Babel

Version 26.16
2026/11/10

Javier Bezos
Current maintainer

Johannes L. Braams
Original author

Localization and internationalization

Plain TEX

NOTE This file is devoted to babel for Plain TEX. For many years, most of the code was shared between LATEX and Plain, but recent advances in LATEX have made joint maintenance increasingly complex. Consequently, the code for Plain will be split from main babel for LATEX and essentially frozen. This does not mean it is unmaintained, and the door remains open to future improvements.

Description of many macros are the same as those in babel-code.pdf, and here only some hints are provided. Please, refer to that document for details.

This file is loaded not only by Plain. LATEX loads it with the package option `base` (and also `plain.def`) with `\bbl@onlyswitch` = `\@empty`. See sec. 5.

It needs to be cleaned up to remove unnecessary code in Plain. Note also the engine specific files are still loaded.

1. A few preliminary definitions

Just a block. To be revised.

```
⟨⟨version=26.16⟩⟩
⟨⟨date=2026/11/10⟩⟩
```

Some files identify themselves with a LATEX macro. The following code is placed before them to define (and then undefine) if not in LATEX.

\language Just for compatibility, for not to touch `hyphen.cfg`.

```
⟨⟨*Define Plain core switching macros⟩⟩ ▸
\ifx\language\@undefined
  \csname newcount\endcsname\language
\fi
⟨⟨/Define Plain core switching macros⟩⟩
```

\last@language Another counter is used to keep track of the allocated languages. TEX and LATEX reserves for this purpose the count 19.

\addlanguage This macro was introduced for TEX < 2. Preserved for compatibility.

```
⟨⟨*Define Plain core switching macros⟩⟩ ▸
\countdef\last@language=19
\def\addlanguage{\csname newlanguage\endcsname}
⟨⟨/Define Plain core switching macros⟩⟩
```

First, exit immediately if previouly loaded.

```
⟨*core⟩
\ifx\ldf@quit\@undefined\else
\endinput\fi % Same line!
\ifx\ProvidesFile\@undefined
  \def\ProvidesFile#1[#2 #3 #4]{%
    \wlog{File: #1 #4 #3 <#2>}%
    \let\ProvidesFile\@undefined}
\fi
\ProvidesFile{babel.def}[<@date@> v<@version@> Babel common definitions]
```

2. Emulating some LATEX features

The file `babel.def` expects some definitions made in the LATEX 2ε style file. So, in Plain we must provide at least some predefined values as well some tools to set them (even if not all options are available). There are no package options, and therefore and alternative mechanism is provided. For the moment, only `\babeloptionstrings` and `\babeloptionmath` are provided, which can be defined before loading babel. `\BabelModifiers` can be set too (but not sure it works).

```
\ifx\AtBeginDocument\@undefined
\def\@empty{}
\def\loadlocalcfg#1{%
  \openin0#1.cfg
  \ifeof0
    \closein0
  \else
    \closein0
    {\immediate\write16{**************************************}%
     \immediate\write16{* Local config file #1.cfg used}%
     \immediate\write16{*}%
     }
    \input #1.cfg\relax
  \fi
  \@endofldf}
```

2.1. General tools

A number of LaTeX macro's that are needed later on.

```
\long\def\@firstofone#1{#1}
\long\def\@firstoftwo#1#2{#1}
\long\def\@secondoftwo#1#2{#2}
\def\@nnil{\@nil}
\def\@gobbletwo#1#2{}
\def\@ifstar#1{\@ifnextchar *{\@firstoftwo{#1}}}
\def\@star@or@long#1{%
  \@ifstar
  {\let\l@ngrel@x\relax#1}%
  {\let\l@ngrel@x\long#1}}
\let\l@ngrel@x\relax
\def\@car#1#2\@nil{#1}
\def\@cdr#1#2\@nil{#2}
\let\@typeset@protect\relax
\let\protected@edef\edef
\long\def\@gobble#1{}
\edef\@backslashchar{\expandafter\@gobble\string\\}
\def\strip@prefix#1>{}
\def\g@addto@macro#1#2{{%
    \toks@\expandafter{#1#2}%
    \xdef#1{\the\toks@}}}
\def\@namedef#1{\expandafter\def\csname #1\endcsname}
\def\@nameuse#1{\csname #1\endcsname}
\def\@ifundefined#1{%
  \expandafter\ifx\csname#1\endcsname\relax
    \expandafter\@firstoftwo
  \else
    \expandafter\@secondoftwo
  \fi}
\def\@expandtwoargs#1#2#3{%
  \edef\reserved@a{\noexpand#1{#2}{#3}}\reserved@a}
\def\zap@space#1 #2{%
  #1%
  \ifx#2\@empty\else\expandafter\zap@space\fi
  #2}
\let\bbl@trace\@gobble
\def\bbl@error#1{% Implicit #2#3#4
  \begingroup
    \catcode`\\=0   \catcode`\==12 \catcode`\`=12
    \catcode`\^^M=5 \catcode`\%=14
    \input errbabel.def
  \endgroup
  \bbl@error{#1}}
```

```
\def\bbl@warning#1{%
  \begingroup
    \newlinechar=`\^^J
    \def\\{^^J(babel) }%
    \message{\\#1}%
  \endgroup}
\let\bbl@infowarn\bbl@warning
\def\bbl@info#1{%
  \begingroup
    \newlinechar=`\^^J
    \def\\{^^J}%
    \wlog{#1}%
  \endgroup}
```

LaTeX 2$_\varepsilon$ has the command `\@onlypreamble` which adds commands to a list of commands that are no longer needed after `\begin{document}`.

```
\ifx\@preamblecmds\@undefined
  \def\@preamblecmds{}
\fi
\def\@onlypreamble#1{%
  \expandafter\gdef\expandafter\@preamblecmds\expandafter{%
    \@preamblecmds\do#1}}
\@onlypreamble\@onlypreamble
```

Mimic LaTeX's `\AtBeginDocument`; for this to work the user needs to add `\begindocument` to his file.

```
\def\begindocument{%
  \@begindocumenthook
  \global\let\@begindocumenthook\@undefined
  \def\do##1{\global\let##1\@undefined}%
  \@preamblecmds
  \global\let\do\noexpand}
```

```
\ifx\@begindocumenthook\@undefined
  \def\@begindocumenthook{}
\fi
\@onlypreamble\@begindocumenthook
\def\AtBeginDocument{\g@addto@macro\@begindocumenthook}
```

We also have to mimic LaTeX's `\AtEndOfPackage`. Our replacement macro is much simpler; it stores its argument in `\@endofldf`.

```
\def\AtEndOfPackage#1{\g@addto@macro\@endofldf{#1}}
\@onlypreamble\AtEndOfPackage
\def\@endofldf{}
\@onlypreamble\@endofldf
\let\bbl@afterlang\@empty
\chardef\bbl@opt@hyphenmap\z@
```

LaTeX needs to be able to switch off writing to its auxiliary files; plain doesn't have them by default. There is a trick to hide some conditional commands from the outer `\ifx`. The same trick is applied below.

```
\catcode`\&=\z@
\ifx&if@filesw\@undefined
  \expandafter\let\csname if@filesw\expandafter\endcsname
    \csname iffalse\endcsname
\fi
\catcode`\&=4
```

Mimic LaTeX's commands to define control sequences.

```
\def\newcommand{\@star@or@long\new@command}
\def\new@command#1{%
  \@testopt{\@newcommand#1}0}
\def\@newcommand#1[#2]{%
  \@ifnextchar [{\@xargdef#1[#2]}%
```

```
                {\@argdef#1[#2]}}
\long\def\@argdef#1[#2]#3{%
  \@yargdef#1\@ne{#2}{#3}}
\long\def\@xargdef#1[#2][#3]#4{%
  \expandafter\def\expandafter#1\expandafter{%
    \expandafter\@protected@testopt\expandafter #1%
    \csname\string#1\expandafter\endcsname{#3}}%
  \expandafter\@yargdef \csname\string#1\endcsname
  \tw@{#2}{#4}}
\long\def\@yargdef#1#2#3{%
  \@tempcnta#3\relax
  \advance \@tempcnta \@ne
  \let\@hash@\relax
  \edef\reserved@a{\ifx#2\tw@ [\@hash@1]\fi}%
  \@tempcntb #2%
  \@whilenum\@tempcntb <\@tempcnta
  \do{%
    \edef\reserved@a{\reserved@a\@hash@\the\@tempcntb}%
    \advance\@tempcntb \@ne}%
  \let\@hash@##%
  \l@ngrel@x\expandafter\def\expandafter#1\reserved@a}
\def\providecommand{\@star@or@long\provide@command}
\def\provide@command#1{%
  \begingroup
    \escapechar\m@ne\xdef\@gtempa{{\string#1}}%
  \endgroup
  \expandafter\@ifundefined\@gtempa
    {\def\reserved@a{\new@command#1}}%
    {\let\reserved@a\relax
     \def\reserved@a{\new@command\reserved@a}}%
   \reserved@a}%
\def\DeclareRobustCommand{\@star@or@long\declare@robustcommand}
\def\declare@robustcommand#1{%
   \edef\reserved@a{\string#1}%
   \def\reserved@b{#1}%
   \edef\reserved@b{\expandafter\strip@prefix\meaning\reserved@b}%
   \edef#1{%
      \ifx\reserved@a\reserved@b
         \noexpand\x@protect
         \noexpand#1%
      \fi
      \noexpand\protect
      \expandafter\noexpand\csname
         \expandafter\@gobble\string#1 \endcsname
   }%
   \expandafter\new@command\csname
      \expandafter\@gobble\string#1 \endcsname
}
\def\x@protect#1{%
   \ifx\protect\@typeset@protect\else
      \@x@protect#1%
   \fi
}
\catcode`\&=\z@  % Trick to hide conditionals
  \def\@x@protect#1&fi#2#3{&fi\protect#1}
```

The following little macro `\in@` is taken from `latex.ltx`; it checks whether its first argument is part of its second argument. It uses the boolean `\in@`; allocating a new boolean inside conditionally executed code is not possible, hence the construct with the temporary definition of `\bbl@tempa`.

```
  \def\bbl@tempa{\csname newif\endcsname&ifin@}
\catcode`\&=4
\ifx\in@\@undefined
```

```
  \def\in@#1#2{%
    \def\in@@##1#1##2##3\in@@{%
      \ifx\in@##2\in@false\else\in@true\fi}%
    \in@@#2#1\in@\in@@}
\else
  \let\bbl@tempa\@empty
\fi
\bbl@tempa
```

Used in some ldf files. Assume true.

```
\def\@ifpackagewith#1#2#3#4{#3}
```

The LaTeX macro `\@ifl@aded` checks whether a file was loaded. No-op in Plain.

```
\def\@ifl@aded#1#2#3#4{}
```

For the following code we need to make sure that the commands `\newcommand` and `\providecommand` exist with some sensible definition. They are not fully equivalent to their LaTeX 2_ε versions; just enough to make things work in plain TeXenvironments.

```
\ifx\@tempcnta\@undefined
  \csname newcount\endcsname\@tempcnta\relax
\fi
\ifx\@tempcntb\@undefined
  \csname newcount\endcsname\@tempcntb\relax
\fi
```

To prevent wasting two counters in LaTeX (because counters with the same name are allocated later by it) we reset the counter that holds the next free counter (`\count10`).

```
\ifx\bye\@undefined
  \advance\count10 by -2\relax
\fi
\ifx\@ifnextchar\@undefined
  \def\@ifnextchar#1#2#3{%
    \let\reserved@d=#1%
    \def\reserved@a{#2}\def\reserved@b{#3}%
    \futurelet\@let@token\@ifnch}
  \def\@ifnch{%
    \ifx\@let@token\@sptoken
      \let\reserved@c\@xifnch
    \else
      \ifx\@let@token\reserved@d
        \let\reserved@c\reserved@a
      \else
        \let\reserved@c\reserved@b
      \fi
    \fi
    \reserved@c}
  \def\:{\let\@sptoken= } \:  % this makes \@sptoken a space token
  \def\:{\@xifnch} \expandafter\def\: {\futurelet\@let@token\@ifnch}
\fi
\def\@testopt#1#2{%
  \@ifnextchar[{#1}{#1[#2]}}
\def\@protected@testopt#1{%
  \ifx\protect\@typeset@protect
    \expandafter\@testopt
  \else
    \@x@protect#1%
  \fi}
\long\def\@whilenum#1\do #2{\ifnum #1\relax #2\relax\@iwhilenum{#1\relax
    #2\relax}\fi}
\long\def\@iwhilenum#1{\ifnum #1\expandafter\@iwhilenum
        \else\expandafter\@gobble\fi{#1}}
```

