A Babel language definition file for French

frenchb.dtx v3.0a, 2014/02/28

Daniel Flipo
daniel.flipo@free.fr

Contents

1 The French language

The file `frenchb.dtx`[1], defines all the language definition macros for the French language.
Customisation for the French language is achieved following the book “Lexique des règles typographiques en usage à l’Imprimerie nationale” troisième édition (1994), ISBN-2-11-081075-0.
First version released: 1.1 (May 1996) as part of babel-3.6beta. Version 2.0a was released in February 2007.
frenchb has been improved using helpful suggestions from many people, mainly from Jacques André, Michel Bovani, Thierry Bouche, Vincent Jalby and Denis Bitouzé. Thanks to all of them!
LaTeX-2.09 is no longer supported. This new version (3.x) has been designed to be used only with LaTeX 2ε and Plain formats based on TeX, pdfTeX, LuaTeX or XeTeX engines.
Changes between version 2.0 and v3.0a are listed in subsection 1.4 p. 8.
An extensive documentation is available in French here:
http://daniel.flipo.free.fr/frenchb

1.1 Basic interface

In a multilingual document, some typographic rules are language dependent, i.e. spaces before ‘high punctuation’ (`: ; ! ?`) in French, others modify the general layout (i.e. layout of lists, footnotes, indentation of first paragraphs of sections) and should apply to the whole document.
frenchb takes account of babel’s *main language* defined as the *last* option at babel’s loading. When French is not babel’s main language, frenchb does not alter the general layout of the document (even in parts where French is the current language): the layout of lists, footnotes, indentation of first paragraphs of sections are not customised by frenchb.
When French is loaded as the last option of babel, frenchb makes the following changes to the global layout, *both in French and in all other languages*[2]:

1. the first paragraph of each section is indented (LaTeX only);
2. the default items in itemize environment are set to ‘—’ instead of ‘•’, and all vertical spacing and glue is deleted; it is possible to change ‘—’ to something else (‘–’ for instance) using `\frenchbsetup{}` (see section 1.2 p. 4);
3. vertical spacing in general LaTeX lists is shortened;
4. footnotes are displayed “à la française”.
5. the separator following the table or figure number in captions is printed as ‘ – ’ instead of ‘: ’; for changing this see 1.2.2 p. 7.

[1] The file described in this section has version number v3.0a and was last revised on 2014/02/28.
[2] For each item, hooks are provided to reset standard LaTeX settings or to emulate the behavior of former versions of frenchb (see command `\frenchbsetup{}`, section 1.2 p. 4).

Regarding local typography, the command `\selectlanguage{french}` switches to the French language[3], with the following effects:

1. French hyphenation patterns are made active;
2. 'high punctuation' characters (`: ; ! ?`) automatically add correct spacing in French; this is achieved using callbacks in Lua(La)TeX or 'XeTeXinterchar' mechanism in Xe(La)TeX; with TeX'82 and pdf(La)TeX these four characters are made active in the whole document;
3. `\today` prints the date in French;
4. the caption names are translated into French (LaTeX only). For customisation of caption names see section 1.2.2 p. 7.
5. the space after `\dots` is removed in French.

Some commands are provided in frenchb to make typesetting easier:

1. French quotation marks can be entered using the commands `\og` and `\fg` which work in LaTeX 2$_\varepsilon$and PlainTeX, their appearance depending on what is available to draw them; even if you use LaTeX 2$_\varepsilon$ *and* `T1`-encoding, you should refrain from entering them as `<<~French quotation marks~>>`: `\og` and `\fg` provide better horizontal spacing. `\og` and `\fg` can be used outside French, they typeset then English quotes ‘ and ’.
2. A command `\up` is provided to typeset superscripts like `M\up{me}` (abbreviation for "Madame"), `1\up{er}` (for "premier"). Other commands are also provided for ordinals: `\ier`, `\iere`, `\iers`, `\ieres`, `\ieme`, `\iemes` (`3\iemes` prints 3$^{\text{es}}$).
3. Family names should be typeset in small capitals and never be hyphenated, the macro `\bsc` (boxed small caps) does this, e.g., `L.~\bsc{Lamport}` will print the same as `L.~\mbox{\textsc{Lamport}}`. Note that composed names (such as Dupont-Durant) may now be hyphenated on explicit hyphens, this differs from frenchb v. 1.x.
4. Commands `\primo`, `\secundo`, `\tertio` and `\quarto` print 1$^{\text{o}}$, 2$^{\text{o}}$, 3$^{\text{o}}$, 4$^{\text{o}}$. `\FrenchEnumerate{6}` prints 6$^{\text{o}}$.
5. Abbreviations for "Numéro(s)" and "numéro(s)" (N$^{\text{o}}$ N$^{\text{os}}$ n$^{\text{o}}$ and n$^{\text{os}}$) are obtained via the commands `\No`, `\Nos`, `\no`, `\nos`.
6. Two commands are provided to typeset the symbol for "degré": `\degre` prints the raw character and `\degres` should be used to typeset temperatures (e.g., "`20~\degres C`" with an unbreakable space), or for alcohols" strengths (e.g., "`45\degres`" with *no* space in French).
7. In math mode the comma has to be surrounded with braces to avoid a spurious space being inserted after it, in decimal numbers for instance (see the TeXbook p. 134). The command `\DecimalMathComma` makes the comma be an ordinary character *in French only* (no space added); as a

[3] `\selectlanguage{francais}` and `\selectlanguage{frenchb}` are kept for backward compatibility but should no longer be used.

counterpart, if `\DecimalMathComma` is active, an explicit space has to be added in lists and intervals: `$[0,\ 1]$`, `$(x,\ y)$`. `\StandardMathComma` switches back to the standard behaviour of the comma.

8. A command `\nombre` was provided in 1.x versions to easily format numbers in slices of three digits separated either by a comma in English or with a space in French; `\nombre` is now mapped to `\numprint` from `numprint.sty`, see `numprint.pdf` for more information.

9. frenchb has been designed to take advantage of the `xspace` package if present: adding `\usepackage{xspace}` in the preamble will force macros like `\fg`, `\ier`, `\ieme`, `\dots`, `...`, to respect the spaces you type after them, for instance typing '`1\ier juin`' will print '1er juin' (no need for a forced space after `1\ier`).

1.2 Customisation

Customisation of frenchb relies on command `\frenchbsetup{}`, options are entered using the `keyval` syntax. The command `\frenchbsetup{}` is to appear in the preamble only (after loading babel).

1.2.1 `\frenchbsetup{`⟨*options*⟩`}`

`\frenchbsetup{ShowOptions}` prints all available options to the `.log` file, it is just meant as a remainder of the list of offered options. As usual with `keyval` syntax, boolean options (as `ShowOptions`) can be entered as `ShowOptions=true` or just `ShowOptions`, the `=true` part can be omitted.

The other options are listed below. Their default value is shown between braces, sometimes followed be a '∗'. The '∗' means that the default shown applies when frenchb is loaded as the *last* option of babel —babel's *main language*—, and is toggled otherwise.

`StandardLayout=true (false∗)` forces frenchb not to interfere with the layout: no action on any kind of lists, first paragraphs of sections are not indented (as in English), no action on footnotes. This option can be used to avoid conflicts with classes or packages which customise lists or footnotes.

`GlobalLayoutFrench=false (true∗)` should no longer be used; it was intended to emulate, when French is the main language, what prior versions of frenchb (pre-2.2) did: lists, and first paragraphs of sections would be displayed the standard way in other languages than French, and "à la française" in French. Note that the layout of footnotes is language independent anyway (see below `FrenchFootnotes` and `AutoSpaceFootnotes`).

`ReduceListSpacing=false (true∗)` ; frenchb reduces the values of the vertical spaces used in the *all* list environments in French (this includes itemize, enumerate, description, but also abstract, quote, quotation and verse and possibly others). Setting this option to `false` reverts to the standard settings of the `list` environment.

`ListOldLayout=true (false)` ; starting with version 2.6a, the layout of lists has changed regarding leftmargins' sizes and default itemize label ('—' instead of '–' up to 2.5k). This option, provided for backward compatibility, displays lists as they were up to version 2.5k.

`CompactItemize=false (true*)` ; should no longer be used (kept only for backward compatibility), it is replaced by the next two options.

`StandardItemizeEnv=true (false*)` ; frenchb redefines the `itemize` environment to suppress any vertical space between items of `itemize` lists in French and customises left margins. Setting this option to `false` reverts to the standard definition of `itemize`.

`StandardEnumerateEnv=true (false*)` ; starting with version 2.6 frenchb redefines the `enumerate` and `description` environments to make left margins match those of the French version of `itemize` lists. Setting this option to `false` reverts to the standard definition of `enumerate` and `description`.

`StandardItemLabels=true (false*)` when set to `true` this option prevents frenchb from changing the labels in `itemize` lists in French.

`ItemLabels=\textbullet, \textendash, \ding{43},...(\textemdash*)` ; when `StandardItemLabels=false` (the default), this option enables to choose the label used in French `itemize` lists for all levels. The next four options do the same but each one for a specific level only. Note that the example `\ding{43}` requires `\usepackage{pifont}`.

`ItemLabeli=\textbullet, \textendash, \ding{43},...(\textemdash*)`

`ItemLabelii=\textbullet, \textendash, \ding{43},...(\textemdash*)`

`ItemLabeliii=\textbullet, \textendash, \ding{43},..(\textemdash*)`

`ItemLabeliv=\textbullet, \textendash, \ding{43},...(\textemdash*)`

`StandardLists=true (false*)` forbids frenchb to customise any kind of list. Try the option `StandardLists` in case of conflicts with classes or packages that customise lists too. This option is just a shorthand setting all four options `ReduceListSpacing=false`, `StandardItemizeEnv=true`, `StandardEnumerateEnv=true` and `StandardItemLabels=true`.

`IndentFirst=false (true*)` ; set this option to `false` if you do not want frenchb to force indentation of the first paragraph of sections. When French is the main language, this options applies to all languages.

