlanguage (envi-		
				ronment) . . .	<a href="#">8.83</a>	

<b>P</b>		
<code>\pageref</code> .....	<a href="#">8.784</a>	
<code>\patterns@loaded</code>	<a href="#">8.149</a>	
<code>\portugesghyphenmins</code> .....	<a href="#">22.68</a>	
<code>\process@language</code>	<a href="#">8.170</a>	
<code>\process@line</code> ...	<a href="#">8.150</a>	
<code>\process@synonym</code>	<a href="#">8.159</a>	
<b>Q</b>		
<code>\quotedblbase</code> ...	<a href="#">8.452</a>	
<code>\quotesinglbase</code> .	<a href="#">8.457</a>	
<b>R</b>		
<code>\ra</code> .....	<a href="#">22.85</a>	
<code>\ref</code> .....	<a href="#">8.784</a>	
<code>\ro</code> .....	<a href="#">22.85</a>	
<b>S</b>		
<code>\save@sf@q</code> ...	<a href="#">6, 8.449</a>	
<code>\selectlanguage</code>	<a href="#">3, 8.46</a>	
<code>\set@hyphenmins</code> .	<a href="#">8.111</a>	
<code>\set@low@box</code> ..	<a href="#">6, 8.446</a>	
<code>\skb</code> .....	<a href="#">35.71</a>	
<code>\sob</code> .....	<a href="#">35.53</a>	
<code>\spanishhyphenmins</code>	<a href="#">23.62</a>	
<code>\spb</code> .....	<a href="#">35.64</a>	
<code>\swedishhyphenmins</code>	<a href="#">29.41</a>	
<code>\system@group</code> ...	<a href="#">8.373</a>	
<code>\system@sh:@</code> ..	<a href="#">20.102</a>	
<code>\system@sh@;</code> ..	<a href="#">20.102</a>	
<code>\system@sh@?@</code> ..	<a href="#">20.102</a>	
<code>\system@sh@@</code> ...	<a href="#">20.102</a>	
<b>T</b>		
<code>\Ttdqpos</code> .....	<a href="#">8.407</a>	
<code>\telepl</code> .....	<a href="#">35.86</a>	
<code>\textacute</code> .....	<a href="#">23.63, 24.69, 25.62</a>	
<code>\textgrave</code> .....	<a href="#">24.69</a>	
<code>\textormath</code> .....	<a href="#">8.367</a>	
<code>\textpl</code> .....	<a href="#">35.77</a>	
<code>\texttilde</code>	<a href="#">23.63, 25.62</a>	
<code>\tildelow</code> .....	<a href="#">32.74</a>	
<code>\turkish@sh:@</code> ..	<a href="#">40.52</a>	
<code>\turksih@sh@@</code> ...	<a href="#">40.52</a>	
<b>U</b>		
<code>\umlauthigh</code> .....	<a href="#">8.564</a>	
<code>\umlautlow</code> .....	<a href="#">8.564</a>	
<code>\up</code> .....	<a href="#">24.172</a>	
<code>\user@group</code> .....	<a href="#">8.373</a>	
<code>\usesorthands</code>	<a href="#">3, 8.376</a>	
<b>V</b>		
<code>\v</code> .....	<a href="#">34.46</a>	

## Change History

babel-3.5a	
\bb1@pr@ms: Added macro	26
"General": Added a system shorthand for the right quote	26
Replaced 16 system shorthands to deal with hex numbers by one	26
\initiate@active@char: Added a check for right quote and adapt \pr@ms if necessary	22
babel 2.0a	
"General": Added text about <code>german.sty</code>	1
babel 2.0b	
"General": Changed order of code to prevent plain TeX from seeing all of it	1
babel 2.1	
"General": Modified user interface, <code>\langTeX</code> no longer necessary	1
babel 2.1a	
"General": Incorporated Nico's comments	1
babel 2.1b	
"General": rename <code>\language</code> to <code>\current@language</code>	1
babel 2.1c	
"General": abstract for report fixed, missing <code>}</code> , found by Nicolas Brouard BROUARD@FRINED51.BITNET	1
babel 2.1d	
"General": Missing right brace in definition of abstract environment, found by Werenfried Spit	1
babel 2.1e	
"General": Incorporated more comments from Nico	1
babel 2.2	
"General": Renamed <code>\newlanguage</code> to <code>\addlanguage</code>	1
babel 2.2a	
"General": Modified the documentation somewhat	1
babel 3.0	
"General": Moved part of the code to <code>hyphen.doc</code> in preparation for TeX 3.0	1
babel 3.0a	
"General": Updated comments in various places	1
babel 3.0b	
"General": Removed some problems in change log	1
babel 3.0c	
"General": Renamed <code>babel.sty</code> and <code>latexhax.sty</code> to <code>.com</code>	1
babel 3.0d	
\doc@style: Removed use of <code>\@ifundefined</code>	34
"General": Removed use of <code>\@ifundefined</code>	11
babel 3.1	
\addto: Added macro	28
"General": Added the support for active characters and for extending a macro	1
Removed definition of <code>\if@restonecol</code>	35
Removed the need for <code>latexhax</code>	1
babel 3.2	
\babel@beginsave: Added macro	27
\babel@save: Added macro	27
\babel@savecnt: Added macro	27
\babel@savevariable: Added macro	28
\bb1@add@special: Added macro	21
\bb1@remove@special: Added macro	21
"General": Some Changes by br	1

babel 3.2a	
"General": Fixups of the code and documentation	1
babel 3.2b	
\allowhyphens: Moved macro from language definition files	28
\save@sf@q: Moved macro from language definition files	29
\set@low@box: Moved macro from language definition files	29
babel 3.2c	
\babel@save: missing backslash led to errors when executing \originalTeX	27
babel 3.2d	
\babel@save: saving in \babel@i and restoring from \@babel@i doesn't work very well...	27
babel 3.2e	
"General": Added slovak	43
babel 3.3	
"General": \headpagename should be \pagename	38
Added catalan and galician	43
Added turkish	43
Included driver file, and prepared for dsitribution	1
babel 3.4	
\addto: Changed to use toks register	28
"General": Added bahasa	43
Added language definition file for bahasa	1
Updated for L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub>	1
babel 3.4b	
"General": Added a small driver to be able to process just this file	1
Use the ltxdoc class instead of article	41
babel 3.4c	
"General": lhyphen.cfg has become lthyphen.cfg	7
babel 3.4e	
\@nolanerr: Use \PackageError in L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub> mode	16
\@nopatterns: Macro added	16
"General": Redid the identification code, provided dummy definition of \ProvidesFile for plain T <sub>E</sub> X	7
\process@language: Added code to detect assignments to left- and righthyphen- min in the patternfile.	18
babel 3.4g	
\@testdef: Moved the \def inside the macrocode environment	39
babel 3.5a	
\@nopatterns: Added \@activated to log active characters	16
\bbl@activate: Added macro	25
\bbl@deactivate: Added macro	25
\bbl@main@language: Macro added	16
"General": Added breton, irish, scottish	43
Changed extension of language definition files to ldf	8
Provided common code to handle the active double quote	1
\initiate@active@char: Added macro	22
\main@language: Macro added	16
\selectlanguage: write the language change to the auxiliary files	14
babel 3.5b	
\bbl@eq@: added macro	17
"General": Added brazilian as alternative for brazil	8
Added estonian option	9
Added lsorbian, usorbian	43
lthyphen.cfg has become hyphen.cfg	7