2.2. Encoding related macros

Code from `ltoutenc.dtx`, adapted for use in the plain TEX environment.

```
\def\DeclareTextCommand{%
   \@dec@text@cmd\providecommand
}
\def\ProvideTextCommand{%
   \@dec@text@cmd\providecommand
}
\def\DeclareTextSymbol#1#2#3{%
   \@dec@text@cmd\chardef#1{#2}#3\relax
}
\def\@dec@text@cmd#1#2#3{%
   \expandafter\def\expandafter#2%
      \expandafter{%
         \csname#3-cmd\expandafter\endcsname
         \expandafter#2%
         \csname#3\string#2\endcsname
      }%
%   \let\@ifdefinable\@rc@ifdefinable
   \expandafter#1\csname#3\string#2\endcsname
}
\def\@current@cmd#1{%
  \ifx\protect\@typeset@protect\else
      \noexpand#1\expandafter\@gobble
  \fi
}
\def\@changed@cmd#1#2{%
   \ifx\protect\@typeset@protect
      \expandafter\ifx\csname\cf@encoding\string#1\endcsname\relax
         \expandafter\ifx\csname ?\string#1\endcsname\relax
            \expandafter\def\csname ?\string#1\endcsname{%
               \@changed@x@err{#1}%
            }%
         \fi
         \global\expandafter\let
           \csname\cf@encoding \string#1\expandafter\endcsname
           \csname ?\string#1\endcsname
      \fi
      \csname\cf@encoding\string#1%
        \expandafter\endcsname
   \else
      \noexpand#1%
   \fi
}
\def\@changed@x@err#1{%
    \errhelp{Your command will be ignored, type <return> to proceed}%
    \errmessage{Command \protect#1 undefined in encoding \cf@encoding}}
\def\DeclareTextCommandDefault#1{%
   \DeclareTextCommand#1?%
}
\def\ProvideTextCommandDefault#1{%
   \ProvideTextCommand#1?%
}
\expandafter\let\csname OT1-cmd\endcsname\@current@cmd
\expandafter\let\csname?-cmd\endcsname\@changed@cmd
\def\DeclareTextAccent#1#2#3{%
  \DeclareTextCommand#1{#2}[1]{\accent#3 ##1}
}
\def\DeclareTextCompositeCommand#1#2#3#4{%
   \expandafter\let\expandafter\reserved@a\csname#2\string#1\endcsname
   \edef\reserved@b{\string##1}%
   \edef\reserved@c{%
```

```
    \expandafter\@strip@args\meaning\reserved@a:-\@strip@args}%
  \ifx\reserved@b\reserved@c
     \expandafter\expandafter\expandafter\ifx
        \expandafter\@car\reserved@a\relax\relax\@nil
        \@text@composite
     \else
        \edef\reserved@b##1{%
           \def\expandafter\noexpand
              \csname#2\string#1\endcsname####1{%
              \noexpand\@text@composite
                 \expandafter\noexpand\csname#2\string#1\endcsname
                 ####1\noexpand\@empty\noexpand\@text@composite
                 {##1}%
           }%
        }%
        \expandafter\reserved@b\expandafter{\reserved@a{##1}}%
     \fi
     \expandafter\def\csname\expandafter\string\csname
        #2\endcsname\string#1-\string#3\endcsname{#4}
  \else
    \errhelp{Your command will be ignored, type <return> to proceed}%
    \errmessage{\string\DeclareTextCompositeCommand\space used on
        inappropriate command \protect#1}
  \fi
}
\def\@text@composite#1#2#3\@text@composite{%
  \expandafter\@text@composite@x
     \csname\string#1-\string#2\endcsname
}
\def\@text@composite@x#1#2{%
  \ifx#1\relax
     #2%
  \else
     #1%
  \fi
}
%
\def\@strip@args#1:#2-#3\@strip@args{#2}
\def\DeclareTextComposite#1#2#3#4{%
  \def\reserved@a{\DeclareTextCompositeCommand#1{#2}{#3}}%
  \bgroup
     \lccode`\@=#4%
     \lowercase{%
  \egroup
     \reserved@a @%
  }%
}
%
\def\UseTextSymbol#1#2{#2}
\def\UseTextAccent#1#2#3{}
\def\@use@text@encoding#1{}
\def\DeclareTextSymbolDefault#1#2{%
  \DeclareTextCommandDefault#1{\UseTextSymbol{#2}#1}%
}
\def\DeclareTextAccentDefault#1#2{%
  \DeclareTextCommandDefault#1{\UseTextAccent{#2}#1}%
}
\def\cf@encoding{OT1}
```

Currently we only use the LaTeX 2_ε method for accents for those that are known to be made active in *some* language definition file.

```
\DeclareTextAccent{\"}{OT1}{127}
\DeclareTextAccent{\'}{OT1}{19}
```

```
\DeclareTextAccent{\^}{OT1}{94}
\DeclareTextAccent{\`}{OT1}{18}
\DeclareTextAccent{\~}{OT1}{126}
```

The following control sequences are used in `babel.def` but are not defined for PLAIN TEX.

```
\DeclareTextSymbol{\textquotedblleft}{OT1}{92}
\DeclareTextSymbol{\textquotedblright}{OT1}{`\"}
\DeclareTextSymbol{\textquoteleft}{OT1}{`\`}
\DeclareTextSymbol{\textquoteright}{OT1}{`\'}
\DeclareTextSymbol{\i}{OT1}{16}
\DeclareTextSymbol{\ss}{OT1}{25}
```

For a couple of languages we need the LATEX-control sequence `\scriptsize` to be available. Because plain TEX doesn't have such a sophisticated font mechanism as LATEX has, we just `\let` it to `\sevenrm`.

```
\ifx\scriptsize\@undefined
  \let\scriptsize\sevenrm
\fi
```

And a few more "dummy" definitions.

```
\def\languagename{english}%
\def\bbl@ifshorthand#1#2#3{#2}%
\let\bbl@language@opts\@empty
\let\bbl@provide@locale\relax
\ifx\babeloptionstrings\@undefined
  \let\bbl@opt@strings\@nnil
\else
  \let\bbl@opt@strings\babeloptionstrings
\fi
\def\BabelStringsDefault{generic}
\def\bbl@tempa{normal}
\ifx\babeloptionmath\bbl@tempa
  \def\bbl@mathnormal{\noexpand\textormath}
\fi
\def\AfterBabelLanguage#1#2{}
\ifx\BabelModifiers\@undefined\let\BabelModifiers\relax\fi
\let\bbl@afterlang\relax
\ifx\@uclclist\@undefined\let\@uclclist\@empty\fi
\chardef\bbl@bidimode\z@
\fi
```

3. Basic macros

The emulation is finished and now comes the basic macros for Plain + `base`. See babel-code.pdf.

```
\bbl@trace{Basic macros}
\def\bbl@stripslash{\expandafter\@gobble\string}
\def\bbl@add#1#2{%
  \bbl@ifunset{\bbl@stripslash#1}%
    {\def#1{#2}}%
    {\expandafter\def\expandafter#1\expandafter{#1#2}}}
\def\bbl@xin@{\@expandtwoargs\in@}
\def\bbl@carg#1#2{\expandafter#1\csname#2\endcsname}%
\def\bbl@ncarg#1#2#3{\expandafter#1\expandafter#2\csname#3\endcsname}%
\def\bbl@ccarg#1#2#3{%
  \expandafter#1\csname#2\expandafter\endcsname\csname#3\endcsname}%
\def\bbl@csarg#1#2{\expandafter#1\csname bbl@#2\endcsname}%
\def\bbl@cs#1{\csname bbl@#1\endcsname}
\def\bbl@cl#1{\csname bbl@#1@\languagename\endcsname}
\def\bbl@loop#1#2#3{\bbl@@loop#1{#3}#2,\@nnil,}
\def\bbl@loopx#1#2{\expandafter\bbl@loop\expandafter#1\expandafter{#2}}
```

```
\def\bbl@@loop#1#2#3,{%
  \ifx\@nnil#3\relax\else
    \def#1{#3}#2\bbl@afterfi\bbl@@loop#1{#2}%
  \fi}
\def\bbl@for#1#2#3{\bbl@loopx#1{#2}{\ifx#1\@empty\else#3\fi}}
```

\bbl@add@list This internal macro adds its second argument to a comma separated list in its first argument.

```
\def\bbl@add@list#1#2{%
  \edef#1{%
    \bbl@ifunset{\bbl@stripslash#1}%
      {}%
      {\ifx#1\@empty\else#1,\fi}%
    #2}}
```

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

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

\bbl@exp Now, just syntactical sugar.

```
\def\bbl@exp#1{%
  \begingroup
    \let\\\noexpand
    \let\<\bbl@exp@en
    \let\[\bbl@exp@ue
    \edef\bbl@exp@aux{\endgroup#1}%
  \bbl@exp@aux}
\def\bbl@exp@en#1>{\expandafter\noexpand\csname#1\endcsname}%
\def\bbl@exp@ue#1]{%
  \unexpanded\expandafter\expandafter\expandafter{\csname#1\endcsname}}%
```

\bbl@trim Define two macros: `\bbl@trim` and `\bbl@trim@def`.

```
\def\bbl@tempa#1{%
  \long\def\bbl@trim##1##2{%
    \futurelet\bbl@trim@a\bbl@trim@c##2\@nil\@nil#1\@nil\relax{##1}}%
  \def\bbl@trim@c{%
    \ifx\bbl@trim@a\@sptoken
      \expandafter\bbl@trim@b
    \else
      \expandafter\bbl@trim@b\expandafter#1%
    \fi}%
  \long\def\bbl@trim@b#1##1 \@nil{\bbl@trim@i##1}}
\bbl@tempa{ }
\long\def\bbl@trim@i#1\@nil#2\relax#3{#3{#1}}
\long\def\bbl@trim@def#1{\bbl@trim{\def#1}}
```

\bbl@ifunset To check if a macro is defined, we create a new macro, which does the same as `\@ifundefined`. However, in an ε-tex engine, it is based on `\ifcsname`, which is more efficient, and does not waste memory. Defined inside a group, to avoid `\ifcsname` being implicitly set to `\relax` by the `\csname` test.

```
\begingroup
  \gdef\bbl@ifunset#1{%
    \expandafter\ifx\csname#1\endcsname\relax
      \expandafter\@firstoftwo
    \else
```

```
      \expandafter\@secondoftwo
    \fi}
  \bbl@ifunset{ifcsname}%
    {}%
    {\gdef\bbl@ifunset#1{%
       \ifcsname#1\endcsname
         \expandafter\ifx\csname#1\endcsname\relax
           \bbl@afterelse\expandafter\@firstoftwo
         \else
           \bbl@afterfi\expandafter\@secondoftwo
         \fi
       \else
         \expandafter\@firstoftwo
       \fi}}
\endgroup
```

\bbl@ifblank A tool from url, by Donald Arseneau, which tests if a string is empty or space.

```
\def\bbl@ifblank#1{%
  \bbl@ifblank@i#1\@nil\@nil\@secondoftwo\@firstoftwo\@nil}
\long\def\bbl@ifblank@i#1#2\@nil#3#4#5\@nil{#4}
\def\bbl@ifset#1#2#3{%
  \bbl@ifunset{#1}{#3}{\bbl@exp{\\\bbl@ifblank{\@nameuse{#1}}}{#3}{#2}}}
```

For each element in the comma separated <key>=<value> list, execute <code> with #1 and #2 as the key and the value of current item (trimmed).

```
\def\bbl@forkv#1#2{%
  \def\bbl@kvcmd##1##2##3{#2}%
  \bbl@kvnext#1,\@nil,}
\def\bbl@kvnext#1,{%
  \ifx\@nil#1\relax\else
    \bbl@ifblank{#1}{}{\bbl@forkv@eq#1=\@empty=\@nil{#1}}%
    \expandafter\bbl@kvnext
  \fi}
\def\bbl@forkv@eq#1=#2=#3\@nil#4{%
  \bbl@trim@def\bbl@forkv@a{#1}%
  \bbl@trim{\expandafter\bbl@kvcmd\expandafter{\bbl@forkv@a}}{#2}{#4}}
```

A *for* loop. Each item (trimmed) is #1. It cannot be nested (it's doable, but we don't need it).

```
\def\bbl@vforeach#1#2{%
  \def\bbl@forcmd##1{#2}%
  \bbl@fornext#1,\@nil,}
\def\bbl@fornext#1,{%
  \ifx\@nil#1\relax\else
    \bbl@ifblank{#1}{}{\bbl@trim\bbl@forcmd{#1}}%
    \expandafter\bbl@fornext
  \fi}
\def\bbl@foreach#1{\expandafter\bbl@vforeach\expandafter{#1}}
```

Some code should be executed once. The first argument is a flag.

```
\global\let\bbl@done\@empty
\def\bbl@once#1#2{%
  \bbl@xin@{,#1,}{,\bbl@done,}%
  \ifin@\else
    #2%
    \xdef\bbl@done{\bbl@done,#1,}%
  \fi}
%    \end{macrode}
%
% \macro{\bbl@replace}
%
% Returns implicitly |\toks@| with the modified string.
```

```
%
%    \begin{macrocode}
\def\bbl@replace#1#2#3{% in #1 -> repl #2 by #3
  \toks@{}%
  \def\bbl@replace@aux##1#2##2#2{%
    \ifx\bbl@nil##2%
      \toks@\expandafter{\the\toks@##1}%
    \else
      \toks@\expandafter{\the\toks@##1#3}%
      \bbl@afterfi
      \bbl@replace@aux##2#2%
    \fi}%
  \expandafter\bbl@replace@aux#1#2\bbl@nil#2%
  \edef#1{\the\toks@}}
```

An extension to the previous macro:

```
\ifx\detokenize\@undefined\else % Unused macros if old Plain TeX
  \bbl@exp{\def\\\bbl@parsedef##1\detokenize{macro:}}#2->#3\relax{%
    \def\bbl@tempa{#1}%
    \def\bbl@tempb{#2}%
    \def\bbl@tempe{#3}}
  \def\bbl@sreplace#1#2#3{%
    \begingroup
      \expandafter\bbl@parsedef\meaning#1\relax
      \def\bbl@tempc{#2}%
      \edef\bbl@tempc{\expandafter\strip@prefix\meaning\bbl@tempc}%
      \def\bbl@tempd{#3}%
      \edef\bbl@tempd{\expandafter\strip@prefix\meaning\bbl@tempd}%
      \bbl@xin@{\bbl@tempc}{\bbl@tempe}% If not in macro, do nothing
      \ifin@
        \bbl@exp{\\\bbl@replace\\\bbl@tempe{\bbl@tempc}{\bbl@tempd}}%
        \def\bbl@tempc{%     Expanded an executed below as 'uplevel'
           \\\makeatletter % "internal" macros with @ are assumed
           \\\scantokens{%
             \bbl@tempa\\\@namedef{\bbl@stripslash#1}\bbl@tempb{\bbl@tempe}%
             \noexpand\noexpand}%
           \catcode64=\the\catcode64\relax}%  Restore @
      \else
        \let\bbl@tempc\@empty  % Not \relax
      \fi
      \bbl@exp{%      For the 'uplevel' assignments
    \endgroup
      \bbl@tempc}}  % empty or expand to set #1 with changes
\fi
```

Two further tools.

```
\def\bbl@ifsamestring#1#2{%
  \begingroup
    \protected@edef\bbl@tempb{#1}%
    \edef\bbl@tempb{\expandafter\strip@prefix\meaning\bbl@tempb}%
    \protected@edef\bbl@tempc{#2}%
    \edef\bbl@tempc{\expandafter\strip@prefix\meaning\bbl@tempc}%
    \ifx\bbl@tempb\bbl@tempc
      \aftergroup\@firstoftwo
    \else
      \aftergroup\@secondoftwo
    \fi
  \endgroup}
\chardef\bbl@engine=%
  \ifx\directlua\@undefined
    \ifx\XeTeXinputencoding\@undefined
      \z@
    \else
```

```
      \tw@
    \fi
  \else
    \@ne
  \fi
```

A somewhat hackish tool (hence its name) to avoid spurious spaces in some contexts.

```
\def\bbl@bsphack{%
  \ifhmode
    \hskip\z@skip
    \def\bbl@esphack{\loop\ifdim\lastskip>\z@\unskip\repeat\unskip}%
  \else
    \let\bbl@esphack\@empty
  \fi}
```

Ordered key/value lists (aka associative arrays).

```
\def\bbl@array@end{\@empty}
\def\bbl@array@set#1#2#3{%
  \def\bbl@array@aux##1\bbl@elt#2\@@##2##3\bbl@array@aux##4{%
    \ifx\bbl@noelt##2%
      \def#1{##1\bbl@elt#2\@@{##4}}%
    \else
      \def\bbl@array@aux####1\bbl@elt#2\@@####2####3\bbl@array@aux####4{%
        \def#1{####1\bbl@elt#2\@@{#####4}####3}}%
      \expandafter\bbl@array@aux#1\bbl@array@aux{#3}%
    \fi}%
  \expandafter\bbl@array@aux#1\bbl@elt#2\@@\bbl@noelt\bbl@array@aux{#3}}
\def\bbl@array@get#1#2#3{%
  \def\bbl@array@aux##1\bbl@elt#2\@@##2##3\bbl@array@aux{%
    \ifx\bbl@array@end##2\@empty\let#3\relax\else\def#3{##2}\fi}%
  \expandafter\bbl@array@aux#1\bbl@elt#2\@@\bbl@array@end\bbl@array@aux}
```

That is all for the moment. Now follows some common stuff, for both Plain and LaTeX. After it, we will resume the LaTeX-only stuff.