`FrenchFootnotes=false (true*)` reverts to the standard layout of footnotes. By default frenchb typesets leading numbers as '1. ' instead of '[1]', but has no effect on footnotes numbered with symbols (as in the `\thanks` command). Two commands `\StandardFootnotes` and `\FrenchFootnotes` are available to change the layout of footnotes locally; `\StandardFootnotes` can help when some footnotes are numbered with letters (inside minipages for instance).

`AutoSpaceFootnotes=false (true*)` ; by default frenchb adds a thin space in the running text before the number or symbol calling the footnote. Making this option `false` reverts to the standard setting (no space added).

`FrenchSuperscripts=false (true)` ; then `\up=\textsuperscript` (option added in version 2.1). Should only be made `false` to recompile older documents. By default `\up` now relies on `\fup` designed to produce better looking superscripts.

`AutoSpacePunctuation=false (true)` ; in French, the user *should* input a space before the four characters ':;!?' but as many people forget about it (even among native French writers!), the default behaviour of frenchb is to automatically add a `\thinspace` before ';' '!' '?' and a normal (unbreakable) space before ':' as recommended by the French Imprimerie nationale. This is convenient in most cases but can lead to addition of spurious spaces in URLs, in MS-DOS paths or in timetables (10:55), except if they are typed in `\texttt` or verbatim mode. When the current font is a monospaced (typewriter) font, `AutoSpacePunctuation` is locally switched to `false`, no spurious space is added in that case, so the default behaviour of of frenchb in that area should be fine in most circumstances.

Choosing `AutoSpacePunctuation=false` will ensure that a proper space will be added before ':;!?' *if and only if* a (normal) space has been typed in. Those who are unsure about their typing in this area should stick to the default option and use the provided `\NoAutoSpacing` command inside a group in case an unwanted space is added by frenchb (i.e. `{\NoAutoSpacing 10:55}`).

`ThinColonSpace=true (false)` changes the inter-word unbreakable space added before the colon ':' to a thin space, so that the same amount of space is added before any of the four 'high punctuation' characters. The default setting is supported by the French Imprimerie nationale.

`LowercaseSuperscripts=false (true)` ; by default frenchb inhibits the uppercasing of superscripts (for instance when they are moved to page headers). Making this option `false` will disable this behaviour (not recommended).

`PartNameFull=false (true)` ; when true, frenchb numbers the title of `\part{}` commands as "Première partie", "Deuxième partie" and so on. With some classes which change the `\part{}` command (AMS classes do so), you could get "Première partie 1", "Deuxième partie 2" in the toc; when this occurs, this option should be set to `false`, part titles will then be printed as "Partie I", "Partie II".

`CustomiseFigTabcaptions=false (true*)` ; when `false` the default separator (colon) is used instead of `\CaptionSeparator`. Anyway, frenchb makes sure that the colon will be typeset with proper preceeding space in French.

`OldFigTabcaptions=true (false)` is to be used when figures' and tables' captions must be typeset as with pre 3.0 versions of frenchb (with `\CaptionSeparator` in French and colon otherwise). Intended for standard LaTeX classes only.

`SuppressWarning=true (false)` ; can be turned to `true` if you are bored with frenchb's warnings.

`og=«, fg=»` ; when guillemets characters are available on the keyboard (through a compose key for instance), it is nice to use them instead of typing `\og` and `\fg`. This option tells frenchb which characters are opening and closing French guillemets (they depend on the input encoding), then you can type either « `guillemets` » or «`guillemets`» (with or without spaces) to get properly typeset French quotes. This option works with LuaLaTeX and XeLaTeX; with pdfLaTeX it requires `inputenc` to be loaded with a proper encoding: 8-bits encoding (latin1, latin9, ansinew, applemac,...) or multi-byte encoding (utf8, utf8x).

Options' order – Please remember that options are read in the order they appear in the `\frenchbsetup{}` command. Someone wishing that frenchb leaves the layout of lists and footnotes untouched but caring for indentation of first paragraph of sections should choose `\frenchbsetup{StandardLayout,IndentFirst}` to get the expected layout. The reverse order `\frenchbsetup{IndentFirst,StandardLayout}` would lead to option `IndentFirst` being overwritten by `StandardLayout`.

1.2.2 Captions

Caption names can be customised in French using the simplified syntax introduced by babel 3.9, for instance: `\def\frenchproofname{Preuve}`. The older syntax `\addto\captionsfrench{\def\proofname{Preuve}}` still works. Keep in mind that *only* `french` can be used to redefine captions, even if babel's option was entered as `francais` or `frenchb`.
For *all* languages, when French is the main language, frenchb changes the separator (colon) used in figures' and tables' captions to `\CaptionSeparator` which defaults to ' – ' and can be redefined in the preamble with `\renewcommand*{\CaptionSeparator}{...}`.
When French is not the main language, the colon is preserved for all languages but frenchb makes sure that a proper space is typeset before it.
Two new options are provided: if `CustomiseFigTabCaptions` is set to `false` the colon will be used as separator in all languages, with a proper space before the colon in French. The second option `OldFigTabCaptions` can be set to `true` to print figures' and tables' captions as they were with versions pre 3.0 of frenchb (using `\CaptionSeparator` in French and colon in other languages); this option only makes sense with the standard LaTeX classes `article`, `report` and `book`.

1.3 Hyphenation checks

Once you have built your format, a good precaution would be to perform some basic tests about hyphenation in French. For LaTeX 2$_\varepsilon$ I suggest this:

- run `pdfLaTeX` on the following file, with the encoding suitable for your machine (*my-encoding* will be `latin1` for Unix machines, `ansinew` for PCs running Windows, `applemac` or `latin1` for Macintoshs, or `utf8`...

```
%%% Test file for French hyphenation.
```

```
\documentclass{article}
\usepackage[my-encoding]{inputenc}
\usepackage[T1]{fontenc} % Use LM fonts
\usepackage{lmodern}     % for French
\usepackage[frenchb]{babel}
\begin{document}
\showhyphens{signal container \'ev\'enement alg\`ebre}
\showhyphens{signal container événement algèbre}
\end{document}
```

- check the hyphenations proposed by TEX in your log-file; in French you should get with both 7-bit and 8-bit encodings
  `si-gnal contai-ner évé-ne-ment al-gèbre`.
  Do not care about how accented characters are displayed in the log-file, what matters is the position of the '`-`' hyphen signs *only*.

If they are all correct, your installation (probably) works fine, if one (or more) is (are) wrong, ask a local wizard to see what's going wrong and perform the test again (or e-mail me about what happens).
Frequent mismatches:

- you get `sig-nal con-tainer`, this probably means that the hyphenation patterns you are using are for US-English, not for French;
- you get no hyphen at all in `évé-ne-ment`, this probably means that you are using CM fonts and the macro `\accent` to produce accented characters. Using 8-bits fonts with built-in accented characters avoids this kind of mismatch.

1.4 Changes

What's new in version 3.0?

Many deep changes lead me to step frenchb's version number to 3.0a:

- babel 3.9 is required now to process `frenchb.ldf`, this change allows for cleaner definitions of dates and captions for the Unicode engines LuaTeX and XeTeX and also provides a simpler syntax for end-users, see section 1.2.2 p.7.
- `\frenchbsetup{}` options management has been completely reworked; two new options added.
- Canadian French didn't work as a normal babel's dialect, it should now; btw. the French language should now be loaded as french, *not as* frenchb or francais and preferably as a *global* option of `\documentclass`. Some tolerance still exists in v3.0, but do not rely on it.
- frenchb no longer loads `frenchb.cfg`: customisation should definitely be done using `\frenchbsetup{}` options.
- Description lists labels are now indented; set `\listendentFB=0pt` to get the former layout.

- The last but not least change affects the (recent) LuaTeX-based engines, (this means version 0.76 as included in TL2013 and up): active characters are no longer used in French for ‘high punctuation’. Functionalities and user interface are unchanged.

  Many thanks to Paul Isambert who provided the basis for the lua code (see his presentation at GUT’2010) and kindly reviewed my first drafts suggesting significant improvements.

  Please note that this code, still experimental, is likely to change until LuaTeX itself has reached version 1.0.

What’s new in version 2.6?

The way frenchb handles list environments has been completely redesigned in version 2.6 due to a long standing bug affecting enumerate lists inside itemize lists. Horizontal indentation of itemize, enumerate and description lists differs now from previous versions, an option for backward compatibility is provided: `\frenchbsetup{ListOldLayout}`.
frenchb is now compatible with the `paralist` package.
Regarding the layout of figures’ and tables’ captions, version 2.6c is now fully compatible with AMS and koma-script classes and with `caption` and `floatrow` packages. Starting with version 2.6c, the `frenchb.cfg` file is no longer generated from `frenchb.dtx`, but it is still loaded (if found) for backward compatibility.

What’s new in version 2.5?

The main change is that active characters are no longer used in French with (recent) XeTeX-based engines (they still are with TeX-based engines). All the functionalities (automatic insertion of missing spaces before `:;!?` or bare replacement of typed spaces with suitable unbreabable ones, tuning of the spaces width) remain available and the user interface is unchanged. The use of active characters is replaced by the `\XeTeXinterchartoks` mechanism (adapted from the `polyglossia` package).
A new command `\NoAutoSpacing` has been added. It should be used *inside a group* instead of `\shorthandoff{;:!?}` whenever active characters or automatic spacing of French punctuation or quote characters conflict with other packages; it is designed to work with TeX-, LuaTeX- and XeTeX-based engines.
Bug corrections: `\frenchspacing` and `\nonfrenchspacing` are no longer messed up by `frenchb.ldf`.

What’s new in version 2.4?

A new option `SuppressWarning` has been added (desactivated by default) to suppress warnings if `\@makecaption` has been redefined or if the `bigfoot` package is in use.
French hyphenation patterns are now coded in Unicode, see file `hyph-fr.tex`.
Extra code has been added to deal with hyphenation of the French “apostrophe” with XeTeX and LuaTeX engines.
Better compatibility with the `enumitem` package.

When typewriter fonts are in use (hence in verbatim mode) no space is added after '«' and before '»' when they are entered as characters (see `\frenchbsetup{}`).

What's new in version 2.3?