Now add a <code>\space</code> and a <code>/</code> character	19
<code>\initiate@active@char</code> : Renamed macro	22
<code>\pageref</code> : Made <code>\ref</code> and <code>\pageref</code> robust (PR1353)	39
<code>\process@language</code> : Added optional reading of file with hyphenation exceptions	18
<code>\process@line</code> : added macro	17
<code>\process@synonym</code> : added macro	17
<code>\selectlanguage</code> : Added an extra level of expansion to separate the switching mechanism from writing to aux files	14
Added default setting of <code>hyphenmin</code> parameters	15
Changed the name of the internal macro to <code>\selectlanguage</code>	14
Separated the setting of the <code>hyphenmin</code> values	14
Store the name of the current language in a control sequence instead of passing the whole macro construct to strip the escape character in the argument of <code>\selectlanguage</code>	13
babel 3.5c	
<code>\@nopatterns</code> : Added missing closing brace	16
"General": Changed the order of including the language files somewhat (PR1652)	43
corrected a few typos (PR1652)	1
Repaired a typo (itlaic, PR1652)	29
babel 3.5d	
<code>\active@prefix</code> : <code>\@protected@cmd</code> has vanished from <code>ltoutenc.dtx</code>	24
<code>\declare@shorthand</code> : Make a 'note' when a shorthand with an argument is defined.	25
<code>\foreignlanguage</code> : Macro added	15
"General": Added <code>british</code> as an alternative for 'english' with a preference for <code>british</code> hyphenation	8
Added options to influence behaviour of active acute and grave accents	10
Load <code>french.ldf</code> when it is found instead of <code>francais.ldf</code>	9
Load language definition files <i>after</i> the check for the hyphenation patterns	8
Merged <code>glyphs.dtx</code> into this file	1
<code>\initiate@active@char</code> : Skip the language-level active char with argument if no shorthands with arguments were defined	23
Skip the user-level active char with argument if no shorthands with arguments were defined	23
<code>\loadlocalcfg</code> : Added macro	40
<code>\pageref</code> : use a different control sequence while making <code>\ref</code> and <code>\pageref</code> robust	39
<code>otherlanguage</code> : environment added	15
babel 3.5e	
<code>\foreignlanguage</code> : Use <code>\relax</code> instead of <code>\undefined</code>	15
<code>otherlanguage</code> : changed name	15
bahasa-0.9c	
"General": Now use <code>\@patterns</code> to produce the warning	157
Removed the use of <code>\filedate</code> and moved identification after the loading of <code>babel.def</code>	157
bahasa-0.9d	
"General": Use <code>\main@language</code> instead of <code>\selectlanguage</code>	158
bahasa-1.0b	
<code>\captionsbahasa</code> : Added <code>\proofname</code> for AMS- $\LaTeX$	158
"General": Added loading of configuration file	158
breton-1.0	
"General": First release	63
breton-1.0b	
<code>\captionsbreton</code> : Added <code>\proofname</code> for AMS- $\LaTeX$	64

"General": Added loading of configuration file	66
breton-1.0c	
"General": Postpone the declaration of the TextCompositeCommands until \AtBeginDocument	66
breton-4.6	
\noextrasbreton: Use the new mechanism for dealing with active chars	64
catalan-1.1	
\captionscatalan: \headpagename should be \pagename	93
catalan-2.0	
\captionscatalan: Added some names	93
\extrascatalan: Macro completely rewritten	94
"General": Removed code to load latexhax.com	92
\noextrascatalan: Macro completely rewritten	94
catalan-2.0b	
"General": Incorporated the changes from spanish.sty	92
catalan-2.1	
"General": Update for L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub>	92
catalan-2.1d	
\captionscatalan: Added a few missing translations	93
"General": Now use \@nopatterns to produce the warning	93
Removed the use of \filedate and moved identification after the loading of babel.def	92
\textacute: Renamed from \acute as that is a \mathaccent	95
catalan-2.2a	
\extrascatalan: Handling of active characters completely rewritten	94
"General": All the code to deal with active characters is now in babel.def	96
\noextrascatalan: All the code for handling active characters is now moved to babel.def	94
catalan-2.2b	
\captionscatalan: Added \proofname for AMS-L <sup>A</sup> T <sub>E</sub> X	93
\datecatalan: Month names in lowercase	94
"General": Added loading of configuration file	98
Changed mathmode definition of acute shorthands to expand to a single prime followed by the next character in the input	97
Made the activation of the grave and acute accents optional	92
\Lgem: Added support for typing the catalan “ela geminada” with the macros \Lgem and \Lgem	97
\noextrascatalan: Make activating the accent characters optional	94
\up: Added definition of macro \up, which can be used to type ordinals	98
catalan-2.2c	
"General": Added ’ as an axtra shorthand, removed `n as a shorthand	97
Added shorthands for guillemets	96
cedile now produced by double quote shorthand	96
Removed the use of the tilde for catalan	92
catalan-2.2d	
\captionscatalan: added translation of Proof	93
croatian-1.0a	
\atcatcode: Modified handling of catcode of @ again.	128
"General": Added \let \originalTeX\relax to test for existence	128
Added reset of catcode of @ before \endinput.	128
Modified handling of catcode of @-sign.	129
Renamed babel.sty in babel.com	128

croatian-1.0b	
\atcatcode: Removed use of \makeatletter and hence the need to load latexhax.com	128
"General": Removed code to load latexhax.com	128
Removed use of \ifundefined	128
croatian-1.0c	
"General": Removed some typos	128
croatian-1.1	
\captionscroatian: Added \seename, \alsoname and \prefacename	129
"General": \originalTeX should be expandable, \let it to \empty	128
Added \relax after the argument of \input	128
Added a warning when no hyphenation patterns were loaded.	128
Brought up-to-date with babel 3.2a	128
croatian-1.2	
\captionscroatian: \headpagename should be \pagename	129
croatian-1.3	
"General": Update for L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub>	128
croatian-1.3d	
\captionscroatian: Added a few translations	129
croatian-1.3e	
\captionscroatian: Added \proofname for AMS-L <sup>A</sup> T <sub>E</sub> X	129
"General": Added loading of configuration file	129
czech-1.0a	
\atcatcode: Modified handling of catcode of @ again.	131
"General": Added \let\originalTeX \relax to test for existence	131
Added reset of catcode of @ before \endinput.	131
Modified handling of catcode of @-sign.	133
Renamed babel.sty in babel.com	131
czech-1.0b	
\atcatcode: Removed use of \makeatletter and hence the need to load latexhax.com	131
"General": Removed use of \ifundefined	131
czech-1.1	
\captionsczech: Added \seename, \alsoname and \prefacename	132
"General": \originalTeX should be expandable, \let it to \empty	131
Added \relax after the argument of \input	131
Added a warning when no hyphenation patterns were loaded.	131
Brought up-to-date with babel 3.2a	131
czech-1.1a	
\noextrasczech: Removed typo, \q was restored twice, once too many.	132
czech-1.2	
"General": Included some features from Kasal's czech.sty	131
czech-1.3	
"General": Update for L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub>	131
czech-1.3d	
"General": Now use \@nopatterns to produce the warning	131
Removed the use of \filedate and moved identification after the loading of babel.def	131
czech-1.3e	
\noextrasczech: now use \bbl@frenchspacing and \bbl@nonfrenchspacing	132
Use L <sup>A</sup> T <sub>E</sub> X's \v and \r accent commands	132
czech-1.3f	
\captionsczech: Added \proofname for AMS-L <sup>A</sup> T <sub>E</sub> X	132
"General": Added loading of configuration file	133