4. Selectors and declaring languages

Still Plain + `base`.

```
\def\bbl@version{<@version@>}
\def\bbl@date{<@date@>}
<@Define Plain core switching macros@>
```

\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.

```
\def\adddialect#1#2{%
  \global\chardef#1#2\relax
  \bbl@usehooks{adddialect}{{#1}{#2}}%
  \begingroup
    \count@#1\relax
    \def\bbl@elt##1##2##3##4{%
      \ifnum\count@=##2\relax
        \edef\bbl@tempa{\expandafter\@gobbletwo\string#1}%
        \bbl@info{Hyphen rules for '\expandafter\@gobble\bbl@tempa'
                  set to \expandafter\string\csname l@##1\endcsname\\%
                  (\string\language\the\count@). Reported}%
        \def\bbl@elt####1####2####3####4{}%
      \fi}%
    \bbl@cs{languages}%
  \endgroup}
```

`\bbl@iflanguage` executes code only if the language `l@` exists.

```
\def\bbl@fixname#1{%
  \begingroup
    \def\bbl@tempe{l@}%
    \edef\bbl@tempd{\noexpand\@ifundefined{\noexpand\bbl@tempe#1}}%
    \bbl@tempd
      {\lowercase\expandafter{\bbl@tempd}%
         {\uppercase\expandafter{\bbl@tempd}%
           \@empty
           {\edef\bbl@tempd{\def\noexpand#1{#1}}%
            \uppercase\expandafter{\bbl@tempd}}}%
         {\edef\bbl@tempd{\def\noexpand#1{#1}}%
          \lowercase\expandafter{\bbl@tempd}}}%
      \@empty
    \edef\bbl@tempd{\endgroup\def\noexpand#1{#1}}%
  \bbl@tempd
  \bbl@exp{\\\bbl@usehooks{languagename}{{\languagename}{#1}}}}
\def\bbl@iflanguage#1{%
  \@ifundefined{l@#1}{\@nolanerr{#1}\@gobble}\@firstofone}
```

\iflanguage Users might want to test (in a private package for instance) which language is currently active.

```
\def\iflanguage#1{%
  \bbl@iflanguage{#1}{%
    \ifnum\csname l@#1\endcsname=\language
      \expandafter\@firstoftwo
    \else
      \expandafter\@secondoftwo
    \fi}}
```

4.1. Selecting the language

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

```
\let\bbl@select@type\z@
\edef\selectlanguage{%
  \noexpand\protect
  \expandafter\noexpand\csname selectlanguage \endcsname}
\ifx\@undefined\protect\let\protect\relax\fi
\let\xstring\string
```

\bbl@pop@language *But* when the language change happens *inside* a group the end of the group doesn't write anything to the auxiliary files.

\bbl@language@stack The previous solution works for one level of nesting groups, but as soon as more levels are used it is no longer adequate.

```
\def\bbl@language@stack{}
```

\bbl@push@language
\bbl@pop@language The stack is simply a list of languagenames, separated with a '+' sign; the push function can be simple:

```
\def\bbl@push@language{%
  \ifx\languagename\@undefined\else
    \ifx\currentgrouplevel\@undefined
      \xdef\bbl@language@stack{\languagename+\bbl@language@stack}%
    \else
      \ifnum\currentgrouplevel=\z@
        \xdef\bbl@language@stack{\languagename+}%
```

```
      \else
        \xdef\bbl@language@stack{\languagename+\bbl@language@stack}%
      \fi
    \fi
  \fi}
```

Retrieving information from the stack is a little bit less simple, as we need to remove the element from the stack while storing it in the macro `\languagename`.

\bbl@pop@lang This macro stores its first element (which is delimited by the '+'-sign) in `\languagename` and stores the rest of the string in `\bbl@language@stack`.

```
\def\bbl@pop@lang#1+#2\@@{%
  \edef\languagename{#1}%
  \xdef\bbl@language@stack{#2}}
\let\bbl@ifrestoring\@secondoftwo
\def\bbl@pop@language{%
  \expandafter\bbl@pop@lang\bbl@language@stack\@@
  \let\bbl@ifrestoring\@firstoftwo
  \expandafter\bbl@set@language\expandafter{\languagename}%
  \let\bbl@ifrestoring\@secondoftwo}
```

An alternative way to identify languages (in the babel sense) with a numerical value is introduced in 3.30.

```
\chardef\localeid\z@
\gdef\bbl@id@last{0}    % No real need for a new counter
\def\bbl@id@assign{%
  \bbl@ifunset{bbl@id@@\languagename}%
    {\count@\bbl@id@last\relax
     \advance\count@\@ne
     \global\bbl@csarg\chardef{id@@\languagename}\count@
     \xdef\bbl@id@last{\the\count@}%
     \ifcase\bbl@engine\or
       \directlua{
         Babel.locale_props[\bbl@id@last] = {}
         Babel.locale_props[\bbl@id@last].name = '\languagename'
         Babel.locale_props[\bbl@id@last].vars = {}
        }%
      \fi}%
    {}%
    \chardef\localeid\bbl@cl{id@}}
```

The unprotected part of `\selectlanguage`. In case it is used as environment, declare `\endselectlaguage`, just for safety.

```
\let\bbl@select@opts\@empty
\expandafter\def\csname selectlanguage \endcsname{%
  \@ifnextchar[\bbl@select@s{\bbl@select@s[]}}
\def\bbl@select@s[#1]#2{%
  \bbl@forkv{#1}{\bbl@array@set\bbl@select@opts{##1}{##2}}%
  \ifnum\bbl@hymapsel=\@cclv\let\bbl@hymapsel\tw@\fi
  \bbl@push@language
  \aftergroup\bbl@pop@language
  \bbl@set@language{#2}}
\let\endselectlanguage\relax
```

\bbl@set@language The macro `\bbl@set@language` takes care of switching the language environment *and* of writing entries on the auxiliary files.

```
\def\BabelContentsFiles{toc,lof,lot}
\def\bbl@set@language#1{% from selectlanguage, pop@
  % The old buggy way. Preserved for compatibility, but simplified
  \edef\languagename{\expandafter\string#1\@empty}%
  \select@language{\languagename}%
```

```
  \bbl@array@get\bbl@select@opts{main}\bbl@tempa
  \ifx\bbl@tempa\relax\else
    \let\bbl@main@language\localename
    \let\mainlocalename\localename
  \fi
  % write to aux files
  \expandafter\ifx\csname date\languagename\endcsname\relax\else
    \if@filesw
      \bbl@array@get\bbl@select@opts{nofiles}\bbl@tempa
      \ifx\bbl@tempa\relax
        \ifx\babel@aux\@gobbletwo\else % Set if single in the first, redundant
          \bbl@savelastskip
          \protected@write\@auxout{}{\string\babel@aux{\bbl@auxname}{}}%
          \bbl@restorelastskip
        \fi
        \bbl@usehooks{write}{}%
      \fi
    \fi
  \fi
  \let\bbl@select@opts\@empty}
%
\let\bbl@restorelastskip\relax
\let\bbl@savelastskip\relax
%
\def\select@language#1{% from set@, babel@aux, babel@toc
  \ifx\bbl@selectorname\@empty
    \def\bbl@selectorname{select}%
  \fi
  % set hymap
  \ifnum\bbl@hymapsel=\@cclv\chardef\bbl@hymapsel4\relax\fi
  % set name (when coming from babel@aux)
  \edef\languagename{#1}%
  \bbl@fixname\languagename
  % define \localename when coming from set@, with a trick
  \ifx\scantokens\@undefined
    \def\localename{??}%
  \else
    \bbl@exp{\\\scantokens{\def\\\localename{\languagename}\\\noexpand}\relax}%
  \fi
  \bbl@provide@locale
  \bbl@iflanguage\languagename{%
    \let\bbl@select@type\z@
    \expandafter\bbl@switch\expandafter{\languagename}}}
 %% TODO %% Plain:
\def\babel@aux#1#2{%
  \select@language{#1}%
  \bbl@foreach\BabelContentsFiles{% \relax -> don't assume vertical mode
    \@writefile{##1}{\babel@toc{#1}{#2}\relax}}}%
\def\babel@toc#1#2{%
  \select@language{#1}}
```

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.

```
\newif\ifbbl@usedategroup
\let\bbl@savedextras\@empty
\def\bbl@switch#1{%  from select@, foreign@
  % restore
  \originalTeX
  \expandafter\def\expandafter\originalTeX\expandafter{%
    \csname noextras#1\endcsname
    \let\originalTeX\@empty
    \babel@beginsave}%
  \bbl@usehooks{afterreset}{}%
```

```
\languageshorthands{none}%
% set the locale id
\bbl@id@assign
% switch captions, date
\bbl@bsphack
  \ifcase\bbl@select@type
    \csname captions#1\endcsname\relax
    \csname date#1\endcsname\relax
  \else
    \bbl@array@get\bbl@select@opts{captions}\bbl@tempa
    \ifx\bbl@tempa\relax\else
      \csname captions#1\endcsname\relax
    \fi
    \bbl@array@get\bbl@select@opts{date}\bbl@tempa
    \ifx\bbl@tempa\relax\else % if \foreign... within \<language>date
      \csname date#1\endcsname\relax
    \fi
  \fi
\bbl@esphack
% switch extras
\csname bbl@preextras@#1\endcsname
\bbl@usehooks{beforeextras}{}%
\csname extras#1\endcsname\relax
\bbl@usehooks{afterextras}{}%
%  > babel-ensure
%  > babel-sh-<short>
%  > babel-bidi
%  > babel-fontspec
\let\bbl@savedextras\@empty
% hyphenation - case mapping
\ifcase\bbl@opt@hyphenmap\or
  \def\BabelLower##1##2{\lccode##1=##2\relax}%
  \ifnum\bbl@hymapsel>4\else
    \csname\languagename @bbl@hyphenmap\endcsname
  \fi
  \chardef\bbl@opt@hyphenmap\z@
\else
  \ifnum\bbl@hymapsel>\bbl@opt@hyphenmap\else
    \csname\languagename @bbl@hyphenmap\endcsname
  \fi
\fi
\let\bbl@hymapsel\@cclv
% hyphenation - select rules
\ifnum\csname l@\languagename\endcsname=\l@unhyphenated
  \edef\bbl@tempa{u}%
\else
  \edef\bbl@tempa{\bbl@cl{lnbrk}}%
\fi
% linebreaking - handle u, e, k (v in the future)
\bbl@xin@{/u}{/\bbl@tempa}%
\ifin@\else\bbl@xin@{/e}{/\bbl@tempa}\fi % elongated forms
\ifin@\else\bbl@xin@{/k}{/\bbl@tempa}\fi % only kashida
\ifin@\else\bbl@xin@{/p}{/\bbl@tempa}\fi % padding (e.g., Tibetan)
\ifin@\else\bbl@xin@{/v}{/\bbl@tempa}\fi % variable font
% hyphenation - save mins
\babel@savevariable\lefthyphenmin
\babel@savevariable\righthyphenmin
\ifnum\bbl@engine=\@ne
  \babel@savevariable\hyphenationmin
\fi
\ifin@
  % unhyphenated/kashida/elongated/padding = allow stretching
  \language\l@unhyphenated
```

```
    \babel@savevariable\emergencystretch
    \emergencystretch\maxdimen
    \babel@savevariable\hbadness
    \hbadness\@M
  \else
    % other = select patterns
    \bbl@patterns{#1}%
  \fi
  % hyphenation - set mins
  \expandafter\ifx\csname #1hyphenmins\endcsname\relax
    \set@hyphenmins\tw@\thr@@\relax
    \@nameuse{bbl@hyphenmins@}%
  \else
    \expandafter\expandafter\expandafter\set@hyphenmins
      \csname #1hyphenmins\endcsname\relax
  \fi
  \@nameuse{bbl@hyphenmins@}%
  \@nameuse{bbl@hyphenmins@\languagename}%
  \@nameuse{bbl@hyphenatmin@}%
  \@nameuse{bbl@hyphenatmin@\languagename}%
  \let\bbl@selectorname\@empty}
```

otherlanguage It can be used as an alternative to using the `\selectlanguage` declarative command.

```
\edef\otherlanguage{%
  \noexpand\protect
  \expandafter\noexpand\csname otherlanguage \endcsname}
\expandafter\def\csname otherlanguage \endcsname{%
  \@ifstar{\@nameuse{otherlanguage*}}\bbl@otherlanguage}
\def\bbl@otherlanguage#1{%
  \def\bbl@selectorname{other}%
  \ifnum\bbl@hymapsel=\@cclv\let\bbl@hymapsel\thr@@\fi
  \csname selectlanguage \endcsname{#1}%
  \ignorespaces}
```