Starting with version 2.3a, frenchb no longer inserts spaces automatically before ':;!?' when a typewriter font is in use; this was suggested by Yannis Haralambous to prevent spurious spaces in computer source code or expressions like `C:/foo`, `http://foo.bar`, etc. An option (`OriginalTypewriter`) is provided to get back to the former behaviour of frenchb.
Another probably invisible change: lowercase conversion in `\up{}` is now achieved by the LaTeX command `\MakeLowercase` instead of TeX's `\lowercase` command. This prevents error messages when diacritics are used inside `\up{}` (diacritics should *never* be used in superscripts though!).

What's new in version 2.2?

Starting with version 2.2a, frenchb alters the layout of lists, footnotes, and the indentation of first paragraphs of sections) *only if* French is the "main language" (i.e. babel's last language option). The layout is global for the whole document: lists, etc. look the same in French and in other languages, everything is typeset "à la française" if French is the "main language", otherwise frenchb doesn't change anything regarding lists, footnotes, and indentation of paragraphs.

What's new in version 2.1?

A new command `\fup` is provided to typeset better looking superscripts; it was designed using ideas from Jacques André, Thierry Bouche and René Fritz, thanks to all of them! Former command `\up` is now defined as `\fup`, an option `FrenchSuperscripts=false` is provided for backward compatibility.

What's new in version 2.0?

Here is the list of all changes:

- Support for LaTeX-2.09 and for LaTeX 2_ε in compatibility mode has been dropped. This version is meant for LaTeX 2_ε and Plain based formats (like `bplain`). LaTeX 2_ε formats based on mlTeX are no longer supported either (plenty of good 8-bits fonts are available now, so T1 encoding should be preferred for typesetting in French). A warning is issued when OT1 encoding is in use at the `\begin{document}`.
- Customisation should now be handled only by command `\frenchbsetup{}`, `frenchb.cfg` (kept for compatibility) should no longer be used. See section 1.2 for the list of available options.
- Captions in figures and tables have changed in French: former abbreviations "Fig." and "Tab." have been replaced by full names "Figure" and "Table". If this leads to formatting problems in captions, you can add the following two commands to your preamble (after loading babel) to get the

former captions
`\addto\captionsfrench{\def\figurename{{\scshape Fig.}}}`
`\addto\captionsfrench{\def\tablename{{\scshape Tab.}}}`.

- The `\nombre` command is now provided by the `numprint` package which has to be loaded *after* `babel` with the option `autolanguage` if number formatting should depend on the current language.

- The `\bsc` command no longer uses an `\hbox` to stop hyphenation of names but a `\kern0pt` instead. This change enables `microtype` to fine tune the length of the argument of `\bsc`; as a side-effect, compound names like Dupont-Durand can now be hyphenated on explicit hyphens. You can get back to the former behaviour of `\bsc` by adding
`\renewcommand*{\bsc}[1]{\leavevmode\hbox{\scshape #1}}`
to the preamble of your document.

- Footnotes are now displayed “à la française” for the whole document, except with an explicit
`\frenchbsetup{AutoSpaceFootnotes=false,FrenchFootnotes=false}`.
Add this command if you want standard footnotes. It is still possible to revert locally to the standard layout of footnotes by adding `\StandardFootnotes` (inside a `minipage` environment for instance).

2 The code

2.1 Initial setup

First make sure that `\l@french` is defined (possibly as 0). In Plain formats, `@` catcode is not 'letter'; LuaTeX and XeTeX formats set `\lang@<languagename>` instead of `\l@<languagename>`.

```
\chardef\atcatcode=\catcode`\@
\catcode`\@=11\relax
\def\FB@nopatterns{%
   \ifx\l@nohyphenation\@undefined
      \edef\bbl@nulllanguage{\string\language=0}%
      \adddialect\l@french0
   \else
      \adddialect\l@french\l@nohyphenation
      \edef\bbl@nulllanguage{\string\language=nohyphenation}%
   \fi
   \@nopatterns{French}}
\ifx\l@french\@undefined
   \ifx\lang@french\@undefined
      \FB@nopatterns
   \else
      \ifnum\lang@french>0
         \let\l@french\lang@french
      \else
         \FB@nopatterns{French}
      \fi
   \fi
\fi
```

If `frenchb.ldf` was loaded with babel's options francais or frenchb, we make it behave as if french was specified.

```
\def\bbl@tempa{francais}
\ifx\CurrentOption\bbl@tempa
  \let\l@francais\l@french
  \def\captionsfrancais{\captionsfrench}
  \def\datefrancais{\datefrench}
  \def\extrasfrancais{\extrasfrench}
  \def\noextrasfrancais{\extrasfrench}
  \def\CurrentOption{french}
\fi
\def\bbl@tempa{frenchb}
\ifx\CurrentOption\bbl@tempa
  \let\l@frenchb\l@french
  \def\captionsfrenchb{\captionsfrench}
  \def\datefrenchb{\datefrench}
  \def\extrasfrenchb{\extrasfrench}
  \def\noextrasfrenchb{\extrasfrench}
  \def\CurrentOption{french}
\fi
\catcode`\@=\atcatcode \let\atcatcode\relax
```

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

```
\LdfInit\CurrentOption\captionsfrench
```

Quit if babel's version is less than 3.9.

```
\let\bbl@tempa\relax
\ifx\StartBabelCommands\@undefined
   \let\bbl@tempa\endinput
   \PackageError{frenchb.ldf}
        {frenchb requires babel v.3.9.\MessageBreak
         Aborting here}
        {Please upgrade Babel!}
\fi
\bbl@tempa
```

`\ifLaTeXe` No support is provided for late LaTeX-2.09: issue a warning and exit if LaTeX-2.09 is in use. Plain is still supported.

```
\newif\ifLaTeXe
\let\bbl@tempa\relax
\ifx\magnification\@undefined
   \ifx\@compatibilitytrue\@undefined
     \PackageError{frenchb.ldf}
        {LaTeX-2.09 format is no longer supported.\MessageBreak
         Aborting here}
        {Please upgrade to LaTeX2e!}
     \let\bbl@tempa\endinput
   \else
     \LaTeXetrue
   \fi
\fi
\bbl@tempa
```

`frenchb.ldf` can be loaded with options canadien or acadian, which both stand for Canadian French. Internally, acadian will be the name of the corresponding babel's dialect, so we set `\CurrentOption` to acadian in both cases. If no specific hyphenation patterns are available, Canadian French will use the French ones.
TODO: Canadien French hyphenation doesn't work with LuaTeX.

```
\ifx\l@acadian\@undefined
   \ifx\l@canadien\@undefined
      \adddialect\l@acadian\l@french
      \adddialect\l@canadien\l@french
   \else
      \adddialect\l@acadian\l@canadien
   \fi
\else
  \adddialect\l@canadien\l@acadian
\fi
\def\bbl@tempa{canadien}
\ifx\CurrentOption\bbl@tempa
  \def\captionscanadien{\captionsacadian}
```

```
  \def\datecanadien{\dateacadian}
  \def\extrascanadien{\extrasacadian}
  \def\noextrascanadien{\extrasacadian}
  \def\CurrentOption{acadian}
\fi
```

Let's provide a substitute for `\PackageWarning` and `\PackageInfo` not defined in Plain:

```
\def\fb@warning#1{%
   \begingroup
      \newlinechar=`\^^J
      \def\\{^^J(frenchb.ldf) }%
      \message{\\#1}%
   \endgroup}
\def\fb@info#1{%
   \begingroup
      \newlinechar=`\^^J
      \def\\{^^J}%
      \wlog{#1}%
   \endgroup}
```

French uses the standard values of `\lefthyphenmin` (2) and `\righthyphenmin` (3); let's provide their values though, as required by babel.

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

`\ifFBunicode` `\ifFBLuaTeX` `\ifFBXeTeX`

French hyphenation patterns are now coded in Unicode, see file `hyph-fr.tex`. XeTeX and LuaTeX engines require some extra code to deal with the French "apostrophe". Let's define three new 'if': `\ifFBLuaTeX`, `\ifFBXeTeX` and `\ifFBunicode` which will be true for XeTeX and LuaTeX engines and false for 8-bits engines.
We cannot rely on ε-TEX's `\ifdefined` at this stage, as it is not defined in Plain TEX format.

```
\newif\ifFBunicode
\newif\ifFBLuaTeX
\newif\ifFBXeTeX
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname luatexversion\endcsname\relax
\else
  \FBunicodetrue \FBLuaTeXtrue
\fi
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname XeTeXrevision\endcsname\relax
\else
  \FBunicodetrue \FBXeTeXtrue
\fi
```

`\extrasfrench` `\noextrasfrench`

The macro `\extrasfrench` will perform all the extra definitions needed for the French language. The macro `\noextrasfrench` is used to cancel the actions of `\extrasfrench`.
In French, character "apostrophe" is a letter in expressions like `l'ambulance` (French hyphenation patterns provide entries for this kind of words). This means

that the \lccode of “apostrophe” has to be non null in French for proper hyphenation of those expressions, and has to be reset to null when exiting French.
The following code ensures correct hyphenation of words like d’aventure, l’utopie, with all TeX engines (XeTeX, LuaTeX, pdfTeX) using hyph-fr.tex patterns.

```
\@namedef{extras\CurrentOption}{%
    \babel@savevariable{\lccode‘\’}%
    \ifFBunicode
       \babel@savevariable{\lccode"2019}%
       \lccode‘\’="2019\lccode"2019="2019
    \else
       \lccode‘\’=‘\’
    \fi
}
\@namedef{noextras\CurrentOption}{}
```

Let’s define a handy command for adding stuff to \extras\CurrentOption, \noextras\CurrentOption or \captions\CurrentOption but first let’s save the value of \CurrentOption for later use in \frenchbsetup{} (‘AfterEndOfPackage’, \CurrentOption will be lost).

```
\let\FB@CurOpt\CurrentOption
\newcommand*{\FB@addto}[2]{%
    \expandafter\addto\csname #1\FB@CurOpt\endcsname{#2}}
```

One more thing \extrasfrench needs to do is to make sure that “Frenchspacing” is in effect. \noextrasfrench will switch “Frenchspacing” off again if necessary.

```
\FB@addto{extras}{\bbl@frenchspacing}
\FB@addto{noextras}{\bbl@nonfrenchspacing}
```