danish-1.0a	
\atcatcode: Modified handling of catcode of @ again. . . . .	107
"General": Added \let\originalTeX\relax to test for existence . . . . .	108
Added reset of catcode of @ before \endinput. . . . .	107
Modified handling of catcode of @-sign. . . . .	109
Renamed babel.sty in babel.com . . . . .	107
danish-1.0b	
\atcatcode: Removed use of \makeatletter and hence the need to load	
latexhax.com . . . . .	107
"General": Removed code to load latexhax.com . . . . .	107
Removed use of \ifundefined . . . . .	107, 108
danish-1.1	
\captionsdanish: Added \seename, \alsoname and \prefacename . . . . .	108
"General": \originalTeX should be expandable, \let it to \empty . . . . .	108
Added \relax after the argument of \input . . . . .	107
Added a warning when no hyphenation patterns were loaded. . . . .	108
Brought up-to-date with babel 3.2a . . . . .	107
danish-1.2	
\captionsdanish: \headpagename should be \pagename . . . . .	108
danish-1.2.2	
\captionsdanish: Added a few translations . . . . .	108
danish-1.3	
"General": Update for L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub> . . . . .	107
danish-1.3a	
\datedanish: Added ‘.’ to definition of \today . . . . .	108
danish-1.3c	
\captionsdanish: Included some revisions from Peter Busk Larsen . . . . .	108
danish-1.3f	
"General": Now use \@nopatterns to produce the warning . . . . .	108
Removed the use of \filedate and moved identification after the loading of	
babel.def . . . . .	107
danish-1.3g	
"General": Added teh active double quote character as suggested by Peter Busk	
Laursen . . . . .	107
danish-1.3h	
\captionsdanish: Added \proofname for AMS-L <sup>A</sup> T <sub>E</sub> X . . . . .	108
\extrasdanish: Added \bbl@frenchspacing . . . . .	108
"General": Added loading of configuration file . . . . .	69, 109
\noextrasdanish: Added \bbl@nonfrenchspacing . . . . .	108
dutch-2.0a	
"General": Added checking of format . . . . .	50
dutch-2.0b	
"General": Added extrasdutch . . . . .	50
dutch-2.0c	
"General": Added grqq macros . . . . .	50
dutch-2.1	
"General": reflect change to version 2.1 in babel and changes in german v2.3 . . . . .	50
dutch-2.1a	
"General": Incorporated Nico’s comments . . . . .	50
dutch-2.1b	
"General": Incorporated more comments by Nico . . . . .	50
dutch-2.1c	
"General": Fixed some typos . . . . .	50

dutch-2.2	
"General": Fixed problem with the use of " in moving arguments while " is active	50
dutch-2.3	
\@trema: \dieresis instead of \accent127	52
"General": \dieresis instead of \accent127	52
When using PostScript fonts with the Adobe font-encoding, the dieresis-accent is located elsewhere, modified code	50
\noextradutch: Added \dieresis	52
dutch-2.3a	
"General": Modified the documentation somewhat	50
dutch-3.0	
"General": Modified for babel 3.0	50
New check before loading babel.com	51
Now use \adddialect if language undefined	51
dutch-3.0a	
"General": Removed some problems in change log	50
dutch-3.0b	
\extradutch: added some comment chars to prevent white space	52
\noextradutch: added some comment chars to prevent white space	52
dutch-3.1	
"General": Add a check for existence \originalTeX	51
Removed bug found by van der Meer	50
dutch-3.1a	
\atcatcode: Made test of catcode of @ more robust	50
\captionsdutch: \pagename should be \headpagename	51
Removed \global definitions	51
\datedutch: Removed \global definitions	52
\extradutch: Removed \global definitions	52
\noextradutch: Removed \global definitions	52
dutch-3.2	
\extradutch: Save all redefined macros	52
"General": added case for "y and "Y	52
\noextradutch: Try to restore everything to its former state	52
dutch-3.2a	
\atcatcode: Modified handling of catcode of @ again.	50
"General": Added \let\originalTeX \relax to test for existence	51
Added reset of catcode of @ before \endinput.	50
Modified handling of catcode of @-sign.	53
Renamed babel.sty in babel.com	50
dutch-3.2b	
"General": removed typo (allowhpyhens)	52
dutch-3.2c	
\atcatcode: Removed use of \makeatletter and hence the need to load latexhax.com	50
"General": Removed code to load latexhax.com	50
removed use of \@ifundefined	50, 51
dutch-3.3	
\extradutch: Macro complete rewritten	52
"General": Rewritten parts of the code to use the new features of babel version 3.1	50
\noextradutch: Macro complete rewritten	52
dutch-3.3a	
\@trema: renamed \@umlaut to \@trema	52
\captionsdutch: added \seename and \alsoname	51

"General": Added <code>\save@sf@q</code> macro from <code>germanb</code> and rewrote all quote macros to use it	52
Moved code to the beginning of the file and added <code>\selectlanguage</code> call	50
dutch-3.3b	
<code>\captionsdutch</code> : added <code>\prefacename</code>	51
<code>\extrasdutch</code> : modified handling of <code>\dospecials</code> and <code>\@sanitize</code>	52
"General": Added warning, if no dutch patterns loaded	51
Set <code>\originalTeX</code> to <code>\empty</code> , because it should be expandable.	51
<code>\noextrasdutch</code> : modified handling of <code>\dospecials</code> and <code>\@sanitize</code>	52
dutch-3.4a	
"General": Added <code>\relax</code> after the argument of <code>\input</code>	51
dutch-3.4b	
"General": moved definition of <code>\allowhyphens</code> , <code>\set@low@box</code> and <code>\save@sf@q</code> to <code>babel.com</code>	52
dutch-3.5	
<code>\captionsdutch</code> : <code>\headpagename</code> should be <code>\pagename</code>	51
dutch-3.6	
"General": Update or <code>LaTeX2e</code>	50
dutch-3.6c	
"General": Now use <code>\@nopatterns</code> to produce the warning	51
Removed the use of <code>\filedate</code> , moved identification after the loading of <code>babel.def</code>	50
dutch-3.7a	
<code>\dutchhyphenmins</code> : use <code>\dutchhyphenmins</code> to store the correct values	52
"General": Moved identification code to the top of the file	50
Moved the definition of <code>\ij</code> and <code>\IJ</code> to <code>glyphs.def</code>	52
moved the definition of the double quote character at the baseline to <code>glyphs.def</code>	52
Now use <code>\Declaredq</code> to define the functions of the active double quote	52
Removed <code>\dlqq</code> , <code>\@dlqq</code> , <code>\drqq</code> , <code>\@drqq</code> and <code>\dieresis</code>	52
Rewrote the code with respect to the active double quote character	50
The support macros for the active double quote have been moved to <code>babel.def</code>	52
Use <code>\ddot</code> instead of <code>\@MATHUMLAUT</code>	52
Use <code>\main@language</code> instead of <code>\selectlanguage</code>	53
Use more general mechanism of <code>\declare@shorthand</code>	52
<code>\IJ</code> : Changed the kerning in the faked <code>ij</code> to match the <code>dc</code> -version of it	30
dutch-3.7b	
"General": Added <code>"</code> shorthand	53
dutch-3.7c	
<code>\captionsdutch</code> : We need the <code>"</code> to be active while defining <code>\captionsdutch</code>	51
dutch-3.7d	
<code>\captionsdutch</code> : Added <code>\proofname</code> for <code>AMS-L<sup>A</sup>T<sub>E</sub>X</code>	51
"General": Added loading of configuration file	53
english-2.0a	
"General": Added checking of format	54
english-2.1	
"General": Reflect changes in <code>babel 2.1</code>	54
english-2.1a	
"General": Incorporated Nico's comments	54
english-2.1b	
"General": merged <code>USenglish.sty</code> into this file	54