The `\endotherlanguage` part of the environment tries to hide itself when it is called in horizontal mode.

```
\long\def\endotherlanguage{\@ignoretrue\ignorespaces}
```

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

```
\expandafter\def\csname otherlanguage*\endcsname{%
  \@ifnextchar[\bbl@otherlanguage@s{\bbl@otherlanguage@s[]}}
\def\bbl@otherlanguage@s[#1]#2{%
  \def\bbl@selectorname{other*}%
  \ifnum\bbl@hymapsel=\@cclv\chardef\bbl@hymapsel4\relax\fi
  \bbl@forkv{#1}{\bbl@array@set\bbl@select@opts{##1}{##2}}%
  \foreign@language{#2}}
\expandafter\let\csname endotherlanguage*\endcsname\relax
```

\foreignlanguage 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.

```
\providecommand\bbl@beforeforeign{}
\edef\foreignlanguage{%
  \noexpand\protect
  \expandafter\noexpand\csname foreignlanguage \endcsname}
\expandafter\def\csname foreignlanguage \endcsname{%
  \@ifstar\bbl@foreign@s\bbl@foreign@x}
\providecommand\bbl@foreign@x[3][]{%
  \begingroup
```

```
    \def\bbl@selectorname{foreign}%
    \bbl@forkv{#1}{\bbl@array@set\bbl@select@opts{##1}{##2}}%
    \let\BabelText\@firstofone
    \bbl@beforeforeign
    \foreign@language{#2}%
    \bbl@usehooks{foreign}{}%
    \BabelText{#3}% Now in horizontal mode!
  \endgroup}
\def\bbl@foreign@s#1#2{%
  \begingroup
    {\par}%
    \def\bbl@selectorname{foreign*}%
    \let\bbl@select@opts\@empty
    \let\BabelText\@firstofone
    \foreign@language{#1}%
    \bbl@usehooks{foreign*}{}%
    \bbl@dirparastext
    \BabelText{#2}% Still in vertical mode!
    {\par}%
  \endgroup}
\providecommand\BabelWrapText[1]{%
   \def\bbl@tempa{\def\BabelText####1}%
   \expandafter\bbl@tempa\expandafter{\BabelText{#1}}}
```

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

```
\def\foreign@language#1{%
  % set name
  \edef\languagename{#1}%
  \ifbbl@usedategroup
    \bbl@array@set\bbl@select@opts{date}{\@empty}%
    \bbl@usedategroupfalse
  \fi
  \bbl@fixname\languagename
  \let\localename\languagename
  \bbl@provide@locale
  \bbl@iflanguage\languagename{%
    \let\bbl@select@type\@ne
    \expandafter\bbl@switch\expandafter{\languagename}}}
```

The following macro executes conditionally some code based on the selector being used.

```
\def\IfBabelSelectorTF#1{%
  \bbl@xin@{,\bbl@selectorname,}{,\zap@space#1 \@empty,}%
  \ifin@
    \expandafter\@firstoftwo
  \else
    \expandafter\@secondoftwo
  \fi}
```

\bbl@patterns This macro selects the hyphenation patterns by changing the `\language` register.

```
\let\bbl@hyphlist\@empty
\let\bbl@hyphenation@\relax
\let\bbl@pttnlist\@empty
\let\bbl@patterns@\relax
\let\bbl@hymapsel=\@cclv
\def\bbl@patterns#1{%
  \language=\expandafter\ifx\csname l@#1:\f@encoding\endcsname\relax
      \csname l@#1\endcsname
      \edef\bbl@tempa{#1}%
    \else
```

```
      \csname l@#1:\f@encoding\endcsname
      \edef\bbl@tempa{#1:\f@encoding}%
    \fi
  \@expandtwoargs\bbl@usehooks{patterns}{{#1}{\bbl@tempa}}%
  %  > luatex
  \@ifundefined{bbl@hyphenation@}{}{% Can be \relax!
    \begingroup
      \bbl@xin@{,\number\language,}{,\bbl@hyphlist}%
      \ifin@\else
        \@expandtwoargs\bbl@usehooks{hyphenation}{{#1}{\bbl@tempa}}%
        \hyphenation{%
          \bbl@hyphenation@
          \@ifundefined{bbl@hyphenation@#1}%
            \@empty
            {\space\csname bbl@hyphenation@#1\endcsname}}%
        \xdef\bbl@hyphlist{\bbl@hyphlist\number\language,}%
      \fi
    \endgroup}}
```

hyphenrules It can be used to select *just* the hyphenation rules.

```
\def\hyphenrules#1{%
  \edef\bbl@tempf{#1}%
  \bbl@fixname\bbl@tempf
  \bbl@iflanguage\bbl@tempf{%
    \expandafter\bbl@patterns\expandafter{\bbl@tempf}%
    \ifx\languageshorthands\@undefined\else
      \languageshorthands{none}%
    \fi
    \expandafter\ifx\csname\bbl@tempf hyphenmins\endcsname\relax
      \set@hyphenmins\tw@\thr@@\relax
    \else
      \expandafter\expandafter\expandafter\set@hyphenmins
      \csname\bbl@tempf hyphenmins\endcsname\relax
    \fi}}
\let\endhyphenrules\@empty
```

\providehyphenmins The macro `\providehyphenmins` should be used in the language definition files to provide a *default* setting for the hyphenation parameters `\lefthyphenmin` and `\righthyphenmin`

```
\def\providehyphenmins#1#2{%
  \expandafter\ifx\csname #1hyphenmins\endcsname\relax
    \@namedef{#1hyphenmins}{#2}%
  \fi}
```

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

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

\ProvidesLanguage The identification code for each file is something that was introduced in LaTeX 2_ε.

```
\ifx\ProvidesFile\@undefined
  \def\ProvidesLanguage#1[#2 #3 #4]{%
    \wlog{Language: #1 #4 #3 <#2>}%
    }
\else
  \def\ProvidesLanguage#1{%
    \begingroup
```

```
      \catcode`\ 10 %
      \@makeother\/%
      \@ifnextchar[%]
        {\@provideslanguage{#1}}{\@provideslanguage{#1}[]}}
  \def\@provideslanguage#1[#2]{%
    \wlog{Language: #1 #2}%
    \expandafter\xdef\csname ver@#1.ldf\endcsname{#2}%
    \endgroup}
\fi
\ifx\originalTeX\@undefined\let\originalTeX\@empty\fi
\ifx\babel@beginsave\@undefined\let\babel@beginsave\relax\fi
```

A few macro names are reserved for future releases of babel, which will use the concept of 'locale':

```
\providecommand\setlocale{\bbl@error{not-yet-available}{}{}{}}
\let\uselocale\setlocale
\let\locale\setlocale
\let\selectlocale\setlocale
\let\textlocale\setlocale
\let\textlanguage\setlocale
\let\languagetext\setlocale
```

4.2. Errors

\@nolanerr

\@nopatterns The babel package will signal an error when a documents tries to select a language that hasn't been defined earlier.

\@noopterr When the package was loaded without options not everything will work as expected. An error message is issued in that case.

```
\edef\bbl@nulllanguage{\string\language=0}
\def\bbl@nocaption#1#2{\NoCaseChange{\protect\bbl@nocaption@i{#1}{#2}}}
\def\bbl@nocaption@i#1#2{% 1: text to be printed 2: caption macro \langXname
  \global\@namedef{#2}{\textbf{?#1?}}%
  \@nameuse{#2}%
  \edef\bbl@tempa{#1}%
  \bbl@sreplace\bbl@tempa{name}{}%
  \bbl@warning{%
    \@backslashchar#1 not set for '\languagename'. Please,\\%
    define it after the language has been loaded\\%
    (typically in the preamble) with:\\%
    \string\setlocalecaption{\languagename}{\bbl@tempa}{..}\\%
    Feel free to contribute on github.com/latex3/babel.\\%
    Reported}}
\def\bbl@tentative{\protect\bbl@tentative@i}
\def\bbl@tentative@i#1{%
  \bbl@warning{%
    Some functions for '#1' are tentative.\\%
    They might not work as expected and their behavior\\%
    could change in the future.\\%
    Reported}}
\def\@nolanerr#1{\bbl@error{undefined-language}{#1}{}{}}
\def\@nopatterns#1{%
  \bbl@warning
    {No hyphenation patterns were preloaded for\\%
     the language '#1' into the format.\\%
     Please, configure your TeX system to add them and\\%
     rebuild the format. Now I will use the patterns\\%
     preloaded for \bbl@nulllanguage\space instead}}
\let\bbl@usehooks\@gobbletwo
```

5. Finish the very basic macros

Here ended the base option (LaTeX).

6. Only Plain

```
\ifx\bbl@onlyswitch\@empty\endinput\fi
```

6.1. More on selection

\babelensure The user command just parses the optional argument and creates a new macro named \bbl@e@⟨*language*⟩.

```
\bbl@trace{Defining babelensure}
\newcommand\babelensure[2][]{%
  \AddBabelHook{babel-ensure}{afterextras}{%
    \ifcase\bbl@select@type
      \bbl@cl{e}%
    \fi}%
  \begingroup
    \let\bbl@ens@include\@empty
    \let\bbl@ens@exclude\@empty
    \def\bbl@ens@fontenc{\relax}%
    \def\bbl@tempb##1{%
      \ifx\@empty##1\else\noexpand##1\expandafter\bbl@tempb\fi}%
    \edef\bbl@tempa{\bbl@tempb#1\@empty}%
    \def\bbl@tempb##1=##2\@@{\@namedef{bbl@ens@##1}{##2}}%
    \bbl@foreach\bbl@tempa{\bbl@tempb##1\@@}%
    \def\bbl@tempc{\bbl@ensure}%
    \expandafter\bbl@add\expandafter\bbl@tempc\expandafter{%
      \expandafter{\bbl@ens@include}}%
    \expandafter\bbl@add\expandafter\bbl@tempc\expandafter{%
      \expandafter{\bbl@ens@exclude}}%
    \toks@\expandafter{\bbl@tempc}%
    \bbl@exp{%
  \endgroup
  \def\<bbl@e@#2>{\the\toks@{\bbl@ens@fontenc}}}}
\def\bbl@ensure#1#2#3{% 1: include 2: exclude 3: fontenc
  \def\bbl@tempb##1{% elt for (excluding) \bbl@captionslist list
    \ifx##1\@undefined % 3.32 - Don't assume the macro exists
      \edef##1{\noexpand\bbl@nocaption
        {\bbl@stripslash##1}{\languagename\bbl@stripslash##1}}%
    \fi
    \ifx##1\@empty\else
      \in@{##1}{#2}%
      \ifin@\else
        \let\bbl@tempc\languagename
        \bbl@replace\bbl@tempc{-}{@@}%
        \bbl@ifunset{bbl@ensure@\bbl@tempc}%
          {\bbl@exp{%
            \\\DeclareRobustCommand\<bbl@ensure@\bbl@tempc>[1]{%
              \\\foreignlanguage{\languagename}%
              {\ifx\relax#3\else
                \\\fontencoding{#3}\\\selectfont
               \fi
               ########1}}}}%
          {}%
        \toks@\expandafter{##1}%
        \edef##1{%
           \bbl@csarg\noexpand{ensure@\bbl@tempc}%
           {\the\toks@}}%
      \fi
      \expandafter\bbl@tempb
```

```
    \fi}%
  \expandafter\bbl@tempb\bbl@captionslist\today\@empty
  \def\bbl@tempa##1{% elt for include list
    \ifx##1\@empty\else
      \let\bbl@tempc\languagename
      \bbl@replace\bbl@tempc{-}{@@}%
      \bbl@csarg\in@{ensure@\bbl@tempc\expandafter}\expandafter{##1}%
      \ifin@\else
        \bbl@tempb##1\@empty
      \fi
      \expandafter\bbl@tempa
    \fi}%
  \bbl@tempa#1\@empty}
\def\bbl@captionslist{%
  \prefacename\refname\abstractname\bibname\chaptername\appendixname
  \contentsname\listfigurename\listtablename\indexname\figurename
  \tablename\partname\enclname\ccname\headtoname\pagename\seename
  \alsoname\proofname\glossaryname}
```

6.2. Short tags

\babeltags This macro is straightforward.

```
\bbl@trace{Short tags}
\newcommand\babeltags[1]{%
  \edef\bbl@tempa{\zap@space#1 \@empty}%
  \def\bbl@tempb##1=##2\@@{%
    \edef\bbl@tempc{%
      \noexpand\newcommand
      \expandafter\noexpand\csname ##1\endcsname{%
        \noexpand\protect
        \expandafter\noexpand\csname otherlanguage*\endcsname{##2}}
      \noexpand\newcommand
      \expandafter\noexpand\csname text##1\endcsname{%
        \noexpand\foreignlanguage{##2}}}
    \bbl@tempc}%
  \bbl@for\bbl@tempa\bbl@tempa{%
    \expandafter\bbl@tempb\bbl@tempa\@@}}
```

6.3. Compatibility with language.def

Plain e-TEX doesn't rely on language.dat, but babel can be made compatible with this format easily.

```
\bbl@trace{Compatibility with language.def}
\ifx\directlua\@undefined\else
  \ifx\bbl@luapatterns\@undefined
    \input luababel.def
  \fi
\fi
\ifx\bbl@languages\@undefined
  \ifx\directlua\@undefined
    \openin0 = language.def
    \ifeof0
      \closein0
      \message{I couldn't find the file language.def}
    \else
      \closein0
      \begingroup
        \def\addlanguage#1#2#3#4#5{%
          \expandafter\ifx\csname lang@#1\endcsname\relax\else
            \global\expandafter\let\csname l@#1\expandafter\endcsname
              \csname lang@#1\endcsname
```

```
          \fi}%
        \def\uselanguage#1{}%
        \input language.def
      \endgroup
    \fi
  \fi
  \chardef\l@english\z@
\fi
```