2.2 Punctuation

As long as no better solution is available, the ‘high punctuation’ characters (; ! ? and :) have to be made \active for an automatic control of the amount of space to be inserted before them. Both XeTeX and LuaTeX provide an alternative to active characters (‘XeTeXinterchar’ mechanism and LuaTeX’s callbacks).
With LuaTeX and XeTeX engines, frenchb handles French quotes together with ‘high punctuation’, a new conditional will be needed:

```
\newif\ifFBAutoSpaceGuill  \FBAutoSpaceGuilltrue
```

\ifFB@active@punct
\ifFB@xetex@punct
\ifFB@luatex@punct

Three internal flags are needed for the three different techniques used for ‘high punctuation’ management.
With LuaTeX, starting with version 0.76, callbacks are used to get rid of active punctuation. With previous versions, ‘high punctuation’ characters remain active (see below).

```
\newif\ifFB@active@punct  \FB@active@puncttrue
\newif\ifFB@luatex@punct
\ifFBLuaTeX
  \ifnum\luatexversion>75
    \FB@luatex@puncttrue\FB@active@punctfalse
```

```
  \fi
\fi
```

For XeTeX, the availability of `\XeTeXinterchartokenstate` decides whether the 'high punctuation' characters (`;` `!` `?` and `:`) have to be made `\active` or not.

```
\newif\ifFB@xetex@punct
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname XeTeXinterchartokenstate\endcsname\relax
\else
  \FB@xetex@puncttrue\FB@active@punctfalse
\fi
```

`\FBcolonspace` `\FBthinspace` `\FBcolonskip` `\FBthinskip`

According to the I.N. specifications, the ':' requires an inter-word space before it, the other three require just a `\thinspace`. We define `\FBcolonspace` as `\space` (inter-word space) and `\FBthinspace` as `\thinspace` (both are user customisable). LuaTeX requires skips instead of commands, so we define `\FBcolonskip` and `\FBthinskip` to hold the specifications (width/stretch/shrink) of `\space` and `\thinspace` for the `lmr10` font; these parameters will be scaled for the current font by the `frenchb.lua` script (see how p. 18). `\FBcolonskip` and `\FBthinskip` are also user customisable.

```
\newcommand*{\FBcolonspace}{\space}
\newcommand*{\FBthinspace}{\thinspace}
\newskip\FBcolonskip
\FBcolonskip=3.33pt plus 1.665pt minus 1.11pt \relax
\newskip\FBthinskip
\FBthinskip=1.66672pt \relax
```

2.2.1 Punctuation with LuaTeX

The following part holds specific code for punctuation with modern LuaTeX engines (version ≥ 0.76).
We define two LuaTeX attributes to control spacing in French for 'high punctuation' and quotes, making sure that `\newluatexattribute` is defined.

```
\ifFB@luatex@punct
  \begingroup\expandafter\expandafter\expandafter\endgroup
  \expandafter\ifx\csname newluatexattribute\endcsname\relax
    \input luaotfload.sty
  \fi
  \newluatexattribute\FB@addDPspace   \FB@addDPspace=1 \relax
  \newluatexattribute\FB@addGUILspace \FB@addGUILspace=0 \relax
  \ifLaTeXe
    \PackageInfo{frenchb.ldf}{No need for active punctuation characters%
                \MessageBreak with this version of LuaTeX!%
                \MessageBreak reported}
  \else
    \fb@info{No need for active punctuation characters\\
             with this version of LuaTeX!}
  \fi
```

`frenchb.lua` holds Lua code to deal with 'high punctuation' and quotes. This code is based on suggestions from Paul Isambert.

First we define two flags to control spacing before French ‘high punctuation’ (thin space or inter-word space).

```
local FB_punct_thin =
  {[string.byte("!")] = true,
   [string.byte("?")] = true,
   [string.byte(";")] = true}
local FB_punct_thick =
  {[string.byte(":")] = true}
```

Managing spacing after ‘«’ (U+00AB) and before ‘»’ (U+00BB) can be done by the way; we define two flags, `FB_punct_left` for characters requiring some space before them and `FB_punct_right` for ‘«’ which must be followed by some space.

```
local FB_punct_left =
  {[string.byte("!")] = true,
   [string.byte("?")] = true,
   [string.byte(";")] = true,
   [string.byte(":")] = true,
   [0xBB]             = true}
local FB_punct_right =
  {[0xAB]              = true}
```

Two more flags will be needed to avoid spurious spaces in strings like !! ?? or (?)

```
local FB_punct_null =
  {[string.byte("!")] = true,
   [string.byte("?")] = true,
   [string.byte("[")] = true,
   [string.byte("(")] = true,
```

or if the user has typed a nobreak space U+00A0 or a nobreak thin space U+202F before a ‘high punctuation’ character: no space should be added by frenchb. Same is true inside French quotes.

```
   [0xA0]             = true,
   [0x202F]           = true}
local FB_guil_null =
  {[0xA0]             = true,
   [0x202F]           = true}
```

Local definitions for nodes:

```
local new_node     = node.new
local copy_node    = node.copy
local node_id      = node.id
local GLUE         = node_id("glue")
local GSPEC        = node_id("glue_spec")
local GLYPH        = node_id("glyph")
local PENALTY      = node_id("penalty")
local nobreak      = new_node(PENALTY)
nobreak.penalty    = 10000
local insert_node_before = node.insert_before
local insert_node_after  = node.insert_after
local remove_node        = nodes.remove
```

Some variables to store `\FBthinskip`, `\FBcolonskip` and `\FBguillskip` (given for `lmr10`); width/stretch/shrink are stored as fractions of `\fontdimen2`, `\fontdimen3`

and \fontdimen4 of lmr10 font respectively...

```
local thin10 = tex.skip['FBthinskip']
local thinwd = thin10.width/65536/3.33
local thinst = thin10.stretch/65536/1.665
local thinsh = thin10.shrink/655.36/1.11
local coln10 = tex.skip['FBcolonskip']
local colnwd = coln10.width/65536/3.33
local colnst = coln10.stretch/65536/1.665
local colnsh = coln10.shrink/65536/1.11
local guil10 = tex.skip['FBguillskip']
local guilwd = guil10.width/65536/3.33
local guilst = guil10.stretch/65536/1.665
local guilsh = guil10.shrink/65536/1.11
```

and a function to scale them for the current font:

```
local font_table = {}
local function new_glue_scaled (fid,width,stretch,shrink)
  local fp = font_table[fid]
  if not fp then
     font_table[fid] = font.getfont(fid).parameters
     fp = font_table[fid]
  end
  local gl = new_node(GLUE,0)
  local gl_spec = new_node(GSPEC)
  gl_spec.width = width * fp.space
  gl_spec.stretch = stretch * fp.space_stretch
  gl_spec.shrink = shrink * fp.space_shrink
  gl.spec = gl_spec
  return gl
end
```

Let's catch LuaTeX attributes \FB@addDPspace and \FB@addGUILspace. Constant FR=lang.id(french) will be defined by command \activate@luatexpunct.

```
local addDPspace   = luatexbase.attributes['FB@addDPspace']
local addGUILspace = luatexbase.attributes['FB@addGUILspace']
local has_attribute = node.has_attribute
```

The following function will be added to pre_linebreak_filter and hpack_filter callbacks. It catches all nodes of type GLYPH in the list starting at head and checks the language attributes of the current glyph: nothing is done if the current language is not French and only specific punctuation characters (those for which FB_punct_left or FB_punct_rigt is true) need a special treatment. In French, local variables are defined to hold the properties of the current glyph (item) and of the previous one (prev) or the next one (next).

```
local function french_punctuation (head)
  for item in node.traverse_id(GLYPH, head) do
    local lang = item.lang
    local char = item.char
    local SIG  = has_attribute(item, addGUILspace)
    if lang == FR and FB_punct_left[char] then
       local fid = item.font
```

```
      local prev = item.prev
      local prev_id, prev_subtype, prev_char
      if prev then
         prev_id = prev.id
         prev_subtype = prev.subtype
         if prev_id == GLYPH then
            prev_char = prev.char
         end
      end
      local glue = prev_id == GLUE and prev_subtype == 0
```

For characters for which FB_punct_thin or FB_punct_thick is *true*, the amount of spacing to be typeset before them is controlled by \FBthinskip (thinwd, thinst, thinsh) or \FBcolonskip (colnwd, colnst, colnsh) respectively. Two options: if a space has been typed in before (turned to *glue* in the node list), we remove the *glue* and add a nobreak penalty and the required *glue*. Otherwise (auto option), the penalty and the required *glue* are inserted *only if* all three conditions are satisfied: a) attribute \FB@addDPspace is set, b) the previous character is not part of type FB_punct_null (this avoids spurious spaces in strings like (!) or ??), c) the punctuation mark does not start a paragraph.

```
      if FB_punct_thin[char] or FB_punct_thick[char] then
         local SBDP = has_attribute(item, addDPspace)
         local fbglue
         if FB_punct_thick[char] then
            fbglue = new_glue_scaled(fid,colnwd,colnst,colnsh)
         else
            fbglue = new_glue_scaled(fid,thinwd,thinst,thinsh)
         end
         local auto =
             SBDP and SBDP > 0 and
            (prev_char and not FB_punct_null[prev_char]) or
            (not prev_char and (prev_id ~= 0 or prev_subtype ~= 3))
         if glue or auto then
            if glue then
               head = remove_node(head,prev,true)
            end
            insert_node_before(head, item, copy_node(nobreak))
            insert_node_before(head, item, copy_node(fbglue))
         end
```

Let's consider '»' now (the only remaining glyph of FB_punct_left class): we just have to remove any *glue* possibly preceeding '»', then insert the nobreak penalty and the proper *glue* (controlled by \FBguillskip). This is done only if French quotes have been 'activated' by options og=«, fg=» in \frenchbsetup{} and can be denied locally with \NoAutoSpacing (this is controlled by the SIG flag).

```
      elseif SIG and SIG > 0 then
         if glue or (prev_char and not FB_guil_null[prev_char]) then
            if glue then
               head = remove_node(head,prev,true)
            end
            local fbglue = new_glue_scaled(fid,guilwd,guilst,guilsh)
```

```
            insert_node_before(head, item, copy_node(nobreak))
            insert_node_before(head, item, copy_node(fbglue))
         end
      end
   end
```