english-2.1c	
"General": fixed typo in definition for american language found by Werenfried Spit (nspit@fys.ruu.nl)	54
english-2.1d	
"General": Fixed some typos	54
english-3.0	
"General": Modified for babel 3.0	54
New check before loading <code>babel.com</code>	54
Now use <code>\adddialect</code> for american	55
Now use <code>\adddialect</code> if language undefined	54
english-3.0a	
"General": Add a check for existence <code>\originalTeX</code>	54
Removed bug found by van der Meer	54
english-3.0b	
<code>\atcatcode</code> : Made test of catcode of <code>@</code> more robust	54
<code>\captionenglish</code> : <code>\pagename</code> should be <code>\headpagename</code>	55
Removed <code>\global</code> definitions	55
<code>\dateamerican</code> : Removed <code>\global</code> definitions	56
<code>\dateenglish</code> : Removed <code>\global</code> definitions	55
"General": Removed <code>\global</code> definitions	55
english-3.0c	
<code>\atcatcode</code> : Modified handling of catcode of <code>@</code> again.	54
"General": Added <code>\let\originalTeX\relax</code> to test for existence	54
Added reset of catcode of <code>@</code> before <code>\endinput</code> .	54
Modified handling of catcode of <code>@</code> -sign.	56
Renamed <code>babel.sty</code> in <code>babel.com</code>	54
english-3.0d	
<code>\atcatcode</code> : Removed use of <code>\makeatletter</code> and hence the need to load <code>latexhax.com</code>	54
"General": Removed code to load <code>latexhax.com</code>	54
removed use of <code>\@ifundefined</code>	54
english-3.1	
"General": Rewrote parts of the code to use the new features of babel version 3.1	54
english-3.1a	
<code>\captionenglish</code> : added <code>\seename</code> and <code>\alsoname</code>	55
"General": Moved code to the beginning of the file and added <code>\selectlanguage</code> call	54
english-3.1b	
<code>\captionenglish</code> : added <code>\prefacename</code>	55
"General": <code>\originalTeX</code> should be expandable, <code>\let</code> it to <code>\empty</code>	54
english-3.1c	
"General": Added <code>\relax</code> after the argument of <code>\input</code>	54
english-3.2	
<code>\captionenglish</code> : <code>\headpagename</code> should be <code>\pagename</code>	55
english-3.3	
"General": Update or $\LaTeX 2\epsilon$	54
english-3.3c	
"General": Now use <code>\@nopatterns</code> to produce the warning	54
Removed the use of <code>\filedate</code> and moved the identification after the loading of <code>babel.def</code>	54
english-3.3e	
<code>\captionenglish</code> : Added <code>\proofname</code> for AMS- $\LaTeX$	55
"General": Added loading of configuration file	56

english-v3.3d	
"General": Only define american as a dialect when no separate patterns have been loaded	55
esperant-1.4f	
"General": Corrected typos (PR1652)	46
esperanto-1.	
\dateesperanto: Removed the capitals from \today	47
esperanto-1.0a	
\atcatcode: Modified handling of catcode of @ again.	46
"General": Added \let\originalTeX\relax to test for existence	47
Added reset of catcode of @ before \endinput.	46
Modified handling of catcode of the @-sign.	49
Renamed babel.sty in babel.com	46
esperanto-1.0b	
\atcatcode: Removed use of 'makeatletter and hence the need to load latexhax.com	46
"General": Removed use of \ifundefined	46
Removed use of \makeatletter	47
esperanto-1.1	
\captionesperanto: Added \seename, \alsoname and \prefacename	47
"General": \originalTeX should be expandable, \let it to \empty	47
Added \relax after the argument of \input	46
Added a warning when no hyphenation patterns were loaded.	47
Brought up-to-date with babel 3.2a	46
esperanto-1.2	
"General": Included code from esperant.sty	46
esperanto-1.3	
\captionesperanto: \headpagename should be \pagename	47
Repaired a number of mistakes, indicated by D. Ederveen	47
esperanto-1.4a	
\captionesperanto: added missing closing brace	47
"General": Updated for L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub>	46
esperanto-1.4d	
"General": Removed the use of \filedate, moved Identification after loading of babel.def	46
Use \@nopatterns for the warning	47
esperanto-1.4e	
"General": Moved identification code to the top of the file	46
esperanto-1.4g	
\captionesperanto: Added \proofname for AMS-L <sup>A</sup> T <sub>E</sub> X	47
"General": Added loading of configuration file	48
estonian-1.0b	
"General": corrected typos	123
estonian-1.0c	
\captionsestonian: Added \proofname for AMS-L <sup>A</sup> T <sub>E</sub> X	124
"General": Added loading of configuration file	127
finnish-1.0a	
\atcatcode: Modified handling of catcode of @ again.	116
"General": Added \let\originalTeX\relax to test for existence	117
Added reset of catcode of @ before rns were loaded.	116
Modified handling of catcode of @-sign.	119
Renamed babel.sty in babel.com	116

finnish-1.0b	
\atcatcode: Removed use of \makeatletter and hence the need to load latexhax.com	116
"General": Removed code to load latexhax.com	116
Removed use of \@ifundefined	117
finnish-1.1	
\captionsfinnish: \headpagename should be \pagename	117
Added \seename, \alsoname and \prefacename	117
"General": \originalTeX should be expandable, \let it to \empty	117
Added \relax after the argument of \input	116
Added a warning when no hyphenation patterns were loaded.	117
Brought up-to-date with babel 3.2a	116
finnish-1.1.2	
\captionsfinnish: Added translations	117
finnish-1.2	
"General": Update for L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub>	116
finnish-1.3c	
"General": Now use \@nopatterns to produce the warning	117
Removed the use of \filedate and moved identification after the loading of babel.def	116
finnish-1.3d	
"General": Removed a few references to babel.com	116
finnish-1.3e	
\datefinnish: Added a‘.’ after the number of the day	117
finnish-1.3f	
\finishhyphenmins: use \finnishhyphenmins to store the correct values	119
\noextrasfinnish: Added the setting of \frenchspacing	118
Added the setting of more hyphenation parameters, according to PR1027	118
finnish-1.3g	
\--: Added change of \-	119
\captionsfinnish: Added \proofname for AMS-L <sup>A</sup> T <sub>E</sub> X	117
"General": Added loading of configuration file	119
\noextrasfinnish: Added the active double quote	118
français-4.6c	
"General": Postpone the declaration of the TextCompositeCommands until \AtBeginDocument	77
french-2.0a	
"General": Added checking of format	74
french-2.1	
"General": Reflect changes in babel 2.1	74
french-2.1a	
"General": Incorporated Nico’s comments	74
french-2.1b	
"General": Fixed some typos	74
french-2.2c	
"General": Modified the documentation somewhat	74
french-3.0	
"General": Modified for babel 3.0	74
New check before loading babel.com	74
Now use \adddialect if language undefined	74
french-3.0a	
"General": removed use of \setlanguage	74
french-3.0b	
"General": renamed from french, including all control sequences	74