\addto It takes two arguments, a ⟨*control sequence*⟩ and TEX-code to be added to the ⟨*control sequence*⟩.

```
\def\addto#1#2{%
  \ifx#1\@undefined
    \def#1{#2}%
  \else
    \ifx#1\relax
      \def#1{#2}%
    \else
      {\toks@\expandafter{#1#2}%
       \xdef#1{\the\toks@}}%
    \fi
  \fi}
```

6.4. Hooks

Admittedly, the current implementation is a somewhat simplistic.

```
\bbl@trace{Hooks}
\newcommand\AddBabelHook[3][]{%
  \bbl@ifunset{bbl@hk@#2}{\EnableBabelHook{#2}}{}%
  \def\bbl@tempa##1,#3=##2,##3\@empty{\def\bbl@tempb{##2}}%
  \expandafter\bbl@tempa\bbl@evargs,#3=,\@empty
  \bbl@ifunset{bbl@ev@#2@#3@#1}%
    {\bbl@csarg\bbl@add{ev@#3@#1}{\bbl@elth{#2}}}%
    {\bbl@csarg\let{ev@#2@#3@#1}\relax}%
  \bbl@csarg\newcommand{ev@#2@#3@#1}[\bbl@tempb]}
\newcommand\EnableBabelHook[1]{\bbl@csarg\let{hk@#1}\@firstofone}
\newcommand\DisableBabelHook[1]{\bbl@csarg\let{hk@#1}\@gobble}
\def\bbl@usehooks{\bbl@usehooks@lang\languagename}
\def\bbl@usehooks@lang#1#2#3{% Test for Plain
  \ifx\UseHook\@undefined\else\UseHook{babel/*/#2}\fi
  \def\bbl@elth##1{%
    \bbl@cs{hk@##1}{\bbl@cs{ev@##1@#2@}#3}}%
  \bbl@cs{ev@#2@}%
  \ifx\languagename\@undefined\else % Test required for Plain (?)
    \ifx\UseHook\@undefined\else\UseHook{babel/#1/#2}\fi
    \def\bbl@elth##1{%
      \bbl@cs{hk@##1}{\bbl@cs{ev@##1@#2@#1}#3}}%
    \bbl@cs{ev@#2@#1}%
  \fi}
\def\bbl@evargs{,% <- don't delete this comma
  everylanguage=1,loadkernel=1,loadpatterns=1,loadexceptions=1,%
  adddialect=2,patterns=2,defaultcommands=0,encodedcommands=2,write=0,%
  beforeextras=0,afterextras=0,stopcommands=0,stringprocess=0,%
  hyphenation=2,initiateactive=3,afterreset=0,foreign=0,foreign*=0,%
  beforestart=0,languagename=2,begindocument=1}
\ifx\NewHook\@undefined\else % Test for Plain (?)
  \def\bbl@tempa#1=#2\@@{\NewHook{babel/#1}\NewHook{babel/*/#1}}
  \bbl@foreach\bbl@evargs{\bbl@tempa#1\@@}
\fi
```

6.5. Setting up language files

\LdfInit `\LdfInit` macro takes two arguments. The first argument is the name of the language that will be defined in the language definition file; the second argument is either a control sequence or a string from which a control sequence should be constructed.

```
\bbl@trace{Macros for setting language files up}
\def\bbl@ldfinit{%
  \let\bbl@screset\@empty
  \let\BabelStrings\bbl@opt@string
  \let\BabelOptions\@empty
  \let\BabelLanguages\relax
  \ifx\originalTeX\@undefined
    \let\originalTeX\@empty
  \else
    \originalTeX
  \fi}
\def\LdfInit#1#2{%
  \chardef\atcatcode=\catcode`\@
  \catcode`\@=11\relax
  \chardef\eqcatcode=\catcode`\=
  \catcode`\==12\relax
  \@ifpackagewith{babel}{ensureinfo=off}{}%
    {\ifx\InputIfFileExists\@undefined\else
       \bbl@ifunset{bbl@lname@#1}%
         {{\let\bbl@ensuring\@empty % Flag used in babel-serbianc.tex
           \def\languagename{#1}%
           \bbl@id@assign
           \bbl@load@info{#1}}}%
         {}%
     \fi}%
  \expandafter\if\expandafter\@backslashchar
                  \expandafter\@car\string#2\@nil
    \ifx#2\@undefined\else
      \ldf@quit{#1}%
    \fi
  \else
    \expandafter\ifx\csname#2\endcsname\relax\else
      \ldf@quit{#1}%
    \fi
  \fi
  \bbl@ldfinit}
```

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

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

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

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

```
  \catcode`\==\eqcatcode \let\eqcatcode\relax}
```

After the preamble of the document the commands `\LdfInit`, `\ldf@quit` and `\ldf@finish` are no longer needed.

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

`\main@language`
`\bbl@main@language` This command should be used in the various language definition files.

```
\def\main@language#1{%
  \def\bbl@main@language{#1}%
  \let\languagename\bbl@main@language
  \let\localename\bbl@main@language
  \let\mainlocalename\bbl@main@language
  \bbl@id@assign
  \ifcase\bbl@engine\or
    \ifx\setattribute\@undefined\else
      \setattribute\bbl@attr@locale\localeid
    \fi
  \fi
  \bbl@patterns{\languagename}}
```

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

```
\AtBeginDocument{\expandafter\selectlanguage\expandafter{\bbl@main@language}}
```

6.6. Shorthands

The macro `\initiate@active@char` below takes all the necessary actions to make its argument a shorthand character.

```
\bbl@trace{Shorhands}
\def\bbl@withactive#1#2{%
  \begingroup
    \lccode`~=`#2\relax
    \lowercase{\endgroup#1~}}
```

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

```
\def\bbl@add@special#1{% 1:a macro like \", \?, etc.
  \bbl@add\dospecials{\do#1}% test @sanitize = \relax, for back. compat.
  \bbl@ifunset{@sanitize}{}{\bbl@add\@sanitize{\@makeother#1}}%
  \ifx\nfss@catcodes\@undefined\else
    \begingroup
      \catcode`#1\active
      \nfss@catcodes
      \ifnum\catcode`#1=\active
        \endgroup
        \bbl@add\nfss@catcodes{\@makeother#1}%
      \else
        \endgroup
      \fi
  \fi}
```

\initiate@active@char A language definition file can call this macro to make a character active.

```
\def\bbl@active@def#1#2#3#4{%
  \@namedef{#3#1}{%
    \expandafter\ifx\csname#2@sh@#1@\endcsname\relax
      \bbl@afterelse\bbl@sh@select#2#1{#3@arg#1}{#4#1}%
    \else
      \bbl@afterfi\csname#2@sh@#1@\endcsname
    \fi}%
  \long\@namedef{#3@arg#1}##1{%
    \expandafter\ifx\csname#2@sh@#1@\string##1@\endcsname\relax
      \bbl@afterelse\csname#4#1\endcsname##1%
    \else
      \bbl@afterfi\csname#2@sh@#1@\string##1@\endcsname
    \fi}}%
\def\initiate@active@char#1{%
  \bbl@ifunset{active@char\string#1}%
    {\bbl@withactive
      {\expandafter\@initiate@active@char\expandafter}#1\string#1#1}%
    {}}
\def\@initiate@active@char#1#2#3{%
  \bbl@csarg\edef{oricat@#2}{\catcode`#2=\the\catcode`#2\relax}%
  \ifx#1\@undefined
    \bbl@csarg\def{oridef@#2}{\def#1{\active@prefix#1\@undefined}}%
  \else
    \bbl@csarg\let{oridef@@#2}#1%
    \bbl@csarg\edef{oridef@#2}{%
      \let\noexpand#1%
      \expandafter\noexpand\csname bbl@oridef@@#2\endcsname}%
  \fi
  \ifx#1#3\relax
    \expandafter\let\csname normal@char#2\endcsname#3%
  \else
    \bbl@info{Making #2 an active character}%
    \ifnum\mathcode`#2=\ifodd\bbl@engine"1000000 \else"8000 \fi
      \@namedef{normal@char#2}{%
        \textormath{#3}{\csname bbl@oridef@@#2\endcsname}}%
    \else
      \@namedef{normal@char#2}{#3}%
    \fi
    \bbl@restoreactive{#2}%
    \AtBeginDocument{%
      \catcode`#2\active
      \if@filesw
        \immediate\write\@mainaux{\catcode`\string#2\active}%
      \fi}%
    \expandafter\bbl@add@special\csname#2\endcsname
    \catcode`#2\active
  \fi
  \let\bbl@tempa\@firstoftwo
  \if\string^#2%
    \def\bbl@tempa{\noexpand\textormath}%
  \else
    \ifx\bbl@mathnormal\@undefined\else
      \let\bbl@tempa\bbl@mathnormal
    \fi
  \fi
  \expandafter\edef\csname active@char#2\endcsname{%
    \bbl@tempa
      {\noexpand\if@safe@actives
         \noexpand\expandafter
         \expandafter\noexpand\csname normal@char#2\endcsname
       \noexpand\else
```

```
        \noexpand\expandafter
        \expandafter\noexpand\csname bbl@doactive#2\endcsname
      \noexpand\fi}%
    {\expandafter\noexpand\csname normal@char#2\endcsname}}%
\bbl@csarg\edef{doactive#2}{%
  \expandafter\noexpand\csname user@active#2\endcsname}%
\bbl@csarg\edef{active@#2}{%
  \noexpand\active@prefix\noexpand#1%
  \expandafter\noexpand\csname active@char#2\endcsname}%
\bbl@csarg\edef{normal@#2}{%
  \noexpand\active@prefix\noexpand#1%
  \expandafter\noexpand\csname normal@char#2\endcsname}%
\bbl@ncarg\let#1{bbl@normal@#2}%
\bbl@active@def#2\user@group{user@active}{language@active}%
\bbl@active@def#2\language@group{language@active}{system@active}%
\bbl@active@def#2\system@group{system@active}{normal@char}%
\expandafter\edef\csname\user@group @sh@#2@@\endcsname
  {\expandafter\noexpand\csname normal@char#2\endcsname}%
\expandafter\edef\csname\user@group @sh@#2@\string\protect@\endcsname
  {\expandafter\noexpand\csname user@active#2\endcsname}%
\if\string'#2%
  \let\prim@s\bbl@prim@s
  \let\active@math@prime#1%
\fi
\bbl@usehooks{initiateactive}{{#1}{#2}{#3}}}
\@ifpackagewith{babel}{KeepShorthandsActive}%
 {\let\bbl@restoreactive\@gobble}%
 {\def\bbl@restoreactive#1{%
    \bbl@exp{%
      \\\AfterBabelLanguage\\\CurrentOption
        {\catcode`#1=\the\catcode`#1\relax}%
      \\\AtEndOfPackage
        {\catcode`#1=\the\catcode`#1\relax}}}%
  \AtEndOfPackage{\let\bbl@restoreactive\@gobble}}
```

\bbl@sh@select This command helps the shorthand supporting macros to select how to proceed.

```
\def\bbl@sh@select#1#2{%
  \expandafter\ifx\csname#1@sh@#2@sel\endcsname\relax
    \bbl@afterelse\bbl@scndcs
  \else
    \bbl@afterfi\csname#1@sh@#2@sel\endcsname
  \fi}
```

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

```
\begingroup
\bbl@ifunset{ifincsname}
 {\gdef\active@prefix#1{%
    \ifx\protect\@typeset@protect
    \else
      \ifx\protect\@unexpandable@protect
        \noexpand#1%
      \else
        \protect#1%
      \fi
      \expandafter\@gobble
    \fi}}
 {\gdef\active@prefix#1{%
    \ifincsname
```

```
      \string#1%
      \expandafter\@gobble
    \else
      \ifx\protect\@typeset@protect
      \else
        \ifx\protect\@unexpandable@protect
          \noexpand#1%
        \else
          \protect#1%
        \fi
        \expandafter\expandafter\expandafter\@gobble
      \fi
    \fi}}
\endgroup
```

if@safe@actives In some circumstances it is necessary to be able to reset the shorthand to its 'normal' value (usually the character with catcode 'other') on the fly.

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

\bbl@restore@actives When the output routine kicks in while the active characters were made "safe" this must be undone in the headers to prevent unexpected typeset results.

```
\def\bbl@restore@actives{\if@safe@actives\@safe@activesfalse\fi}
```

\bbl@activate
\bbl@deactivate Used to change the definition of an active character to expand to `\active@char`⟨*char*⟩ in the case of `\bbl@activate`, or `\normal@char`⟨*char*⟩ in the case of `\bbl@deactivate`.

```
\chardef\bbl@activated\z@
\def\bbl@activate#1{%
  \chardef\bbl@activated\@ne
  \bbl@withactive{\expandafter\let\expandafter}#1%
    \csname bbl@active@\string#1\endcsname}
\def\bbl@deactivate#1{%
  \chardef\bbl@activated\tw@
  \bbl@withactive{\expandafter\let\expandafter}#1%
    \csname bbl@normal@\string#1\endcsname}
```

\bbl@firstcs
\bbl@scndcs These macros are used only as a trick when declaring shorthands.