Similarly, for ‘«’ (unique member of the `FB_punct_right` class), we check the following node looking for *glue* or `FB_guil_null` character, in order to remove any *glue* possibly following it and to insert the proper *glue* and nobreak penalty in this order if necessary.

```
   if lang == FR and FB_punct_right[char] and SIG and SIG > 0 then
      local next = item.next
      local next_id, next_subtype, next_char
      if next then
         next_id = next.id
         next_subtype = next.subtype
         if next_id == GLYPH then
            next_char = next.char
         end
      end
      local glue = next_id == GLUE and next_subtype == 0
      if glue or (next_char and not FB_guil_null[next_char]) then
         if glue then
            head = remove_node(head,next,true)
         end
         local fid = item.font
         local fbglue = new_glue_scaled(fid,guilwd,guilst,guilsh)
         insert_node_after(head, item, copy_node(fbglue))
         insert_node_after(head, item, copy_node(nobreak))
      end
   end
  end
  return head
end
return french_punctuation
```

As a language tag is part of glyph nodes in LuaTeX, nothing needs to be added to `\extrasfrench` and `\noextrasfrench`; we will just redefine `\shorthandoff` and `\shorthandon` in French to issue a warning reminding the user that active characters are no longer used in French with recent LuaTeX engines.

```
\newcommand*{\FB@luatex@punct@french}{%
   \ifx\shorthandoffORI\@undefined
     \let\shorthandonORI\shorthandon
     \let\shorthandoffORI\shorthandoff
   \fi
   \def\shorthandoff##1{%
      \ifx\PackageWarning\@undefined
        \fb@warning{\noexpand\shorthandoff{;:!?} is helpless with
         LuaTeX,\\ use \noexpand\NoAutoSpacing
         *inside a group* instead.}%
      \else
```

```
          \PackageWarning{frenchb.ldf}{\protect\shorthandoff{;:!?} is
           helpless with LuaTeX,\MessageBreak use \protect\NoAutoSpacing
           \space *inside a group* instead;\MessageBreak reported}%
        \fi}%
    \def\shorthandon##1{}%
}
\newcommand*{\FB@luatex@punct@nonfrench}{%
    \ifx\shorthandoffORI\@undefined
    \else
      \let\shorthandon\shorthandonORI
      \let\shorthandoff\shorthandoffORI
    \fi
}
\FB@addto{extras}{\FB@luatex@punct@french}
\FB@addto{noextras}{\FB@luatex@punct@nonfrench}
```

In LaTeX 2ε, file `frenchb.lua` will be loaded 'AtBeginDocument' *after* processing options (`ThinColonSpace` needs to be taken into account). The next definition will be used to activate Lua punctuation: it sets the language number for French, loads `frenchb.lua` and adds function `french_punctuation` to both callbacks `pre_linebreak_filter` (paragraph building) and `hpack_filter` (`\hbox` building).

```
\def\activate@luatexpunct{%
    \directlua{%
      FR = \the\l@french
      local f = dofile("frenchb.lua")
      luatexbase.add_to_callback("pre_linebreak_filter",
                 f, "frenchb.french_punctuation",1)
      luatexbase.add_to_callback("hpack_filter",
                 f, "frenchb.french_punctuation",1)
      }%
  }
```

End of specific code for punctuation with LuaTeX engines.

```
\fi
```

2.2.2 Punctuation with XeTeX

If `\XeTeXinterchartokenstate` is available, we use the "inter char" mechanism to provide correct spacing in French before the four characters `;` `!` `?` and `:`. The basis of the following code was borrowed from the `polyglossia` package, see `gloss-french.ldf`. We use the same mechanism for French quotes (« and »), when automatic spacing for quotes is required by options `og=` and `fg=` in `\frenchbsetup{}` (see section 2.9).

For every character used in French text-mode (except spaces), `\XeTeXcharclass` value must be 0. `\XeTeXcharclass` value for spaces is assumed to be 255. Otherwise, the spacing before the 'high punctuation' characters and inside quotes might not be correct.

We switch `\XeTeXinterchartokenstate` to 1 and change the `\XeTeXcharclass` values of `;` `!` `?` `:` `(` `]` « and » when entering French. Special care is taken to restore

them to their inital values when leaving French.
The following part holds specific code for punctuation with XeTeX engines.

```
\newcount\FB@interchartokenstateORI
\ifFB@xetex@punct
  \ifLaTeXe
   \PackageInfo{frenchb.ldf}{No need for active punctuation characters%
                  \MessageBreak with this version of XeTeX!%
                  \MessageBreak reported}
  \else
   \fb@info{No need for active punctuation characters\\
            with this version of XeTeX!}
  \fi
```

The following code is borrowed from `ltcntrl.dtx` (LaTeX base) for loops (`\@for` undefined in Plain):

```
  \ifx\@for\@undefined
    \def\@nnil{\@nil}%
    \def\@empty{}%
    \def\@fornoop#1\@@#2#3{}%
    \long\def\@for#1:=#2\do#3{%
      \expandafter\def\expandafter\@fortmp\expandafter{#2}%
      \ifx\@fortmp\@empty \else
        \expandafter\@forloop#2,\@nil,\@nil\@@#1{#3}\fi}%
    \long\def\@forloop#1,#2,#3\@@#4#5{\def#4{#1}\ifx #4\@nnil \else
      #5\def#4{#2}\ifx #4\@nnil \else#5\@iforloop #3\@@#4{#5}\fi\fi}%
    \long\def\@iforloop#1,#2\@@#3#4{\def#3{#1}\ifx #3\@nnil
          \expandafter\@fornoop \else
      #4\relax\expandafter\@iforloop\fi#2\@@#3{#4}}%
    \def\@tfor#1:={\@tf@r#1 }%
    \long\def\@tf@r#1#2\do#3{\def\@fortmp{#2}\ifx\@fortmp\space\else
        \@tforloop#2\@nil\@nil\@@#1{#3}\fi}%
    \long\def\@tforloop#1#2\@@#3#4{\def#3{#1}\ifx #3\@nnil
          \expandafter\@fornoop \else
         #4\relax\expandafter\@tforloop\fi#2\@@#3{#4}}%
  \fi
```

Six new character classes are defined for frenchb.

```
  \newXeTeXintercharclass\FB@punctthick
  \newXeTeXintercharclass\FB@punctthin
  \newXeTeXintercharclass\FB@punctnul
  \newXeTeXintercharclass\FB@guilo
  \newXeTeXintercharclass\FB@guilf
  \newXeTeXintercharclass\FB@guilnul
```

We define a command to store the `\XeTeXcharclass` values which will be modified for French (as a comma separated list) and a command to retrieve them.

```
  \def\FB@charclassesORI{}
  \def\empty{}
  \def\FB@parse#1,#2\endparse{\def\FB@class{#1}%
       \def\FB@charclassesORI{#2}}%
```

`\FB@xetex@punct@french` The following command will be executed when entering French, it first saves the

values to be modified, then fits them to our needs. It also redefines `\shorthandoff` and `\shorthandon` (locally) to avoid error messages with XeTeX-based engines.

```
\newcommand*{\FB@xetex@punct@french}{%
```

Saving must not be repeated if saved values are already in.

```
  \ifx\FB@charclassesORI\empty
    \FB@interchartokenstateORI=\XeTeXinterchartokenstate
    \@for\FB@char:={‘\:,‘\;,‘\!,‘\?,"AB,"BB,%
                    ‘\(,‘\[,‘\{,‘\,,‘\.,‘\-,‘\),‘\],‘\},%
                    ‘\%,"22,"27,"60,"2019,"A0,"202F}\do
        {\edef\FB@charclassesORI{\FB@charclassesORI%
                                 \the\XeTeXcharclass\FB@char,}}%
    \let\shorthandonORI\shorthandon
    \let\shorthandoffORI\shorthandoff
  \fi
```

Set the classes and interactions between classes.

```
  \XeTeXinterchartokenstate=1
  \XeTeXcharclass ‘\: = \FB@punctthick
  \XeTeXinterchartoks \z@ \FB@punctthick = {%
                  \ifhmode\FDP@colonspace\fi}%
  \XeTeXinterchartoks \FB@guilf \FB@punctthick = {%
                  \FDP@colonspace}%
  \XeTeXinterchartoks 255 \FB@punctthick = {%
                  \ifhmode\unskip\penalty\@M\FBcolonspace\fi}%
  \@for\FB@char:={‘\;,‘\!,‘\?}\do
       {\XeTeXcharclass\FB@char=\FB@punctthin}%
  \XeTeXinterchartoks \z@ \FB@punctthin = {%
                  \ifhmode\FDP@thinspace\fi}%
  \XeTeXinterchartoks \FB@guilf \FB@punctthin = {%
                  \FDP@thinspace}%
  \XeTeXinterchartoks 255 \FB@punctthin = {%
                  \ifhmode\unskip\penalty\@M\FBthinspace\fi}%
  \XeTeXinterchartoks \FB@guilo \z@ = {%
             \ifFBAutoSpaceGuill\FBguill@spacing\fi}%
  \XeTeXinterchartoks \FB@guilo 255 = {%
             \ifFBAutoSpaceGuill\FBguill@spacing\ignorespaces\fi}%
  \XeTeXinterchartoks \z@ \FB@guilf = {%
             \ifFBAutoSpaceGuill\FBguill@spacing\fi}%
  \XeTeXinterchartoks \FB@punctthin \FB@guilf = {%
             \ifFBAutoSpaceGuill\FBguill@spacing\fi}%
  \XeTeXinterchartoks 255 \FB@guilf = {%
             \ifFBAutoSpaceGuill\unskip\FBguill@spacing\fi}%
```

This will avoid spurious spaces in (!), [?] and with Unicode nobreakspaces (U+00A0, U+202F):

```
  \@for\FB@char:={‘\[,‘\(,"A0,"202F}\do
       {\XeTeXcharclass\FB@char=\FB@punctnul}%
```