french-4.0	
"General": included code from GUTenberg french.sty	74
french-4.1	
"General": Add a check for existence \originalTeX	74
Removed bug found by van der Meer	74
french-4.1a	
\atcatcode: Made test of catcode of @ more robust	74
\captionsfrançais: Removed \bsl global definitions	75
\datefrançais: Removed \global definitions	75
french-4.2a	
\atcatcode: Modified handling of catcode of @ again.	74
"General": Added \let\originalTeX\relax to test for existence	74
Added reset of catcode of @ before \endinput.	74
Modified handling of catcode of @-sign.	78
Renamed babel.sty in babel.com	74
french-4.2e	
\atcatcode: Removed use of \makeatletter and hence the need to load latexhax.com	74
"General": Removed code to load latexhax.com	74
Removed use of \@ifundefined	74
french-4.2f	
\captionsfrançais: Added \seename, \alsoname and \prefacename	75
"General": \originalTeX should be expandable, \let it to \empty	74
Added \relax after the argument of \input	74
Added a warning when no hyphenation patterns were loaded.	74
Brought up-to-date with babel 3.2a	74
Removed crossreferencing macros as they are dealt with in babel.com	74
french-4.3	
\extrasfrançais: Completely rewrote macro	75
french-4.3a	
\noextrasfrançais: Removed spurious \endgroup and }	76
french-4.3b	
\french@sh@: Replaced \, with \thinspace to make it work with plain TeX.	76
\french@sh@;: Replaced \, with \thinspace to make it work with plain TeX.	76
\french@sh@?: Replaced \, with \thinspace to make it work with plain TeX.	77
\french@sh@@: Replaced \, with \thinspace to make it work with plain TeX.	76
french-4.4	
\captionsfrançais: \headpagename should be \pagename	75
french-4.5	
"General": Update for LaTeX2e	74
french-4.5c	
"General": Now use \@nopatterns to produce the warning	74
Removed the use of \filedate and moved the identification after the loading of babel.def	74
french-4.6a	
\french@sh@: Use new \DefineActiveNoArg	76
\french@sh@;: Use new \DefineActiveNoArg	76
Use the more general mechanism of \declare@shorthand	76
\french@sh@?: Use new \DefineActiveNoArg	77
Use the more general mechanism of \declare@shorthand	77
\french@sh@@: Use new \DefineActiveNoArg	76
Use the more general mechanism of \declare@shorthand	76
\noextrasfrançais: now use \bbl@frenchspacing and \bbl@nonfrenchspacing	76
Use the new mechanism for dealing with active chars	75

french-4.6b	
<code>\system@sh@;</code> : Added system level shorthands	77
french-4.6c	
<code>\captionsfrancais</code> : Added <code>\proofname</code> for AMS- $\LaTeX$	75
"General": Added loading of configuration file	78
Also allow the hyphenation patterns to be loaded for 'french'	74
galician-1.1	
"General": Update for $\LaTeX 2\epsilon$	99
galician-1.1c	
"General": Now use <code>\@nopatterns</code> to produce the warning	100
Removed the use of <code>\filedate</code> and moved identification after the loading of	
<code>babel.def</code>	99
<code>\textacute</code> : Renamed from <code>\acute</code> as that is a <code>\mathaccent</code>	101
<code>\texttilde</code> : Renamed from <code>\tilde</code> as that is a <code>\mathaccent</code>	101
galician-1.1d	
<code>\captionsgalician</code> : Added a few missing translations	100
galician-1.2a	
<code>\extrasgalician</code> : Handling of active characters completely rewritten	101
<code>\noextrasgalician</code> : All the code for handling active characters is now moved to	
<code>babel.def</code>	101
galician-1.2b	
<code>\captionsgalician</code> : Added <code>\proofname</code> for AMS- $\LaTeX$	100
"General": Added loading of configuration file	103
Changed mathmode definition of acute shorthands to expand to a single prime	
followed by the next character in the input	102
galician-1.2c	
<code>\noextrasgalician</code> : Make active accent optional	101
galician1.1d	
<code>\dategalician</code> : Corrected the name of the month marzo from marzal	100
german-2.6a	
"General": Moved all quotation characters to <code>glyphs.dtx</code>	60
Use <code>\ddot</code> instead of <code>\@MATHUMLAUT</code>	60
german-2.6b	
<code>\captionsgerman</code> : Added <code>\proofname</code> for AMS- $\LaTeX$	59
"General": Added loading of configuration file	61
germanb-1.0a	
"General": Incorporated Nico's comments	57
germanb-1.0b	
"General": fixed typo in definition for austrian language found by Werenfried	
Spit <code>nspit@fys.ruu.nl</code>	57
germanb-1.0c	
"General": Fixed some typos	57
germanb-1.1	
"General": When using PostScript fonts with the Adobe fontencoding, the	
dieresis-accent is located elsewhere, modified code	57
<code>\noextrasgerman</code> : Added <code>\dieresis</code>	59
germanb-1.1a	
"General": Modified the documentation somewhat	57
germanb-2.0	
"General": Modified for babel 3.0	57
New check before loading <code>babel.com</code>	58
Now use <code>\adddialect</code> for austrian	58
Now use <code>\adddialect</code> if language undefined	58

germanb-2.0a	
"General": Removed some problems in change log	57
germanb-2.0b	
\extrasgerman: added some comment chars to prevent white space	59
\noextrasgerman: added some comment chars to prevent white space	59
germanb-2.1	
"General": Add a check for existence of \originalTeX	58
Removed bug found by van der Meer	57
germanb-2.2	
\atcatcode: Made test of catcode of @ more robust	58
\captionsgerman: \pagename should be \headpagename	59
Removed \global definitions	59
\extrasgerman: Save all redefined macros	59
"General": Removed global assignments, brought uptodate with german.tex	
v2.3d	57
\noextrasgerman: Try to restore everything to its former state	59
germanb-2.2a	
\atcatcode: Modified handling of catcode of @ again.	58
"General": Added \let\originalTeX\relax to test for existence	58
Added reset of catcode of @ before \endinput.	58
Modified handling of catcode of @-sign.	62
Renamed babel.sty in babel.com	57
germanb-2.2d	
\atcatcode: Removed use of \makeatletter and hence the need to load	
latexhax.com	58
"General": Removed code to load latexhax.com	57
Removed use of \ifundefined	58
germanb-2.3	
"General": Rewritten parts of the code to use the new features of babel version	
3.1	57
germanb-2.3e	
\captionsgerman: Added \prefacename, \seename and \alsoname	59
\dategerman: Added \month@german	59
"General": Added \save@sf@q macro and rewrote all quote macros to use it	60
Added warning, if no german patterns loaded	58
Brought up-to-date with german.tex v2.3e (plus some bug fixes) [br]	57
Moved code to the beginning of the file and added \selectlanguage call	58
germanb-2.3f	
"General": Set \originalTeX to \bbl empty, because it should be expandable.	58
germanb-2.3g	
"General": Added \relax after the argument of \input	58
germanb-2.3h	
"General": moved definition of \allowhyphens, \set@low@box and \save@sf@q	
to babel.com	60
germanb-2.4	
\captionsgerman: \headpagename should be \pagename	59
germanb-2.5	
"General": Update or L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub>	57
germanb-2.5c	
"General": Now use \@nopatterns to produce the warning	58
Removed the use of \filedate and moved the identification after the loading	
of babel.def	57