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

\declare@shorthand Used to declare a shorthand on a certain level.

```
\def\babel@texpdf#1#2#3#4{%
  \ifx\texorpdfstring\@undefined
    \textormath{#1}{#3}%
  \else
    \texorpdfstring{\textormath{#1}{#3}}{#2}%
    % \texorpdfstring{\textormath{#1}{#3}}{\textormath{#2}{#4}}%
  \fi}
%
\def\declare@shorthand#1#2{\@decl@short{#1}#2\@nil}
\def\@decl@short#1#2#3\@nil#4{%
  \def\bbl@tempa{#3}%
  \ifx\bbl@tempa\@empty
    \expandafter\let\csname #1@sh@\string#2@sel\endcsname\bbl@scndcs
    \bbl@ifunset{#1@sh@\string#2@}{}%
```

```
      {\def\bbl@tempa{#4}%
       \expandafter\ifx\csname#1@sh@\string#2@\endcsname\bbl@tempa
       \else
         \bbl@info
           {Redefining #1 shorthand \string#2\\%
            in language \CurrentOption}%
       \fi}%
    \@namedef{#1@sh@\string#2@}{#4}%
  \else
    \expandafter\let\csname #1@sh@\string#2@sel\endcsname\bbl@firstcs
    \bbl@ifunset{#1@sh@\string#2@\string#3@}{}%
      {\def\bbl@tempa{#4}%
       \expandafter\ifx\csname#1@sh@\string#2@\string#3@\endcsname\bbl@tempa
       \else
         \bbl@info
           {Redefining #1 shorthand \string#2\string#3\\%
            in language \CurrentOption}%
       \fi}%
    \@namedef{#1@sh@\string#2@\string#3@}{#4}%
  \fi}
```

\textormath Some of the shorthands that will be declared by the language definition files have to be usable in both text and mathmode.

```
\def\textormath{%
  \ifmmode
    \expandafter\@secondoftwo
  \else
    \expandafter\@firstoftwo
  \fi}
```

\user@group
\language@group
\system@group The current concept of 'shorthands' supports three levels or groups of shorthands.

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

\useshorthands This is the user level macro.

```
\def\useshorthands{%
  \@ifstar\bbl@usesh@s{\bbl@usesh@x{}}}
\def\bbl@usesh@s#1{%
  \bbl@usesh@x
    {\AddBabelHook{babel-sh-\string#1}{afterextras}{\bbl@activate{#1}}}%
    {#1}}
\def\bbl@usesh@x#1#2{%
  \bbl@ifshorthand{#2}%
    {\def\user@group{user}%
     \initiate@active@char{#2}%
     #1%
     \bbl@activate{#2}}%
    {\bbl@error{shorthand-is-off}{}{#2}{}}}
```

\defineshorthand Currently we only support two groups of user level shorthands, named internally `user` and `user@`⟨*language*⟩ (language-dependent user shorthands).

```
\def\user@language@group{user@\language@group}
\def\bbl@set@user@generic#1#2{%
  \bbl@ifunset{user@generic@active#1}%
    {\bbl@active@def#1\user@language@group{user@active}{user@generic@active}%
```

```
    \bbl@active@def#1\user@group{user@generic@active}{language@active}%
    \expandafter\edef\csname#2@sh@#1@@\endcsname{%
      \expandafter\noexpand\csname normal@char#1\endcsname}%
    \expandafter\edef\csname#2@sh@#1@\string\protect@\endcsname{%
      \expandafter\noexpand\csname user@active#1\endcsname}}%
  \@empty}
\newcommand\defineshorthand[3][user]{%
  \edef\bbl@tempa{\zap@space#1 \@empty}%
  \bbl@for\bbl@tempb\bbl@tempa{%
    \if*\expandafter\@car\bbl@tempb\@nil
      \edef\bbl@tempb{user@\expandafter\@gobble\bbl@tempb}%
      \@expandtwoargs
        \bbl@set@user@generic{\expandafter\string\@car#2\@nil}\bbl@tempb
    \fi
    \declare@shorthand{\bbl@tempb}{#2}{#3}}}
```

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

```
\def\languageshorthands#1{%
  \bbl@ifsamestring{none}{#1}{}{%
    \bbl@once{short-\localename-#1}{%
      \bbl@info{'\localename' activates '#1' shorthands.\\Reported}}}%
  \def\language@group{#1}}
```

\aliasshorthand *Deprecated.*

```
\def\aliasshorthand#1#2{%
  \bbl@ifshorthand{#2}%
    {\expandafter\ifx\csname active@char\string#2\endcsname\relax
       \ifx\document\@notprerr
         \@notshorthand{#2}%
       \else
         \initiate@active@char{#2}%
         \bbl@ccarg\let{active@char\string#2}{active@char\string#1}%
         \bbl@ccarg\let{normal@char\string#2}{normal@char\string#1}%
         \bbl@activate{#2}%
       \fi
     \fi}%
    {\bbl@error{shorthand-is-off}{}{#2}{}}}
```

\@notshorthand

```
\def\@notshorthand#1{\bbl@error{not-a-shorthand}{#1}{}{}}
```

\shorthandon
\shorthandoff The first level definition of these macros just passes the argument on to `\bbl@switch@sh`, adding `\@nil` at the end to denote the end of the list of characters.

```
\newcommand*\shorthandon[1]{\bbl@switch@sh\@ne#1\@nnil}
\DeclareRobustCommand*\shorthandoff{%
  \@ifstar{\bbl@shorthandoff\tw@}{\bbl@shorthandoff\z@}}
\def\bbl@shorthandoff#1#2{\bbl@switch@sh#1#2\@nnil}
```

\bbl@switch@sh The macro `\bbl@switch@sh` takes the list of characters apart one by one and subsequently switches the category code of the shorthand character according to the first argument of `\bbl@switch@sh`.

```
\def\bbl@switch@sh#1#2{%
  \ifx#2\@nnil\else
    \bbl@ifunset{bbl@active@\string#2}%
      {\bbl@error{not-a-shorthand-b}{}{#2}{}}%
      {\ifcase#1%   off, on, off*
```

```
        \catcode`#212\relax
      \or
        \catcode`#2\active
        \bbl@ifunset{bbl@shdef@\string#2}%
          {}%
          {\bbl@withactive{\expandafter\let\expandafter}#2%
             \csname bbl@shdef@\string#2\endcsname
           \bbl@csarg\let{shdef@\string#2}\relax}%
        \ifcase\bbl@activated\or
          \bbl@activate{#2}%
        \else
          \bbl@deactivate{#2}%
        \fi
      \or
        \bbl@ifunset{bbl@shdef@\string#2}%
          {\bbl@withactive{\bbl@csarg\let{shdef@\string#2}}#2}%
          {}%
        \csname bbl@oricat@\string#2\endcsname
        \csname bbl@oridef@\string#2\endcsname
      \fi}%
    \bbl@afterfi\bbl@switch@sh#1%
  \fi}
```

Note the value is that at the expansion time; e.g., in the preamble shorthands are usually deactivated.

```
\def\babelshorthand{\active@prefix\babelshorthand\bbl@putsh}
\def\bbl@putsh#1{%
  \bbl@ifunset{bbl@active@\string#1}%
     {\bbl@putsh@i#1\@empty\@nnil}%
     {\csname bbl@active@\string#1\endcsname}}
\def\bbl@putsh@i#1#2\@nnil{%
  \csname\language@group @sh@\string#1@%
    \ifx\@empty#2\else\string#2@\fi\endcsname}
```

You may want to test if a character is a shorthand. Note it does not test whether the shorthand is on or off.

```
\newcommand\ifbabelshorthand[3]{\bbl@ifunset{bbl@active@\string#1}{#3}{#2}}
```

\bbl@prim@s
\bbl@pr@m@s One of the internal macros that are involved in substituting `\prime` for each right quote in mathmode is `\prim@s`.

```
\def\bbl@prim@s{%
  \prime\futurelet\@let@token\bbl@pr@m@s}
\def\bbl@if@primes#1#2{%
  \ifx#1\@let@token
    \expandafter\@firstoftwo
  \else\ifx#2\@let@token
    \bbl@afterelse\expandafter\@firstoftwo
  \else
    \bbl@afterfi\expandafter\@secondoftwo
  \fi\fi}
\begingroup
  \catcode`\^=7  \catcode`\*=\active  \lccode`\*=`\^
  \catcode`\'=12 \catcode`\"=\active  \lccode`\"=`\'
  \lowercase{%
    \gdef\bbl@pr@m@s{%
      \bbl@if@primes"'%
        \pr@@@s
        {\bbl@if@primes*^\pr@@@t\egroup}}}
\endgroup
```

Usually the ~ is active and expands to `\penalty\@M\␣`. When it is written to the aux file it is written expanded. To prevent that and to be able to use the character ~ as a start

character for a shorthand, it is redefined here as a one character shorthand on system level.

```
\initiate@active@char{~}
\declare@shorthand{system}{~}{\leavevmode\nobreak\ }
\bbl@activate{~}
```

\OT1dqpos
\T1dqpos The position of the double quote character is different for the OT1 and T1 encodings.

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

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

6.7. Language attributes

Language attributes provide a means to give the user control over which features of the language definition files he wants to enable.

\languageattribute The macro `\languageattribute` checks whether its arguments are valid and then activates the selected language attribute.

```
\bbl@trace{Language attributes}
\newcommand\languageattribute[2]{%
  \def\bbl@tempc{#1}%
  \bbl@fixname\bbl@tempc
  \bbl@iflanguage\bbl@tempc{%
    \bbl@vforeach{#2}{%
      \ifx\bbl@known@attribs\@undefined
        \in@false
      \else
        \bbl@xin@{,\bbl@tempc-##1,}{,\bbl@known@attribs,}%
      \fi
      \ifin@
        \bbl@warning{%
          You have more than once selected the attribute '##1'\\%
          for language #1. Reported}%
      \else
        \bbl@info{Activated '##1' attribute for\\%
          '\bbl@tempc'. Reported}%
        \bbl@exp{%
          \\\bbl@add@list\\\bbl@known@attribs{\bbl@tempc-##1}}%
        \edef\bbl@tempa{\bbl@tempc-##1}%
        \expandafter\bbl@ifknown@ttrib\expandafter{\bbl@tempa}\bbl@attributes
          {\csname\bbl@tempc @attr@##1\endcsname}%
          {\bbl@error{unknown-attribute}{\bbl@tempc}{##1}{}}%
     \fi}}}
\@onlypreamble\languageattribute
```

\bbl@declare@ttribute This command adds the new language/attribute combination to the list of known attributes.

```
\def\bbl@declare@ttribute#1#2#3{%
  \bbl@xin@{,#2,}{,\BabelModifiers,}%
  \ifin@
    \AfterBabelLanguage{#1}{\languageattribute{#1}{#2}}%
  \fi
```

```
  \bbl@add@list\bbl@attributes{#1-#2}%
  \expandafter\def\csname#1@attr@#2\endcsname{#3}}
```

\bbl@ifattributeset This internal macro has 4 arguments. It can be used to interpret TEX code based on whether a certain attribute was set. T

```
\def\bbl@ifattributeset#1#2#3#4{%
  \ifx\bbl@known@attribs\@undefined
    \in@false
  \else
    \bbl@xin@{,#1-#2,}{,\bbl@known@attribs,}%
  \fi
  \ifin@
    \bbl@afterelse#3%
  \else
    \bbl@afterfi#4%
  \fi}
```

\bbl@ifknown@ttrib An internal macro to check whether a given language/attribute is known.

```
\def\bbl@ifknown@ttrib#1#2{%
  \let\bbl@tempa\@secondoftwo
  \bbl@loopx\bbl@tempb{#2}{%
    \expandafter\in@\expandafter{\expandafter,\bbl@tempb,}{,#1,}%
    \ifin@
      \let\bbl@tempa\@firstoftwo
    \else
    \fi}%
  \bbl@tempa}
```

\bbl@clear@ttribs This macro removes all the attribute code from LATEX's memory at `\begin{document}` time (if any is present).

```
\def\bbl@clear@ttribs{%
  \ifx\bbl@attributes\@undefined\else
    \bbl@loopx\bbl@tempa{\bbl@attributes}{%
      \expandafter\bbl@clear@ttrib\bbl@tempa.}%
    \let\bbl@attributes\@undefined
  \fi}
\def\bbl@clear@ttrib#1-#2.{%
  \expandafter\let\csname#1@attr@#2\endcsname\@undefined}
\AtBeginDocument{\bbl@clear@ttribs}
```

6.8. Support for saving and redefining macros

To save the meaning of control sequences using `\babel@save`, we use temporary control sequences.

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

```
\bbl@trace{Macros for saving definitions}
\def\babel@beginsave{\babel@savecnt\z@}
```

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

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

\babel@save

\babel@savevariable The macro `\babel@save⟨csname⟩` saves the current meaning of the control sequence ⟨*csname*⟩ to `\originalTeX` (which has to be expandable, i.e., you shouldn't let it to `\relax`).

```
\def\babel@save#1{%
  \def\bbl@tempa{{,#1,}}% Clumsy, for Plain
  \expandafter\bbl@add\expandafter\bbl@tempa\expandafter{%
    \expandafter{\expandafter,\bbl@savedextras,}}%
  \expandafter\in@\bbl@tempa
  \ifin@\else
    \bbl@add\bbl@savedextras{,#1,}%
    \bbl@carg\let{babel@\number\babel@savecnt}#1\relax
    \toks@\expandafter{\originalTeX\let#1=}%
    \bbl@exp{%
      \def\\\originalTeX{\the\toks@\<babel@\number\babel@savecnt>\relax}}%
    \advance\babel@savecnt\@ne
  \fi}
\def\babel@savevariable#1{%
  \toks@\expandafter{\originalTeX #1=}%
  \bbl@exp{\def\\\originalTeX{\the\toks@\the#1\relax}}}
```

\bbl@redefine To redefine a command, we save the old meaning of the macro.

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

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

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

\bbl@redefinerobust For commands that are redefined, but which *might* be robust we need a slightly more intelligent macro.

```
\def\bbl@redefinerobust#1{%
  \edef\bbl@tempa{\bbl@stripslash#1}%
  \bbl@ifunset{\bbl@tempa\space}%
    {\expandafter\let\csname org@\bbl@tempa\endcsname#1%
     \bbl@exp{\def\\#1{\\\protect\<\bbl@tempa\space>}}}%
    {\bbl@exp{\let\<org@\bbl@tempa>\<\bbl@tempa\space>}}%
    \@namedef{\bbl@tempa\space}}
\@onlypreamble\bbl@redefinerobust
```

6.9. French spacing

\bbl@frenchspacing
\bbl@nonfrenchspacing Some languages need to have `\frenchspacing` in effect.