These characters have their class changed by `xeCJK.sty`, let’s reset them to 0 in French.

```
  \@for\FB@char:={‘\{,‘\,,‘\.,‘\-,‘\),‘\],‘\},‘\%,%
```

```
                    "22,"27,"60,"2019}\do
          {\XeTeXcharclass\FB@char=\z@}%
```

With Xe(La)TeX, French defines no active shorthands.

```
    \def\shorthandoff##1{%
      \ifx\PackageWarning\@undefined
        \fb@warning{\noexpand\shorthandoff{;:!?} is helpless with
         XeTeX,\\ use \noexpand\NoAutoSpacing
         *inside a group* instead.}%
      \else
        \PackageWarning{frenchb.ldf}{\protect\shorthandoff{;:!?} is
         helpless with XeTeX,\MessageBreak use \protect\NoAutoSpacing
         \space *inside a group* instead;\MessageBreak reported}%
      \fi}%
    \def\shorthandon##1{}%
  }
```

`\FB@xetex@punct@nonfrench` The following command will be executed when leaving French for restoring classes and commands modified in French. When French is not the main language, `\noextrasfrench` is executed 'AtBeginDocument', so the test on `\FB@charclassesORI` is mandatory.

```
  \newcommand*{\FB@xetex@punct@nonfrench}{%
    \ifx\FB@charclassesORI\empty
    \else
      \@for\FB@char:={‘\:,‘\;,‘\!,‘\?,"AB,"BB,%
                      ‘\(,‘\[,‘\{,‘\,,‘\.,‘\-,‘\),‘\],‘\},%
                      ‘\%,"22,"27,"60,"2019,"A0,"202F}\do
          {\expandafter\FB@parse\FB@charclassesORI\endparse
           \XeTeXcharclass\FB@char=\FB@class}%
      \def\FB@charclassesORI{}%
      \XeTeXinterchartokenstate=\FB@interchartokenstateORI
      \let\shorthandon\shorthandonORI
      \let\shorthandoff\shorthandoffORI
    \fi
  }
  \FB@addto{extras}{\FB@xetex@punct@french}
  \FB@addto{noextras}{\FB@xetex@punct@nonfrench}
```

End of specific code for punctuation with modern XeTeX engines.

```
\fi
```

2.2.3 Punctuation with standard (pdf)TeX

In standard (pdf)TeX we need to make the four characters `;` `!` `?` and `:` 'active' and provide their definitions.

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

We first tune the amount of space before ; ! ? and :. This should only happen in horizontal mode, hence the test \ifhmode.
In horizontal mode, if a space has been typed before ';' we remove it and put an unbreakable \FBthinspace instead. If no space has been typed, we add \FDP@thinspace which will be defined, up to the user's wishes, as \FBthinspace, or as \@empty.

```
\declare@shorthand{french}{;}{%
    \ifhmode
    \ifdim\lastskip>\z@
        \unskip\penalty\@M\FBthinspace
        \else
          \FDP@thinspace
      \fi
    \fi
```

Now we can insert a ; character.

```
    \string;}
```

The next three definitions are very similar.

```
\declare@shorthand{french}{!}{%
    \ifhmode
      \ifdim\lastskip>\z@
        \unskip\penalty\@M\FBthinspace
      \else
        \FDP@thinspace
      \fi
    \fi
    \string!}
\declare@shorthand{french}{?}{%
    \ifhmode
      \ifdim\lastskip>\z@
        \unskip\penalty\@M\FBthinspace
      \else
        \FDP@thinspace
      \fi
    \fi
    \string?}
\declare@shorthand{french}{:}{%
    \ifhmode
      \ifdim\lastskip>\z@
        \unskip\penalty\@M\FBcolonspace
      \else
        \FDP@colonspace
      \fi
    \fi
    \string:}
```

When the active characters appear in an environment where their French behaviour is not wanted they should give an 'expected' result. Therefore we define shorthands at system level as well.

```
\declare@shorthand{system}{:}{\string:}
```

```
  \declare@shorthand{system}{!}{\string!}
  \declare@shorthand{system}{?}{\string?}
  \declare@shorthand{system}{;}{\string;}
%}
```

We specify that the French group of shorthands should be used when switching to French.

```
  \FB@addto{extras}{\languageshorthands{french}%
```

These characters are 'turned on' once, later their definition may vary. Don't misunderstand the following code: they keep being active all along the document, even when leaving French.

```
    \bbl@activate{:}\bbl@activate{;}%
    \bbl@activate{!}\bbl@activate{?}%
  }
  \FB@addto{noextras}{%
    \bbl@deactivate{:}\bbl@deactivate{;}%
    \bbl@deactivate{!}\bbl@deactivate{?}%
  }
\fi
```

2.2.4 Punctuation switches common to all engines

A new 'if' `\ifFBAutoSpacePunctuation` needs to be defined now to control the two possible ways of dealing with 'high punctuation'. it's default value is true, but it can be set to false by `\frenchbsetup{AutoSpacePunctuation=false}` for finer control.

```
\newif\ifFBAutoSpacePunctuation  \FBAutoSpacePunctuationtrue
```

`\AutoSpaceBeforeFDP` `\NoAutoSpaceBeforeFDP`

`\autospace@beforeFDP` and `\noautospace@beforeFDP` are internal commands. `\autospace@beforeFDP` defines `\FDP@thinspace` and `\FDP@colonspace` as unbreakable spaces and sets LuaTeX attribute `\FB@addDPspace` to 1 (true), while `\noautospace@beforeFDP` lets these spaces empty and sets flag `\FB@addDPspace` to 0 (false). User commands `\AutoSpaceBeforeFDP` and `\NoAutoSpaceBeforeFDP` do the same and take care of the flag `\ifFBAutoSpacePunctuation` in LaTeX. Set the default now for Plain (done later for LaTeX).

```
\def\autospace@beforeFDP{%
        \ifFB@luatex@punct\FB@addDPspace=1 \fi
        \def\FDP@thinspace{\penalty\@M\FBthinspace}%
        \def\FDP@colonspace{\penalty\@M\FBcolonspace}}
\def\noautospace@beforeFDP{%
        \ifFB@luatex@punct\FB@addDPspace=0 \fi
        \let\FDP@thinspace\@empty
        \let\FDP@colonspace\@empty}
\ifLaTeXe
    \def\AutoSpaceBeforeFDP{\autospace@beforeFDP
                            \FBAutoSpacePunctuationtrue}
    \def\NoAutoSpaceBeforeFDP{\noautospace@beforeFDP
                              \FBAutoSpacePunctuationfalse}
\else
    \let\AutoSpaceBeforeFDP\autospace@beforeFDP
```

```
    \let\NoAutoSpaceBeforeFDP\noautospace@beforeFDP
  \fi
\AutoSpaceBeforeFDP
```

In LaTeX 2ε `\ttfamily` (and hence `\texttt`) will be redefined 'AtBeginDocument' as `\ttfamilyFB` so that no space is added before the four `; : ! ?` characters, even if `AutoSpacePunctuation` is `true`. `\rmfamily` and `\sffamily` need to be redefined also (`\ttfamily` is not always used inside a group, its effect can be cancelled by `\rmfamily` or `\sffamily`).
These redefinitions can be canceled if necessary, for instance to recompile older documents, see option `OriginalTypewriter` below.
To be consistent with what is done for the `; : ! ?` characters, `\ttfamilyFB` also switches off insertion of spaces inside French guillemets *when they are typed in as characters* with the 'og'/'fg' options in `\frenchbsetup{}`. This is also a workaround for the weird behaviour of these characters in verbatim mode.

```
\ifLaTeXe
    %\let\ttfamilyORI\ttfamily
    %\let\rmfamilyORI\rmfamily
    %\let\sffamilyORI\sffamily
    \DeclareRobustCommand\ttfamilyFB{%
         \FBAutoSpaceGuillfalse
         \ifFB@luatex@punct\FB@addGUILspace=0 \fi
         \noautospace@beforeFDP\ttfamilyORI}%
    \DeclareRobustCommand\rmfamilyFB{%
         \FBAutoSpaceGuilltrue
         \ifFB@luatex@punct\FB@addGUILspace=1 \fi
         \ifFBAutoSpacePunctuation
           \autospace@beforeFDP
         \else
           \noautospace@beforeFDP
         \fi
         \rmfamilyORI}%
    \DeclareRobustCommand\sffamilyFB{%
         \FBAutoSpaceGuilltrue
         \ifFB@luatex@punct\FB@addGUILspace=1 \fi
         \ifFBAutoSpacePunctuation
           \autospace@beforeFDP
         \else
           \noautospace@beforeFDP
         \fi
         \sffamilyORI}%
\fi
```

`\NoAutoSpacing` The following command will switch off active punctuation characters (if any) and disable automatic spacing for French quote characters. It is engine independent (works for TeX, LuaTeX and XeTeX based engines) and is meant to be used inside a group.

```
\newcommand*{\NoAutoSpacing}{\FBAutoSpaceGuillfalse
   \ifFB@active@punct\shorthandoff{;:!?}\fi
```

```
    \ifFB@xetex@punct\XeTeXinterchartokenstate=0 \fi
    \ifFB@luatex@punct\FB@addDPspace=0 \FB@addGUILspace=0 \fi
}
```

2.3 Commands for French quotation marks

\og \fg

The top macros for quotation marks will be called `\og` (“ouvrez guillemets”) and `\fg` (“fermez guillemets”). Another option for typesetting quotes in multilingual texts is to use the package `csquotes` and its command `\enquote`. Dummy definition of `\og` and `\fg` just to ensure that this commands are not yet defined. The default definition of `\og` and `\fg` will be set later (for English) by `\bbl@nonfrenchguillemets`.

```
\newcommand*{\og}{\@empty}
\newcommand*{\fg}{\@empty}
```

\guillemotleft \guillemotright \textquoteddblleft \textquoteddblright

LaTeX users are supposed to use 8-bit output encodings (T1, LY1,...) to typeset French, those who still stick to OT1 should call `aeguill` or a similar package. In both cases the commands `\guillemotleft` and `\guillemotright` will print the French opening and closing quote characters from the output font. For XeLaTeX and LuaLaTeX, `\guillemotleft` and `\guillemotright` are defined by package `xunicode` loaded by `fontspec`.