germanb-2.6a	
"General": \umlautlow and \umlauthigh moved to <code>glyphs.dtx</code> , as well as	
\newumlaut (now \lower@umlaut	60
Moved the identification to the top of the file	57
Rewrote the code that handles the active double quote character	57
\noextrasgerman: All the code to handle the active double quote has been moved	
to <code>babel.def</code>	60
Removed <code>\3</code> as it is no longer in <code>german.ldf</code>	60
use <code>\germanhyphenmins</code> to store the correct values	60
hyphen-1.0b	
\iflanguage: Added <code>\@bsphack</code> and <code>\@esphack</code>	13
\selectlanguage: Added <code>\@bsphack</code> and <code>\@esphack</code>	14
hyphen-1.0c	
\selectlanguage: Made <code>\selectlanguage</code> robust	13
Replaced <code>\gdef</code> with <code>\def</code>	14
hyphen-1.0d	
\iflanguage: Added comment character after <code>#2</code>	13
hyphen-1.0e	
\iflanguage: Removed superfluous <code>\expandafter</code>	13
\process@language: Removed superfluous <code>\expandafter</code>	18
\selectlanguage: Removed superfluous <code>\expandafter</code>	14
hyphen-1.0f	
\process@language: Reinserted <code>\expandafter</code>	18
hyphen-1.0h	
\@nopatterns: Added a percent sign to remove unwanted white space	16
\iflanguage: Removed space hacks and use of <code>\@ifundefined</code>	13
\selectlanguage: Removed space hacks and use of <code>\@ifundefined</code>	14
hyphen-1.0i	
\patterns@loaded: Added a token register for collecting the names of patterns	
that are loaded by <code>iniTeX</code> .	17
\process@language: Added the collection of pattern names.	18
hyphen-1.1	
"General": Added redefinition of <code>\dump</code> to add a message to <code>\everyjob</code>	20
Removed the extra <code>if</code> control sequence	19
Removed use of <code>\toks0</code>	19
\selectlanguage: <code>\originalTeX</code> should only be executed once	14
hyphen-1.1a	
\adddialect: Added <code>\relax</code>	13
\addlanguage: Added a <code>%</code> , removed by	12
\iflanguage: Refrased <code>\ifnum</code> test	13
\selectlanguage: Modified to allow arguments that start with an escape character	13
hyphen-1.1b	
\selectlanguage: Added <code>\relax</code> as first command to stop an expansion if	
<code>\protect</code> is empty	14
Added three <code>\expandafters</code> to save macro space for <code>\originalTeX</code>	14
Simplified the modification to allow the use in a <code>\write</code> command	13
hyphen-1.1c	
"General": Free macro space for <code>\process@language</code>	19
\selectlanguage: Moved definition of <code>\originalTeX</code> before <code>\extras&lt;lang&gt;</code>	14
hyphen-1.1d	
"General": Set <code>\originalTeX</code> to <code>\empty</code> , because it should be expandable.	16
\selectlanguage: Set <code>\originalTeX</code> to <code>\empty</code> , because it should be expandable.	14

irish-1.0c	
\captionssirish: Added \proofname for AMS- $\LaTeX$	69
italian-0.99	
"General": First version, from english.doc	79
italian-1.0	
"General": Modified for babel 3.0	79
New check before loading babel.com	79
Now use \adddialect if language undefined	79
italian-1.0a	
"General": removed typo	79
italian-1.0b	
"General": Add a check for existence \originalTeX	79
Removed bug found by van der Meer	79
italian-1.0c	
\atcatcode: Made test of catcode of @ more robust	79
\captionssitalian: \pagename should be \headpagename	80
Removed \global definitions	80
\dateitalian: Removed \global definitions	80
italian-1.0d	
\captionssitalian: ‘contine’ substituted by ‘Allegati’ as suggested by Marco Bozzo (BOZZO@CERNVM).	80
italian-1.0e	
\atcatcode: Modified handling of catcode of @ again.	79
"General": Added \let\originalTeX\relax to test for existence	79
Added reset of catcode of @ before \endinput.	79
Modified handling of catcode of @-sign.	81
Renamed babel.sty in babel.com	79
italian-1.0f	
\atcatcode: fixed typo, missing right brace	79
italian-1.0h	
\atcatcode: Removed use of \makeatletter and hence the need to load latexhax.com	79
"General": Removed code to load latexhax.com	79
Removed use of \@ifundefined	79
italian-1.1	
\captionssitalian: \headpagename should be \pagename	80
Added \seename, \alsiname and \prefacename	80
"General": \originalTeX should be expandable, \let it to \empty	79
Added \relax after the argument of \input	79
Added a warning when no hyphenation patterns were loaded.	79
Brought up-to-date with babel 3.2a	79
italian-1.2	
"General": Update for LaTeX <sub>E</sub>	79
italian-1.2b	
\captionssitalian: Changed some of the words following suggestions from Claudio Beccari	80
\italianhyphenmins: Added setting of left and righthyphenmin according to Claudio Beccari’s suggestion	80
\noextrasitalian: Added setting of club- and widowpenalty	80
Added setting of finallyphendemerits	80
italian-1.2e	
"General": Now use \@nopatterns to produce the warning	79
Removed the use of \filedate and moved identification after the loading of babel.def	79

italian-1.2f	
"General": Updated for babel 3.5	79
italian-1.2g	
\captionsofitalian: Added \proofname for AMS- $\LaTeX$	80
"General": Added loading of configuration file	81
itish-1.0b	
"General": Corrected typo (PR1652)	68
Lsorbian-0.1	
"General": First version	146
lsorbian-1.0b	
\captionsofLsorbian: Added \proofname for AMS- $\LaTeX$	147
"General": Added loading of configuration file	148
magyar-1.0a	
\atcatcode: Modified handling of catcode of @ again.	120
"General": Added \let\originalTeX\relax to test for existence	120
Added reset of catcode of @ before \endinput.	120
Modified handling of catcode of @-sign.	122
Renamed babel.sty in babel.com	120
magyar-1.0b	
\atcatcode: Removed use of \makeatletter and hence the need to load	
latexhax.com	120
"General": Removed code to load latexhax.com	120
Removed use of \@ifundefined	120
magyar-1.1	
\captionsofmagyar: \headpagename should be \pagename	121
Added \seename, \alsoname and \prefacename	121
"General": \originalTeX should be expandable, \let it to \empty	120
Added \relax after the argument of \input	120
Added a warning when no hyphenation patterns were loaded.	120
Brought up-to-date with babel 3.2a	120
magyar-1.1.3	
\captionsofmagyar: Added translations, fixed typos	121
\ondatemagyar: The date number should not be followed by a dot.	122
magyar-1.1.4	
\datemagyar: Rewritten to produce the correct date format	121
"General": Further spelling corrections	120
\ondatemagyar: Renamed from \datemagyar; no longer redefines \today.	122
magyar-1.1.5	
"General": Still more spelling corrections	120
magyar-1.2	
"General": Update for $\LaTeX 2\epsilon$	120
magyar-1.3c	
"General": Now use \nopatterns to produce the warning	120
Removed the use of \filedate and moved identification after the loading of	
babel.def	120
magyar-1.3e	
\captionsofmagyar: Added \proofname for AMS- $\LaTeX$	121
"General": Added loading of configuration file	122
norsk-1.0a	
\atcatcode: Modified handling of catcode of @ again.	110
"General": Added \let\originalTeX\relax to test for existence	110