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

6.10. Hyphens

\babelhyphenation This macro saves hyphenation exceptions.

```
\bbl@trace{Hyphens}
\@onlypreamble\babelhyphenation
\AtEndOfPackage{%
  \newcommand\babelhyphenation[2][\@empty]{%
    \ifx\bbl@hyphenation@\relax
      \let\bbl@hyphenation@\@empty
    \fi
    \ifx\bbl@hyphlist\@empty\else
      \bbl@warning{%
        You must not intermingle \string\selectlanguage\space and\\%
        \string\babelhyphenation\space or some exceptions will not\\%
        be taken into account. Reported}%
    \fi
    \ifx\@empty#1%
      \protected@edef\bbl@hyphenation@{\bbl@hyphenation@\space#2}%
    \else
      \bbl@vforeach{#1}{%
        \def\bbl@tempa{##1}%
        \bbl@fixname\bbl@tempa
        \bbl@iflanguage\bbl@tempa{%
          \bbl@csarg\protected@edef{hyphenation@\bbl@tempa}{%
            \bbl@ifunset{bbl@hyphenation@\bbl@tempa}%
              {}%
              {\csname bbl@hyphenation@\bbl@tempa\endcsname\space}%
            #2}}}%
    \fi}}
```

\bbl@allowhyphens This macro makes hyphenation possible.

```
\def\bbl@allowhyphens{\ifvmode\else\nobreak\hskip\z@skip\fi}
\def\bbl@t@one{T1}
\def\allowhyphens{\ifx\cf@encoding\bbl@t@one\else\bbl@allowhyphens\fi}
```

\babelhyphen Macros to insert common hyphens.

```
\newcommand\babelnullhyphen{\char\hyphenchar\font}
\def\babelhyphen{\active@prefix\babelhyphen\bbl@hyphen}
\def\bbl@hyphen{%
  \@ifstar{\bbl@hyphen@i @}{\bbl@hyphen@i\@empty}}
\def\bbl@hyphen@i#1#2{%
  \lowercase{\bbl@ifunset{bbl@hy@#1#2\@empty}}%
    {\csname bbl@#1usehyphen\endcsname{\discretionary{#2}{}{#2}}}%
    {\lowercase{\csname bbl@hy@#1#2\@empty\endcsname}}}
```

The following two commands are used to wrap the “hyphen” and set the behavior of the rest of the word.

```
\def\bbl@usehyphen#1{%
  \leavevmode
  \ifdim\lastskip>\z@\mbox{#1}\else\nobreak#1\fi
  \nobreak\hskip\z@skip}
\def\bbl@@usehyphen#1{%
  \leavevmode\ifdim\lastskip>\z@\mbox{#1}\else#1\fi}
```

The following macro inserts the hyphen char.

```
\def\bbl@hyphenchar{%
  \ifnum\hyphenchar\font=\m@ne
    \babelnullhyphen
  \else
    \char\hyphenchar\font
  \fi}
```

Finally, we define the hyphen "types". Their names will not change, so you may use them in `ldf`'s. After a space, the `\mbox` in `\bbl@hy@nobreak` is redundant.

```
\def\bbl@hy@soft{\bbl@usehyphen{\discretionary{\bbl@hyphenchar}{}{}}}
\def\bbl@hy@@soft{\bbl@@usehyphen{\discretionary{\bbl@hyphenchar}{}{}}}
\def\bbl@hy@hard{\bbl@usehyphen\bbl@hyphenchar}
\def\bbl@hy@@hard{\bbl@@usehyphen\bbl@hyphenchar}
\def\bbl@hy@nobreak{\bbl@usehyphen{\mbox{\bbl@hyphenchar}}}
\def\bbl@hy@@nobreak{\mbox{\bbl@hyphenchar}}
\def\bbl@hy@repeat{%
  \bbl@usehyphen{%
    \discretionary{\bbl@hyphenchar}{\bbl@hyphenchar}{\bbl@hyphenchar}}}
\def\bbl@hy@@repeat{%
  \bbl@@usehyphen{%
    \discretionary{\bbl@hyphenchar}{\bbl@hyphenchar}{\bbl@hyphenchar}}}
\def\bbl@hy@empty{\hskip\z@skip}
\def\bbl@hy@@empty{\discretionary{}{}{}}
```

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

```
\def\bbl@disc#1#2{\nobreak\discretionary{#2-}{}{#1}\bbl@allowhyphens}
```

6.11. Multiencoding strings

The aim following commands is to provide a common interface for strings in several encodings.

Tools But first, a tool. It makes global a local variable. This is not the best solution, but it works.

```
\bbl@trace{Multiencoding strings}
\def\bbl@toglobal#1{\global\let#1#1}
```

Main command This is the main command.

```
\@onlypreamble\StartBabelCommands
\def\StartBabelCommands{%
  \begingroup
  \@tempcnta="7F
  \def\bbl@tempa{%
    \ifnum\@tempcnta>"FF\else
      \catcode\@tempcnta=11
      \advance\@tempcnta\@ne
      \expandafter\bbl@tempa
    \fi}%
  \bbl@tempa
  <@Macros local to Plain BabelCommands@>
  \def\bbl@provstring##1##2{%
    \providecommand##1{##2}%
    \bbl@toglobal##1}%
  \global\let\bbl@scafter\@empty
  \let\StartBabelCommands\bbl@startcmds
  \ifx\BabelLanguages\relax
     \let\BabelLanguages\CurrentOption
  \fi
  \begingroup
  \let\bbl@screset\@nnil % local flag - disable 1st stopcommands
  \StartBabelCommands}
\def\bbl@startcmds{%
  \ifx\bbl@screset\@nnil\else
    \bbl@usehooks{stopcommands}{}%
  \fi
```

```
  \endgroup
  \begingroup
  \@ifstar
    {\ifx\bbl@opt@strings\@nnil
       \let\bbl@opt@strings\BabelStringsDefault
     \fi
     \bbl@startcmds@i}%
    \bbl@startcmds@i}
\def\bbl@startcmds@i#1#2{%
  \edef\bbl@L{\zap@space#1 \@empty}%
  \edef\bbl@G{\zap@space#2 \@empty}%
  \bbl@startcmds@ii}
\let\bbl@startcommands\StartBabelCommands
```

Parse the encoding info to get the label, input, and font parts.

```
\newcommand\bbl@startcmds@ii[1][\@empty]{%
  \let\SetString\@gobbletwo
  \let\bbl@stringdef\@gobbletwo
  \let\AfterBabelCommands\@gobble
  \ifx\@empty#1%
    \def\bbl@sc@label{generic}%
    \def\bbl@encstring##1##2{%
      \ProvideTextCommandDefault##1{##2}%
      \bbl@toglobal##1%
      \expandafter\bbl@toglobal\csname\string?\string##1\endcsname}%
    \let\bbl@sctest\in@true
  \else
    \let\bbl@sc@charset\space % <- zapped below
    \let\bbl@sc@fontenc\space % <-   "      "
    \def\bbl@tempa##1=##2\@nil{%
      \bbl@csarg\edef{sc@\zap@space##1 \@empty}{##2 }}%
    \bbl@vforeach{label=#1}{\bbl@tempa##1\@nil}%
    \def\bbl@tempa##1 ##2{% space -> comma
      ##1%
      \ifx\@empty##2\else\ifx,##1,\else,\fi\bbl@afterfi\bbl@tempa##2\fi}%
    \edef\bbl@sc@fontenc{\expandafter\bbl@tempa\bbl@sc@fontenc\@empty}%
    \edef\bbl@sc@label{\expandafter\zap@space\bbl@sc@label\@empty}%
    \edef\bbl@sc@charset{\expandafter\zap@space\bbl@sc@charset\@empty}%
    \def\bbl@encstring##1##2{%
      \bbl@foreach\bbl@sc@fontenc{%
        \bbl@ifunset{T@####1}%
          {}%
          {\ProvideTextCommand##1{####1}{##2}%
           \bbl@toglobal##1%
           \expandafter
           \bbl@toglobal\csname####1\string##1\endcsname}}}%
    \def\bbl@sctest{%
      \bbl@xin@{,\bbl@opt@strings,}{,\bbl@sc@label,\bbl@sc@fontenc,}}%
  \fi
  \ifx\bbl@opt@strings\@nnil         % i.e., no strings key -> defaults
  \else\ifx\bbl@opt@strings\relax    % i.e., strings=encoded
    \let\AfterBabelCommands\bbl@aftercmds
    \let\SetString\bbl@setstring
    \let\bbl@stringdef\bbl@encstring
  \else       % i.e., strings=value
  \bbl@sctest
  \ifin@
    \let\AfterBabelCommands\bbl@aftercmds
    \let\SetString\bbl@setstring
    \let\bbl@stringdef\bbl@provstring
  \fi\fi\fi
  \bbl@scswitch
  \ifx\bbl@G\@empty
```

```
    \def\SetString##1##2{%
      \bbl@error{missing-group}{##1}{}{}}%
  \fi
  \ifx\@empty#1%
    \bbl@usehooks{defaultcommands}{}%
  \else
    \@expandtwoargs
    \bbl@usehooks{encodedcommands}{{\bbl@sc@charset}{\bbl@sc@fontenc}}%
  \fi}
```

There are two versions of `\bbl@scswitch`.

```
\def\bbl@forlang#1#2{%
  \bbl@for#1\bbl@L{%
    \bbl@xin@{,#1,}{,\BabelLanguages,}%
    \ifin@#2\relax\fi}}
\def\bbl@scswitch{%
  \bbl@forlang\bbl@tempa{%
    \ifx\bbl@G\@empty\else
      \ifx\SetString\@gobbletwo\else
        \edef\bbl@GL{\bbl@G\bbl@tempa}%
        \bbl@xin@{,\bbl@GL,}{,\bbl@screset,}%
        \ifin@\else
          \global\expandafter\let\csname\bbl@GL\endcsname\@undefined
          \xdef\bbl@screset{\bbl@screset,\bbl@GL}%
        \fi
      \fi
    \fi}}
\AtEndOfPackage{%
  \def\bbl@forlang#1#2{\bbl@for#1\bbl@L{\bbl@ifunset{date#1}{}{#2}}}%
  \let\bbl@scswitch\relax}
\@onlypreamble\EndBabelCommands
\def\EndBabelCommands{%
  \bbl@usehooks{stopcommands}{}%
  \endgroup
  \endgroup
  \bbl@scafter}
\let\bbl@endcommands\EndBabelCommands
```

Now we define commands to be used inside `\StartBabelCommands`.

Strings The following macro is the actual definition of `\SetString` when it is "active",

```
\def\bbl@setstring#1#2{% e.g., \prefacename{<string>}
  \bbl@forlang\bbl@tempa{%
    \edef\bbl@LC{\bbl@tempa\bbl@stripslash#1}%
    \bbl@ifunset{\bbl@LC}% e.g., \germanchaptername
      {\bbl@exp{%
         \global\\\bbl@add\<\bbl@G\bbl@tempa>{\\\bbl@scset\\#1\<\bbl@LC>}}}%
      {}%
    \def\BabelString{#2}%
    \bbl@usehooks{stringprocess}{}%
    \expandafter\bbl@stringdef
      \csname\bbl@LC\expandafter\endcsname\expandafter{\BabelString}}}
```

A little auxiliary command sets the string. Formerly used with casing. Very likely no longer necessary, although it's used in `\setlocalecaption`.

```
\def\bbl@scset#1#2{\def#1{#2}}
```

Define `\SetStringLoop`, which is actually set inside `\StartBabelCommands`.

```
⟨*Macros local to Plain BabelCommands⟩  ▸
\def\SetStringLoop##1##2{%
    \def\bbl@templ####1{\expandafter\noexpand\csname##1\endcsname}%
    \count@\z@
    \bbl@loop\bbl@tempa{##2}{% empty items and spaces are ok
```

```
      \advance\count@\@ne
      \toks@\expandafter{\bbl@tempa}%
      \bbl@exp{%
        \\\SetString\bbl@templ{\romannumeral\count@}{\the\toks@}%
        \count@=\the\count@\relax}}%
⟨⟨/Macros local to Plain BabelCommands⟩⟩
```

Delaying code Now the definition of `\AfterBabelCommands` when it is activated.

```
\def\bbl@aftercmds#1{%
  \toks@\expandafter{\bbl@scafter#1}%
  \xdef\bbl@scafter{\the\toks@}}
```

Case mapping The command `\SetCase` is deprecated (at least in LaTeX). This is the original definition.

```
\def\bbl@patchuclc{%
  \global\let\bbl@patchuclc\relax
  \g@addto@macro\@uclclist{\reserved@b{\reserved@b\bbl@uclc}}%
  \gdef\bbl@uclc##1{%
    \let\bbl@encoded\bbl@encoded@uclc
    \bbl@ifunset{\languagename @bbl@uclc}% and resumes it
      {##1}%
      {\let\bbl@tempa##1\relax % Used by LANG@bbl@uclc
       \csname\languagename @bbl@uclc\endcsname}%
    {\bbl@tolower\@empty}{\bbl@toupper\@empty}}%
  \gdef\bbl@tolower{\csname\languagename @bbl@lc\endcsname}%
  \gdef\bbl@toupper{\csname\languagename @bbl@uc\endcsname}}
⟨⟨*Macros local to Plain BabelCommands⟩⟩  ▸
  \newcommand\SetCase[3][]{%
    \bbl@patchuclc
    \bbl@forlang\bbl@tempa{%
      \bbl@carg\bbl@encstring{\bbl@tempa @bbl@uclc}{\bbl@tempa##1}%
      \bbl@carg\bbl@encstring{\bbl@tempa @bbl@uc}{##2}%
      \bbl@carg\bbl@encstring{\bbl@tempa @bbl@lc}{##3}}}%
⟨⟨/Macros local to Plain BabelCommands⟩⟩
```

Macros to deal with case mapping for hyphenation.

```
⟨⟨*Macros local to Plain BabelCommands⟩⟩  ▸
  \newcommand\SetHyphenMap[1]{%
    \bbl@forlang\bbl@tempa{%
      \expandafter\bbl@stringdef
        \csname\bbl@tempa @bbl@hyphenmap\endcsname{##1}}}%
⟨⟨/Macros local to Plain BabelCommands⟩⟩
```