We provide the following definitions for non-LaTeX users only as fall-back, they are welcome to change them for anything better.

```
\ifLaTeXe
\else
  \ifFBunicode
     \def\guillemotleft{{\char"00AB}}
     \def\guillemotright{{\char"00BB}}
     \def\textquotedblleft{{\char"201C}}
     \def\textquotedblright{{\char"201D}}
  \else
    \def\guillemotleft{\leavevmode\raise0.25ex
                       \hbox{$\scriptscriptstyle\ll$}}
    \def\guillemotright{\raise0.25ex
                        \hbox{$\scriptscriptstyle\gg$}}
    \def\textquotedblleft{``}
    \def\textquotedblright{''}
  \fi
  \let\xspace\relax
\fi
```

The next step is to provide correct spacing after `\guillemotleft` and before `\guillemotright`: a space precedes and follows quotation marks but no line break is allowed neither *after* the opening one, nor *before* the closing one. `\FBguill@spacing` which does the spacing, has been fine tuned by Thierry Bouche to 80% of an inter-word space but with reduced stretchability. French quotes (including spacing) are printed by `\FB@og` and `\FB@fg`, the expansion of the top level commands `\og` and `\og` is different in and outside French. We’ll try to be smart to users of David Carlisle’s `xspace` package: if this package is loaded there

will be no need for {} or \ to get a space after \fg, otherwise \xspace will be defined as \relax (done at the end of this file).
LuaTeX which requires skips; \FBguillskip is computed from \FBguill@spacing for the lmr10 font, its dimensions will be scaled by frenchb.lua for the current font.

```
\newskip\FBguillskip
\FBguillskip=2.664pt plus 0.500pt minus 0.888pt \relax
\newcommand*{\FBguill@spacing}{\penalty\@M\hskip.8\fontdimen2\font
                                            plus.3\fontdimen3\font
                                           minus.8\fontdimen4\font}
```

\FBguill@spacing is not used with LuaTeX.

```
\ifFB@luatex@punct
  \DeclareRobustCommand*{\FB@og}{\leavevmode
          \bgroup\FB@addGUILspace=1 \guillemotleft\egroup}
  \DeclareRobustCommand*{\FB@fg}{\ifdim\lastskip>\z@\unskip\fi
          \bgroup\FB@addGUILspace=1 \guillemotright\egroup\xspace}
\fi
```

With XeTeX, \FBAutoSpaceGuill is set to false locally to prevent the quotes characters from adding space when option og=«, fg=» is set. characters.

```
\ifFB@xetex@punct
  \DeclareRobustCommand*{\FB@og}{\leavevmode
        \bgroup\FBAutoSpaceGuillfalse\guillemotleft\egroup
        \FBguill@spacing}
  \DeclareRobustCommand*{\FB@fg}{\ifdim\lastskip>\z@\unskip\fi
        \FBguill@spacing
        \bgroup\FBAutoSpaceGuillfalse\guillemotright\egroup\xspace}
\fi
\ifFB@active@punct
  \DeclareRobustCommand*{\FB@og}{\leavevmode
        \guillemotleft
        \FBguill@spacing}
  \DeclareRobustCommand*{\FB@fg}{\ifdim\lastskip>\z@\unskip\fi
        \FBguill@spacing
        \guillemotright\xspace}
\fi
```

The top level definitions for French quotation marks are switched on and off through the \extrasfrench \noextrasfrench mechanism. Outside French, \og and \fg will typeset standard English opening and closing double quotes.

```
\ifLaTeXe
  \def\bbl@frenchguillemets{\renewcommand*{\og}{\FB@og}%
                            \renewcommand*{\fg}{\FB@fg}}
  \renewcommand*{\og}{\textquotedblleft}
  \renewcommand*{\fg}{\ifdim\lastskip>\z@\unskip\fi \textquotedblright}
\else
  \def\bbl@frenchguillemets{\let\og\FB@og
                            \let\fg\FB@fg}
  \def\og{\textquotedblleft}
  \def\fg{\ifdim\lastskip>\z@\unskip\fi\textquotedblright}
```

```
\fi
\FB@addto{extras}{\babel@save\og \babel@save\fg \bbl@frenchguillemets}
```

2.4 Date in French

\datefrench The macro \datefrench redefines the command \today to produce French dates. This new implementation requires babel 3.9 or newer.

```
\ifLaTeXe
  \def\BabelLanguages{french,acadian}
  \StartBabelCommands*{\BabelLanguages}{date}
      [unicode, fontenc=EU1 EU2, charset=utf8]
    \SetString\monthiiname{février}
    \SetString\monthviiiname{août}
    \SetString\monthxiiname{décembre}
  \StartBabelCommands*{\BabelLanguages}{date}
    \SetStringLoop{month#1name}{%
        janvier,f\'evrier,mars,avril,mai,juin,juillet,%
        ao\^ut,septembre,octobre,novembre,d\'ecembre}
    \SetString\today{{\number\day}\ifnum1=\day {\ier}\fi\space
        \csname month\romannumeral\month name\endcsname \space
        \number\year
       }
  \EndBabelCommands
\else
  \ifFBunicode
    \@namedef{date\CurrentOption}{%
      \def\today{{\number\day}\ifnum1=\day {\ier}\fi \space
          \ifcase\month
            \or janvier\or février\or mars\or avril\or mai\or
            juin\or juillet\or août\or septembre\or
            octobre\or novembre\or décembre\fi
          \space \number\year}}
  \else
    \@namedef{date\CurrentOption}{%
      \def\today{{\number\day}\ifnum1=\day {\ier}\fi \space
        \ifcase\month
          \or janvier\or f\'evrier\or mars\or avril\or mai\or
          juin\or juillet\or ao\^ut\or septembre\or
          octobre\or novembre\or d\'ecembre\fi
        \space \number\year}}
  \fi
\fi
```

2.5 Extra utilities

Let's provide the French user with some extra utilities.

\up \fup \up eases the typesetting of superscripts like '1er'. Up to version 2.0 of frenchb \up was just a shortcut for \textsuperscript in LaTeX 2_ε, but several users complained that \textsuperscript typesets superscripts too high and too big, so we

now define \fup as an attempt to produce better looking superscripts. \up is defined as \fup but \frenchbsetup{FrenchSuperscripts=false} redefines \up as \textsuperscript for compatibility with previous versions.
When a font has built-in superscripts, the best thing to do is to just use them, otherwise \fup has to simulate superscripts by scaling and raising ordinary letters. Scaling is done using package scalefnt which will be loaded at the end of babel's loading (frenchb being an option of babel, it cannot load a package while being read).

```
\newif\ifFB@poorman
\newdimen\FB@Mht
\ifLaTeXe
  \AtEndOfPackage{\RequirePackage{scalefnt}}
```

\FB@up@fake holds the definition of fake superscripts. The scaling ratio is 0.65, raising is computed to put the top of lower case letters (like 'm') just under the top of upper case letters (like 'M'), precisely 12% down. The chosen settings look correct for most fonts, but can be tuned by the end-user if necessary by changing \FBsupR and \FBsupS commands.
\FB@lc is defined as \MakeLowercase to inhibit the uppercasing of superscripts (this may happen in page headers with the standard classes but is wrong); \FB@lc can be redefined to do nothing by option LowercaseSuperscripts=false of \frenchbsetup{}.

```
  \newcommand*{\FBsupR}{-0.12}
  \newcommand*{\FBsupS}{0.65}
  \newcommand*{\FB@lc}[1]{\MakeLowercase{#1}}
  \DeclareRobustCommand*{\FB@up@fake}[1]{%
    \settoheight{\FB@Mht}{M}%
    \addtolength{\FB@Mht}{\FBsupR \FB@Mht}%
    \addtolength{\FB@Mht}{-\FBsupS ex}%
    \raisebox{\FB@Mht}{\scalefont{\FBsupS}{\FB@lc{#1}}}%
    }
```

The only packages I currently know to take advantage of real superscripts are a) realscripts used in conjunction with XeLaTeX or LuaLaTeX and OpenType fonts having the font feature 'VerticalPosition=Superior' and b) fourier (from version 1.6) when Expert Utopia fonts are available.
\FB@up checks whether the current font is a Type1 'Expert' (or 'Pro') font with real superscripts or not (the code works currently only with fourier-1.6 but could work with any Expert Type1 font with built-in superscripts, see below), and decides to use real or fake superscripts. It works as follows: the content of \f@family (family name of the current font) is split by \FB@split into two pieces, the first three characters ('fut' for Fourier, 'ppl' for Adobe's Palatino, ...) stored in \FB@firstthree and the rest stored in \FB@suffix which is expected to be 'x' or 'j' for expert fonts.

```
  \def\FB@split#1#2#3#4\@nil{\def\FB@firstthree{#1#2#3}%
                             \def\FB@suffix{#4}}
  \def\FB@x{x}
  \def\FB@j{j}
  \DeclareRobustCommand*{\FB@up}[1]{%
    \bgroup \FB@poormantrue
```

```
      \expandafter\FB@split\f@family\@nil
```

Then `\FB@up` looks for a `.fd` file named `t1fut-sup.fd` (Fourier) or `t1ppl-sup.fd` (Palatino), etc. supposed to define the subfamily (`fut-sup` or `ppl-sup`, etc.) giving access to the built-in superscripts. If the `.fd` file is not found by `\IfFileExists`, `\FB@up` falls back on fake superscripts, otherwise `\FB@suffix` is checked to decide whether to use fake or real superscripts.

```
      \edef\reserved@a{\lowercase{%
         \noexpand\IfFileExists{\f@encoding\FB@firstthree -sup.fd}}}%
      \reserved@a
        {\ifx\FB@suffix\FB@x \FB@poormanfalse\fi
         \ifx\FB@suffix\FB@j \FB@poormanfalse\fi
         \ifFB@poorman \FB@up@fake{#1}%
         \else         \FB@up@real{#1}%
         \fi}%
        {\FB@up@fake{#1}}%
    \egroup}
```