Added reset of catcode of @ before <code>\endinput</code> .	110
Modified handling of catcode of @-sign.	112
Renamed <code>babel.sty</code> in <code>babel.com</code>	110
norsk-1.0c	
<code>\atcatcode</code> : Removed use of <code>\makeatletter</code> and hence the need to load <code>latexhax.com</code>	110
"General": Removed code to load <code>latexhax.com</code>	110
Removed use of <code>\@ifundefined</code>	110
norsk-1.1	
<code>\captionsnorsk</code> : <code>\headpagename</code> should be <code>\pagename</code>	111
Added <code>\seename</code> , <code>\alsoname</code> and <code>\prefacename</code>	111
<code>\captionsnynorsk</code> : <code>\headpagename</code> should be <code>\pagename</code>	111
Added <code>\seename</code> , <code>\alsoname</code> and <code>\prefacename</code>	111
"General": <code>\originalTeX</code> should be expandable, <code>\let</code> it to <code>\empty</code>	110
Added <code>\relax</code> after the argument of <code>\input</code>	110
Added a warning when no hyphenation patterns were loaded.	110
Brought up-to-date with babel 3.2a	110
norsk-1.1.3	
"General": Added a couple of translations (from Per Norman Oma, TeX@itk.unit.no)	110
norsk-1.2	
"General": Update for L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub>	110
norsk-1.2d	
"General": Now use <code>\@nopatterns</code> to produce the warning	110
Removed the use of <code>\filedate</code> and moved identification after the loading of <code>babel.def</code>	110
norsk-1.2f	
<code>\captionsnorsk</code> : Added <code>\proofname</code> for AMS-L <sup>A</sup> T <sub>E</sub> X	111
"General": Added loading of configuration file	112
<code>\nynorskhyphenmins</code> : Added setting of hyphenmin parameters	111
polish-1.0a	
<code>\textpl</code> : Initially execute <code>'textpl</code>	137
polish-1.1c	
"General": Now use <code>\@nopatterns</code> to produce the warning	135
Removed the use of <code>\filedate</code> and moved identification after the loading of <code>babel.def</code>	134
polish-1.1d	
"General": The <code>dqmacro</code> for C used <code>\'c</code>	138
polish-1.2a	
"General": Don't modify <code>\rm</code> and friends for L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub> , take <code>\selectfont</code> instead	137
polish-1.2b	
<code>\captionspolish</code> : Added <code>\proofname</code> for AMS-L <sup>A</sup> T <sub>E</sub> X	135
"General": Added loading of configuration file	139
portuges-1.0a	
<code>\atcatcode</code> : Modified handling of catcode of @ again.	82
"General": Added <code>\let\originalTeX\relax</code> to test for existence	83
Added reset of catcode of @ before <code>\endinput</code> .	82
Modified handling of catcode of @-sign.	85
Renamed <code>babel.sty</code> in <code>babel.com</code>	82
portuges-1.0b	
<code>\atcatcode</code> : Removed use of <code>\makeatletter</code> and hence the need to load <code>latexhax.com</code>	82

"General": Removed code to load latexhax.com	82
Removed use of \ifundefined	82
Removed use of cs@ifundefined	83
portuges-1.1	
\captionsportuges: \headpagename should be \pagename	83
Added \seename, \alsoname and \prefacename	83
"General": \originalTeX should be expandable, \let it to \empty	83
Added \relax after the argument of \input	82
Added a warning when no hyphenation patterns were loaded.	83
Brought up-to-date with babel 3.2a	82
portuges-1.2	
"General": Update for L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub>	82
portuges-1.2d	
"General": Now use \nopatterns to produce the warning	83
Removed the use of \filedate and moved identification after the loading of babel.def	82
portuges-1.2e	
\captionsportuges: Added a few missing translations	83
portuges-1.2f	
"General": Use \main@language instead of \selectlanguage	85
portuges-1.2g	
\brasilhyphenmins: Added setting of hyphenmin values	84
\captionsbrazil: The coptions for brazilian and portuguese are different now	83
\extraspportuges: Added using some " shorthands	84
"General": Enhanced support for brasilian	82
\ord: Added macro	85
\orda: Added macro	85
\ra: Added macro	85
\ro: Added macro	85
portuges-1.2h	
\captionsportuges: Added \proofname for AMS-L <sup>A</sup> T <sub>E</sub> X	83
"General": Added loading of configuration file	85
romanian-1.0a	
\atcatcode: Modified handling of catcode of @ again.	104
"General": Added \let\originalTeX\relax to test for existence	104
Added reset of catcode of @ before \endinput.	104
Modified handling of catcode of @-sign.	105
Renamed babel.sty in babel.com	104
romanian-1.0b	
\atcatcode: Removed use of \makeatletter and hence the need to load latexhax.com	104
"General": Removed code to load latexhax.com	104
Removed use of \ifundefined	104
romanian-1.1	
\captionsromanian: \headpagename should be \pagename	105
Added \seename, \alsoname and \prefacename	105
Translation errors found by Robert Juhasz fixed	105
\dateromanian: Translation errors found by Robert Juhasz fixed	105
"General": \originalTeX should be expandable, \let it to \empty	104
Added \relax after the argument of \input	104
Added a warning when no hyphenation patterns were loaded.	104
Brought up-to-date with babel 3.2a	104

romanian-1.1b	
"General": Added translations	104
romanian-1.2	
"General": Update for LaTeX2e	104
romanian-1.2d	
"General": Now use <code>\@nopatterns</code> to produce the warning	104
Removed the use of <code>\filedate</code> and moved identification after the loading of	
<code>babel.def</code>	104
romanian-1.2e	
"General": Updated for babel release 3.5	104
romanian-1.2f	
<code>\captionsslovene</code> : Added <code>\proofname</code> for AMS-L <sup>A</sup> T <sub>E</sub> X	105
"General": Added loading of configuration file	105
scottish-1.0b	
"General": Corrected typos (PR1652)	71
scottish-1.0c	
<code>\captionsscottish</code> : Added <code>\proofname</code> for AMS-L <sup>A</sup> T <sub>E</sub> X	72
"General": Added loading of configuration file	72
slovak-1.0	
"General": First version	140
slovak-1.2	
"General": Update for L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub>	140
slovak-1.2b	
<code>\noextrasslovak</code> : Added setting of left- and righthypenmin	141
slovak-1.2d	
"General": Now use <code>\@nopatterns</code> to produce the warning	140
Removed the use of <code>\filedate</code> and moved identification after the loading of	
<code>babel.def</code>	140
slovak-1.2e	
<code>\noextrasslovak</code> : Now use <code>\slovakhypenmins</code>	141
Use L <sup>A</sup> T <sub>E</sub> X's <code>\v</code> accent command	141
slovak-1.2g	
<code>\captionsslovak</code> : Added <code>\proofname</code> for AMS-L <sup>A</sup> T <sub>E</sub> X	141
"General": Added loading of configuration file	141
slovene-1.0a	
<code>\atcatcode</code> : Modified handling of catcode of <code>@</code> again.	143
"General": Added <code>\let csoriginalTeX\relax</code> to test for existence	144
Added reset of catcode of <code>@</code> before <code>\endinput</code> .	143
Modified handling of catcode of <code>@</code> -sign.	145
Renamed <code>babel.sty</code> in <code>babel.com</code>	143
slovene-1.0b	
<code>\atcatcode</code> : Removed use of <code>\makeatletter</code> and hence the need to load	
<code>latexhax.com</code>	143
"General": Removed code to load <code>latexhax.com</code>	143
Removed use of <code>\@ifundefined</code>	143, 144
slovene-1.1	
<code>\captionsslovene</code> : <code>\headpagename</code> should be <code>\pagename</code>	144
Added <code>\seename</code> , <code>\alsoname</code> and <code>\prefacename</code>	144
"General": <code>\originalTeX</code> should be expandable, <code>\</code> it to <code>\empty</code>	144
Added <code>\relax</code> after the argument of <code>\input</code>	143
Added a warning when no hyphenation patterns were loaded.	144
Brought up-to-date with babel 3.2a	143