There are 3 helper macros which do most of the work for you.

```
\newcommand\BabelLower[2]{% one to one.
  \ifnum\lccode#1=#2\else
    \babel@savevariable{\lccode#1}%
    \lccode#1=#2\relax
  \fi}
\newcommand\BabelLowerMM[4]{% many-to-many
  \@tempcnta=#1\relax
  \@tempcntb=#4\relax
  \def\bbl@tempa{%
    \ifnum\@tempcnta>#2\else
      \@expandtwoargs\BabelLower{\the\@tempcnta}{\the\@tempcntb}%
      \advance\@tempcnta#3\relax
      \advance\@tempcntb#3\relax
      \expandafter\bbl@tempa
    \fi}%
  \bbl@tempa}
\newcommand\BabelLowerMO[4]{% many-to-one
```

```
  \@tempcnta=#1\relax
  \def\bbl@tempa{%
    \ifnum\@tempcnta>#2\else
      \@expandtwoargs\BabelLower{\the\@tempcnta}{#4}%
      \advance\@tempcnta#3
      \expandafter\bbl@tempa
    \fi}%
  \bbl@tempa}
```

Initial setup to provide a default behavior if hyphenmap is not set.

```
\AtEndOfPackage{%
  \ifx\bbl@opt@hyphenmap\@undefined
    \bbl@xin@{,}{\bbl@language@opts}%
    \chardef\bbl@opt@hyphenmap\ifin@4\else\@ne\fi
  \fi}
```

6.12. Tailor captions

A general tool for resetting the caption names with a unique interface.

```
\newcommand\setlocalecaption{%
  \@ifstar\bbl@setcaption@s\bbl@setcaption@x}
\def\bbl@setcaption@x#1#2#3{%  language caption-name string
  \bbl@trim@def\bbl@tempa{#2}%
  \bbl@xin@{.template}{\bbl@tempa}%
  \ifin@
    \bbl@ini@captions@template{#3}{#1}%
  \else
    \edef\bbl@tempd{%
      \expandafter\expandafter\expandafter
      \strip@prefix\expandafter\meaning\csname captions#1\endcsname}%
    \bbl@xin@
      {\expandafter\string\csname #2name\endcsname}%
      {\bbl@tempd}%
    \ifin@ % Renew caption
      \bbl@xin@{\string\bbl@scset}{\bbl@tempd}%
      \ifin@
        \bbl@exp{%
          \\\bbl@ifsamestring{\bbl@tempa}{\languagename}%
            {\\\bbl@scset\<#2name>\<#1#2name>}%
            {}}%
      \else % Old way converts to new way
        \bbl@ifunset{#1#2name}%
          {\bbl@exp{%
            \\\bbl@add\<captions#1>{\def\<#2name>{\<#1#2name>}}%
            \\\bbl@ifsamestring{\bbl@tempa}{\languagename}%
              {\def\<#2name>{\<#1#2name>}}%
              {}}}%
          {}%
      \fi
    \else
      \bbl@xin@{\string\bbl@scset}{\bbl@tempd}% New
      \ifin@ % New way
        \bbl@exp{%
          \\\bbl@add\<captions#1>{\\\bbl@scset\<#2name>\<#1#2name>}%
          \\\bbl@ifsamestring{\bbl@tempa}{\languagename}%
            {\\\bbl@scset\<#2name>\<#1#2name>}%
            {}}%
      \else  % Old way, but defined in the new way
        \bbl@exp{%
          \\\bbl@add\<captions#1>{\def\<#2name>{\<#1#2name>}}%
          \\\bbl@ifsamestring{\bbl@tempa}{\languagename}%
            {\def\<#2name>{\<#1#2name>}}%
```

```
            {}}%
      \fi%
    \fi
    \@namedef{#1#2name}{#3}%
    \toks@\expandafter{\bbl@captionslist}%
    \bbl@exp{\\\in@{\<#2name>}{\the\toks@}}%
    \ifin@\else
      \bbl@exp{\\\bbl@add\\\bbl@captionslist{\<#2name>}}%
      \bbl@toglobal\bbl@captionslist
    \fi
  \fi}
```

6.13. Making glyphs available

This section 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.

\set@low@box The following macro is used to lower quotes to the same level as the comma.

```
\bbl@trace{Macros related to glyphs}
\def\set@low@box#1{\setbox\tw@\hbox{,}\setbox\z@\hbox{#1}%
    \dimen\z@\ht\z@ \advance\dimen\z@ -\ht\tw@%
    \setbox\z@\hbox{\lower\dimen\z@ \box\z@}\ht\z@\ht\tw@ \dp\z@\dp\tw@}
```

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

```
\def\save@sf@q#1{\leavevmode
  \begingroup
    \edef\@SF{\spacefactor\the\spacefactor}#1\@SF
  \endgroup}
```

6.13.1. Quotation marks

\quotedblbase In the OT1 encoding it is not available, therefore we make it available by lowering the normal open quote character to the baseline.

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

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

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

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

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

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

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

\guillemetleft

\guillemetright The guillemet characters are not available in OT1 encoding.

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

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

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

\guilsinglleft
\guilsinglright The single guillemets are not available in OT1 encoding. They are faked.

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

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

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

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

6.13.2. Letters

\ij
\IJ The dutch language uses the letter 'ij'.

```
\DeclareTextCommand{\ij}{OT1}{%
  i\kern-0.02em\bbl@allowhyphens j}
\DeclareTextCommand{\IJ}{OT1}{%
  I\kern-0.02em\bbl@allowhyphens J}
\DeclareTextCommand{\ij}{T1}{\char188}
\DeclareTextCommand{\IJ}{T1}{\char156}
```

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

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

\dj
\DJ The croatian language needs the letters \dj and \DJ,

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

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

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

\SS For the T1 encoding \SS is defined and selects a specific glyph from the font, but for other encodings it is not available.

```
\DeclareTextCommand{\SS}{OT1}{SS}
\ProvideTextCommandDefault{\SS}{\UseTextSymbol{OT1}{\SS}}
```

6.13.3. Shorthands for quotation marks

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

\glq

\grq The ‘german’ single quotes.

```
\ProvideTextCommandDefault{\glq}{%
  \textormath{\quotesinglbase}{\mbox{\quotesinglbase}}}
```

The definition of `\grq` depends on the fontencoding. With `T1` encoding no extra kerning is needed.

```
\ProvideTextCommand{\grq}{T1}{%
  \textormath{\kern\z@\textquoteleft}{\mbox{\textquoteleft}}}
\ProvideTextCommand{\grq}{TU}{%
  \textormath{\textquoteleft}{\mbox{\textquoteleft}}}
\ProvideTextCommand{\grq}{OT1}{%
  \save@sf@q{\kern-.0125em
    \textormath{\textquoteleft}{\mbox{\textquoteleft}}%
    \kern.07em\relax}}
\ProvideTextCommandDefault{\grq}{\UseTextSymbol{OT1}\grq}
```

\glqq
\grqq The ‘german’ double quotes.

```
\ProvideTextCommandDefault{\glqq}{%
  \textormath{\quotedblbase}{\mbox{\quotedblbase}}}
```

The definition of `\grqq` depends on the fontencoding. With `T1` encoding no extra kerning is needed.

```
\ProvideTextCommand{\grqq}{T1}{%
  \textormath{\textquotedblleft}{\mbox{\textquotedblleft}}}
\ProvideTextCommand{\grqq}{TU}{%
  \textormath{\textquotedblleft}{\mbox{\textquotedblleft}}}
\ProvideTextCommand{\grqq}{OT1}{%
  \save@sf@q{\kern-.07em
    \textormath{\textquotedblleft}{\mbox{\textquotedblleft}}%
    \kern.07em\relax}}
\ProvideTextCommandDefault{\grqq}{\UseTextSymbol{OT1}\grqq}
```

\flq
\frq The ‘french’ single guillemets.

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

\flqq
\frqq The ‘french’ double guillemets.

```
\ProvideTextCommandDefault{\flqq}{%
  \textormath{\guillemetleft}{\mbox{\guillemetleft}}}
\ProvideTextCommandDefault{\frqq}{%
  \textormath{\guillemetright}{\mbox{\guillemetright}}}
```

6.13.4. Umlauts and tremas

The command `\"` needs to have a different effect for different languages.

\umlauthigh

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

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

\lower@umlaut Used to position the \" closer to the letter.

```
\expandafter\ifx\csname U@D\endcsname\relax
  \csname newdimen\endcsname\U@D
\fi
\def\lower@umlaut#1{%
  \leavevmode\bgroup
    \U@D 1ex%
    {\setbox\z@\hbox{%
      \char\csname\f@encoding dqpos\endcsname}%
      \dimen@ -.45ex\advance\dimen@\ht\z@
      \ifdim 1ex<\dimen@ \fontdimen5\font\dimen@ \fi}%
    \accent\csname\f@encoding dqpos\endcsname
    \fontdimen5\font\U@D #1%
  \egroup}
\AtBeginDocument{%
  \DeclareTextCompositeCommand{\"}{OT1}{a}{\bbl@umlauta{a}}%
  \DeclareTextCompositeCommand{\"}{OT1}{e}{\bbl@umlaute{e}}%
  \DeclareTextCompositeCommand{\"}{OT1}{i}{\bbl@umlaute{\i}}%
  \DeclareTextCompositeCommand{\"}{OT1}{\i}{\bbl@umlaute{\i}}%
  \DeclareTextCompositeCommand{\"}{OT1}{o}{\bbl@umlauta{o}}%
  \DeclareTextCompositeCommand{\"}{OT1}{u}{\bbl@umlauta{u}}%
  \DeclareTextCompositeCommand{\"}{OT1}{A}{\bbl@umlauta{A}}%
  \DeclareTextCompositeCommand{\"}{OT1}{E}{\bbl@umlaute{E}}%
  \DeclareTextCompositeCommand{\"}{OT1}{I}{\bbl@umlaute{I}}%
  \DeclareTextCompositeCommand{\"}{OT1}{O}{\bbl@umlauta{O}}%
  \DeclareTextCompositeCommand{\"}{OT1}{U}{\bbl@umlauta{U}}}
```

Finally, make sure the default hyphenrules are defined (even if empty).

```
\ifx\l@english\@undefined
  \chardef\l@english\z@
\fi
\ifx\l@unhyphenated\@undefined
  \newlanguage\l@unhyphenated
\fi
```

6.14. Layout

Layout is mainly intended to set bidi documents, but there is at least a tool useful in general.

```
\bbl@trace{Bidi layout}
\providecommand\IfBabelLayout[3]{#3}%
```

6.15. Load engine specific macros

Some macros are not defined in all engines, so, after loading the files define them if necessary to raise an error.

```
\bbl@trace{Input engine specific macros}
```

```
\ifcase\bbl@engine\or
  \input luababel.def
\or
  \input xebabel.def
\fi
\providecommand\babelfont{\bbl@error{only-lua-xe}{}{}{}}
\providecommand\babelprehyphenation{\bbl@error{only-lua}{}{}{}}
\ifx\babelposthyphenation\@undefined
  \let\babelposthyphenation\babelprehyphenation
  \let\babelpatterns\babelprehyphenation
  \let\babelcharproperty\babelprehyphenation
\fi
⟨/core⟩
```

7. Additional support for Plain TEX

7.1. Not renaming `hyphen.tex`

As Don Knuth has declared that the filename `hyphen.tex` may only be used to designate *his* version of the american English hyphenation patterns, a new solution has to be found in order to be able to load hyphenation patterns for other languages in a plain-based TEX-format. When asked he responded:

> That file name is “sacred”, and if anybody changes it they will cause severe upward/downward compatibility headaches.
>
> People can have a file localhyphen.tex or whatever they like, but they mustn’t diddle with hyphen.tex (or plain.tex except to preload additional fonts).

The files `bplain.tex` and `blplain.tex` can be used as replacement wrappers around `plain.tex` and `lplain.tex` to achieve the desired effect, based on the `babel` package. If you load each of them with iniTEX, you will get a file called either `bplain.fmt` or `blplain.fmt`, which you can use as replacements for `plain.fmt` and `lplain.fmt`.

As these files are going to be read as the first thing iniTEX sees, we need to set some category codes just to be able to change the definition of `\input`.

```
⟨*bplain|blplain⟩
\catcode`\{=1 % left brace is begin-group character
\catcode`\}=2 % right brace is end-group character
\catcode`\#=6 % hash mark is macro parameter character
```

If a file called `hyphen.cfg` can be found, we make sure that *it* will be read instead of the file `hyphen.tex`. We do this by first saving the original meaning of `\input` (and I use a one letter control sequence for that so as not to waste multi-letter control sequence on this in the format).

```
\openin 0 hyphen.cfg
\ifeof0
\else
  \let\a\input
```

Then `\input` is defined to forget about its argument and load `hyphen.cfg` instead. Once that’s done the original meaning of `\input` can be restored and the definition of `\a` can be forgotten.

```
  \def\input #1 {%
    \let\input\a
    \a hyphen.cfg
    \let\a\undefined
  }
\fi
⟨/bplain|blplain⟩
```

Now that we have made sure that `hyphen.cfg` will be loaded at the right moment it is time to load `plain.tex`.

```
2192 ⟨bplain⟩\a plain.tex
2193 ⟨blplain⟩\a lplain.tex
```

Finally we change the contents of `\fmtname` to indicate that this is *not* the plain format, but a format based on plain with the `babel` package preloaded.

```
2194 ⟨bplain⟩\def\fmtname{babel-plain}
2195 ⟨blplain⟩\def\fmtname{babel-lplain}
```

When you are using a different format, based on plain.tex you can make a copy of blplain.tex, rename it and replace `plain.tex` with the name of your format file.

8. Compatibility

A couple of proxy files, for compatibility (`plain.def` and `switch.def`).

```
2196 ⟨*plain⟩
2197 \input babel.def
2198 ⟨/plain⟩
2199 ⟨*kernel⟩
2200 \let\bbl@onlyswitch\@empty
2201 \input babel.def
2202 \let\bbl@onlyswitch\@undefined
2203 ⟨/kernel⟩
```