slovene-1.2	
"General": Update for L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub>	143
slovene-1.2b	
\captionsslovene: Added extra translations from Josef Leydold, leydold@statrix2.wu-wien.ac.at	144
slovene-1.2d	
"General": Now use \@nopatterns to produce the warning	144
Removed the use of \filedate and moved identification after the loading of	
babel.def	143
slovene-1.2f	
\noextrasslovene: Introduced the active "	145
slovene-1.2g	
\captionsslovene: Added \proofname for AMS-L <sup>A</sup> T <sub>E</sub> X	144
"General": Added loading of configuration file	145
spanish-1.1	
"General": Date format corrected. Wrong change history deleted	86
spanish-1.1a	
"General": \I does not exist, modified	86
spanish-2.0	
"General": Modified for babel 3.0	86
Now use \adddialect if language undefined	88
spanish-2.0a	
"General": Add a check for existence \originalTeX	87
removed use of \setlanguage	86
spanish-2.0b	
"General": New check before loading babel.sty	86
New check before loading babel.com	87
spanish-2.0c	
\atcatcode: Made test of catcode of @ more robust	87
\captionsspanish: Removed \global definitions	88
\datespanish: Removed csglobal definitions	88
spanish-2.0d	
\datespanish: Capitalize months as suggested by E. Torrente (TORRENTE@CERNVM).	88
spanish-2.1	
"General": Added catalan as a 'dialect'	86
spanish-2.1a	
\atcatcode: Modified handling of catcode of @ again.	87
"General": Added \let\originalTeX \relax to test for existence	87
Added reset of catcode of @ before \endinput.	87
Modified handling of catcode of @-sign.	91
Renamed babel.sty in babel.com	86
spanish-3.0	
\captionsspanish: Capitals are accented, some strings changed	88
\datespanish: Uncapitalize months, since that seems to be the correct, modern	
usage	88
\extrasspanish: Formerly empty, all code is new.	88
"General": Catalan deleted	86
Major rewriting, new macros, active accents, catalan removed	86
spanish-3.0a	
\@tilde: Added fix for \dotlessi	90
"General": Text fixed	88

spanish-3.1	
\atcatcode: Removed use of \makeatletter and hence the need to load latexhax.com	87
\captionsspanish: \headpagename should be \pagename	88
added \seename, and \alsoname and \prefacename	88
\extrasspanish: Rewrote the macro.	88
"General": Added \relax after the argument of \input	87
Added warning, if no spanish patterns were loaded	88
Brought up-to-date with babel 3.2a	86
Moved code to the beginning of the file and added \selectlanguage call	87
Removed code to load latexhax.com	87
removed use of \@ifundefined	87, 88
Set \originalTeX to \empty, because it should be expandable.	87
spanish-3.1.1	
"General": The accents had to be made active during their own definition.	
Changed address for goya.	86
spanish-3.1.2	
"General": Added address, phone and fax for Julio Sánchez. The definition of the active tilde was not being restored on exit.	86
spanish-3.2	
\@tilde: All this code is new	90
\captionsspanish: added translated strings for \seename \alsoname and \prefacename	88
\extrasspanish: Major rewrite. Now works like in germanb and dutch.	88
"General": Active character definitions changed as in germanb.	86
Changed \acute to \textacute and \tilde to \texttilde because the old names were already used for math accents.	89
Update for L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub>	86
spanish-3.3d	
"General": Now use \@nopatterns to produce the warning	88
Removed the use of \filedate and moved identification after the loading of babel.def	86
spanish-3.4a	
\extrasspanish: Yet another major rewrite	88
\noextrasspanish: All the code for handling active characters is now moved to babel.def	89
spanish-3.4b	
"General": corrected typo (PR1652)	86
spanish-3.4c	
\captionsspanish: Added \proofname for AMS-L <sup>A</sup> T <sub>E</sub> X	88
"General": Added ’ as an aextra shorthand, removed ’n as a shorthand	91
Added loading of configuration file	91
Changed mathmode definition of acute shorthands to expand to a single prime followed by the next character in the input	90
made active acute optional	86
swedish-1.0a	
\atcatcode: Modified handling of catcode of @ again.	113
"General": Added \let\originalTeX\relax to test for existence	113
Added reset of catcode of @ before \endinput.	113
Modified handling of catcode of @-sign.	115
Renamed babel.sty in babel.com	113
swedish-1.0b	
\captionsswedish: added definition for \enclname	114
made definition of \refname pluralis	114

removed type in definition of <code>\contentsname</code> .....	114
swedish-1.0c	
<code>\atcatcode</code> : Removed use of <code>\makeatletter</code> and hence the need to load <code>latexhax.com</code> .....	113
"General": Removed code to load <code>latexhax.com</code> .....	113
Removed use of <code>\@ifundefined</code> .....	113
swedish-1.1	
<code>\captionsswedish</code> : <code>\headpagename</code> should be <code>\pagename</code> .....	114
Added <code>\seename</code> , <code>\alsoname</code> and <code>\prefacename</code> .....	114
"General": <code>\originalTeX</code> should be expandable, <code>\let</code> it to <code>\empty</code> .....	113
Added <code>\relax</code> after the argument of <code>\input</code> .....	113
Added a warning when no hyphenation patterns were loaded. ....	113
Brought up-to-date with babel 3.2a .....	113
swedish-1.1b	
<code>\captionsswedish</code> : Added translations .....	114
swedish-1.2	
"General": Update for LaTeX2e .....	113
swedish-1.3d	
<code>\captionsswedish</code> : Changed <code>\aa</code> to <code>\csname aa\endcsname</code> , to make <code>\uppercase</code> do the right thing .....	114
"General": Now use <code>\@nopatterns</code> to produce the warning .....	113
Removed the use of <code>\filedate</code> and moved identification after the loading of <code>babel.def</code> .....	113
swedish-1.3e	
<code>\captionsswedish</code> : Changed <code>\alsoname</code> from 'se också' .....	114
<code>\extrasswedish</code> : Added <code>\bbl@frenchspacing</code> .....	114
"General": Update for release 3.5 .....	113
<code>\noextrasswedish</code> : Added <code>\bbl@nonfrenchspacing</code> .....	114
<code>\swedishhyphenmins</code> : use <code>\swedishhyphenmins</code> to store the correct values ..	114
swedish-1.3f	
<code>\captionsswedish</code> : Added <code>\proofname</code> for AMS-LATEX .....	114
"General": Added loading of configuration file .....	114
turkish-1.1	
<code>\captionsturkish</code> : <code>\headpagename</code> should be <code>\pagename</code> .....	154
turkish-1.2	
"General": Update for L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub> .....	153
turkish-1.2b	
<code>\captionsturkish</code> : Added braces behind <code>\i</code> in strings .....	154
<code>\dateturkish</code> : Added braces behind <code>\i</code> in strings .....	154
turkish-1.2c	
"General": Now use <code>\@nopatterns</code> to produce the warning .....	153
Removed the use of <code>\filedate</code> and moved identification after the loading of <code>babel.def</code> .....	153
turkish-1.2d	
<code>\dateturkish</code> : removed extra closing brace, <code>\mont</code> should be <code>\month</code> .....	154
turkish-1.2e	
<code>\extrasturkish</code> : Completely rewrote macro .....	154
<code>\noextrasturkish</code> : now use <code>\bbl@frenchspacing</code> and <code>\bbl@nonfrenchspacing</code> ..	155
turkish-1.2f	
"General": Added loading of configuration file .....	155
usorbian-0.1	
"General": First version .....	149

usorbian-0.1b	
"General": Made it possible to run through L <sup>A</sup> T <sub>E</sub> X; added \MF and removed extra \endmacro .....	149
usorbian-0.1c	
\captionusorbian: Removed two typos (Kapitel and Dodatki) .....	150
usorbian-1.0a	
"General": Removed stuff that has been moved to babel.def .....	151
usorbian-1.0b	
\captionusorbian: Added \proofname for AMS-L <sup>A</sup> T <sub>E</sub> X .....	150
"General": Added loading of configuration file .....	152
v1.2f	
\captionsturkish: Added \proofname for AMS-L <sup>A</sup> T <sub>E</sub> X .....	154