`\FB@up@real` just picks up the superscripts from the subfamily (and forces lowercase).

```
  \newcommand*{\FB@up@real}[1]{\bgroup
       \fontfamily{\FB@firstthree -sup}\selectfont \FB@lc{#1}\egroup}
```

`\fup` is defined as `\FB@up` unless `\realsuperscript` is defined by `realscripts.sty`.

```
  \DeclareRobustCommand*{\fup}[1]{%
    \ifx\realsuperscript\@undefined
       \FB@up{#1}%
    \else
      \bgroup\let\fakesuperscript\FB@up@fake
             \realsuperscript{\FB@lc{#1}}\egroup
    \fi}
```

Let's provide a temporary definition for `\up` (redefined 'AtBeginDocument' as `\fup` or `\textsuperscript` according to `\frenchbsetup{}` options).

```
  \providecommand*{\up}{\relax}
```

Poor man's definition of `\up` for Plain.

```
\else
  \providecommand*{\up}[1]{\leavevmode\raise1ex\hbox{\sevenrm #1}}
\fi
```

`\ieme` `\ier` `\iere` `\iemes` `\iers` `\ieres`

Some handy macros for those who don't know how to abbreviate ordinals:

```
\def\ieme{\up{\lowercase{e}}\xspace}
\def\iemes{\up{\lowercase{es}}\xspace}
\def\ier{\up{\lowercase{er}}\xspace}
\def\iers{\up{\lowercase{ers}}\xspace}
\def\iere{\up{\lowercase{re}}\xspace}
\def\ieres{\up{\lowercase{res}}\xspace}
```

`\No` `\no` `\Nos` `\nos` `\primo` `\fprimo)`

And some more macros relying on `\up` for numbering, first two support macros.

```
\newcommand*{\FrenchEnumerate}[1]{%
                      #1\up{\lowercase{o}}\kern+.3em}
```

```
\newcommand*{\FrenchPopularEnumerate}[1]{%
                        #1\up{\lowercase{o}})\kern+.3em}
```

Typing `\primo` should result in ‘1o ’,

```
\def\primo{\FrenchEnumerate1}
\def\secundo{\FrenchEnumerate2}
\def\tertio{\FrenchEnumerate3}
\def\quarto{\FrenchEnumerate4}
```

while typing `\fprimo)` gives ‘1o) .

```
\def\fprimo){\FrenchPopularEnumerate1}
\def\fsecundo){\FrenchPopularEnumerate2}
\def\ftertio){\FrenchPopularEnumerate3}
\def\fquarto){\FrenchPopularEnumerate4}
```

Let’s provide four macros for the common abbreviations of “Numéro”.

```
\DeclareRobustCommand*{\No}{N\up{\lowercase{o}}\kern+.2em}
\DeclareRobustCommand*{\no}{n\up{\lowercase{o}}\kern+.2em}
\DeclareRobustCommand*{\Nos}{N\up{\lowercase{os}}\kern+.2em}
\DeclareRobustCommand*{\nos}{n\up{\lowercase{os}}\kern+.2em}
```

`\bsc` As family names should be written in small capitals and never be hyphenated, we provide a command (its name comes from Boxed Small Caps) to input them easily. Note that this command has changed with version 2 of frenchb: a `\kern0pt` is used instead of `\hbox` because `\hbox` would break microtype’s font expansion; as a (positive?) side effect, composed names (such as Dupont-Durand) can now be hyphenated on explicit hyphens. Usage: `Jean~\bsc{Duchemin}`.

```
\DeclareRobustCommand*{\bsc}[1]{\leavevmode\begingroup\kern0pt
                                          \scshape #1\endgroup}
\ifLaTeXe\else\let\scshape\relax\fi
```

Some definitions for special characters. We won’t define `\tilde` as a Text Symbol not to conflict with the macro `\tilde` for math mode and use the name `\tild` instead. Note that `\boi` may *not* be used in math mode, its name in math mode is `\backslash`. `\degre` can be accessed by the command `\r{}` for ring accent.

```
\ifFBunicode
    \newcommand*{\at}{{\char"0040}}
    \newcommand*{\circonflexe}{{\char"005E}}
    \newcommand*{\tild}{{\char"007E}}
    \newcommand*{\boi}{\textbackslash}
    \newcommand*{\degre}{{\char"00B0}}
\else
  \ifLaTeXe
    \DeclareTextSymbol{\at}{T1}{64}
    \DeclareTextSymbol{\circonflexe}{T1}{94}
    \DeclareTextSymbol{\tild}{T1}{126}
    \DeclareTextSymbolDefault{\at}{T1}
    \DeclareTextSymbolDefault{\circonflexe}{T1}
    \DeclareTextSymbolDefault{\tild}{T1}
    \DeclareRobustCommand*{\boi}{\textbackslash}
    \DeclareRobustCommand*{\degre}{\r{}}
```

```
  \else
    \def\T@one{T1}
    \ifx\f@encoding\T@one
      \newcommand*{\degre}{{\char6}}
    \else
      \newcommand*{\degre}{{\char23}}
    \fi
    \newcommand*{\at}{{\char64}}
    \newcommand*{\circonflexe}{{\char94}}
    \newcommand*{\tild}{{\char126}}
    \newcommand*{\boi}{$\backslash$}
  \fi
\fi
```

`\degres` We now define a macro `\degres` for typesetting the abbreviation for 'degrees' (as in 'degrees Celsius'). As the bounding box of the character 'degree' has *very* different widths in CM/EC and PostScript fonts, we fix the width of the bounding box of `\degres` to 0.3 em, this lets the symbol 'degree' stick to the preceding (e.g., `45\degres`) or following character (e.g., `20~\degres C`).
If TEX Companion fonts are available (`textcomp.sty`), we pick up `\textdegree` from them instead of emulating 'degrees' from the `\r{}` accent. Otherwise we advise the user (once only) to use TS1-encoding.

```
\ifLaTeXe
  \newcommand*{\degres}{\degre}
  \ifFBunicode
    \DeclareRobustCommand*{\degres}{\degre}
  \else
    \def\Warning@degree@TSone{%
          \PackageWarning{frenchb.ldf}{%
             Degrees would look better in TS1-encoding:%
             \MessageBreak add \protect
             \usepackage{textcomp} to the preamble.%
             \MessageBreak Degrees used}}
    \AtBeginDocument{\ifx\DeclareEncodingSubset\@undefined
                          \DeclareRobustCommand*{\degres}{%
                             \leavevmode\hbox to 0.3em{\hss\degre\hss}%
                          \Warning@degree@TSone
                          \global\let\Warning@degree@TSone\relax}%
                      \else
                          \DeclareRobustCommand*{\degres}{%
                            \hbox{\UseTextSymbol{TS1}{\textdegree}}}%
                      \fi
                      }
  \fi
\else
  \newcommand*{\degres}{%
    \leavevmode\hbox to 0.3em{\hss\degre\hss}}
\fi
```

2.6 Formatting numbers

\DecimalMathComma
\StandardMathComma

As mentioned in the TEXbook p. 134, the comma is of type `\mathpunct` in math mode: it is automatically followed by a space. This is convenient in lists and intervals but unpleasant when the comma is used as a decimal separator in French: it has to be entered as `{,}`. `\DecimalMathComma` makes the comma be an ordinary character (of type `\mathord`) in French *only* (no space added); `\StandardMathComma` switches back to the standard behaviour of the comma.

```
\newcount\std@mcc
\newcount\dec@mcc
\std@mcc=\mathcode`\,
\dec@mcc=\std@mcc
\@tempcnta=\std@mcc
\divide\@tempcnta by "1000
\multiply\@tempcnta by "1000
\advance\dec@mcc by -\@tempcnta
\newcommand*{\DecimalMathComma}{\iflanguage{french}%
                                 {\mathcode`\,=\dec@mcc}{}%
   \FB@addto{extras}{\mathcode`\,=\dec@mcc}%
}
\newcommand*{\StandardMathComma}{\mathcode`\,=\std@mcc
   \FB@addto{extras}{\mathcode`\,=\std@mcc}%
}
\FB@addto{noextras}{\mathcode`\,=\std@mcc}
```

\nombre

The command `\nombre` is now borrowed from `numprint.sty` for LaTeX 2ε. There is no point to maintain the former tricky code when a package is dedicated to do the same job and more. For Plain based formats, `\nombre` no longer formats numbers, it prints them as is and issues a warning about the change.
Fake command `\nombre` for Plain based formats, warning users of frenchb v. 1.x. of the change.

```
\newcommand*{\nombre}[1]{{#1}\fb@warning{*** \noexpand\nombre
                                 no longer formats numbers\string! ***}}
```

The next definitions only make sense for LaTeX 2ε. For Plain based formats, let's activate LuaTeX punctuation if necessary, then cleanup and exit. Temporary fix: `\l@french` is not properly set by babel 3.9h with Plain LuaTeX format.

```
\let\FBstop@here\relax
\def\FBclean@on@exit{\let\ifLaTeXe\undefined
                     \let\LaTeXetrue\undefined
                     \let\LaTeXefalse\undefined}
\ifx\magnification\@undefined
\else
   \def\FBstop@here{\ifFB@luatex@punct
                       \activate@luatexpunct
                    \fi
                    \FBclean@on@exit
                    \ldf@quit\CurrentOption\endinput}
\fi
\FBstop@here
```

What follows is for LaTeX 2ε *only*; as all LaTeX 2ε based formats include ε-TeX, we can use `\ifdefined` now. We redefine `\nombre` for LaTeX 2ε. A warning is issued at the first call of `\nombre` if `\numprint` is not defined, suggesting what to do. The package `numprint` is *not* loaded automatically by frenchb because of possible options conflict.

```
\renewcommand*{\nombre}[1]{\Warning@nombre\numprint{#1}}
\newcommand*{\Warning@nombre}{%
   \ifdefined\numprint
   \else
      \PackageWarning{frenchb.ldf}{%
         \protect\nombre\space now relies on package numprint.sty,%
         \MessageBreak add \protect
         \usepackage[autolanguage]{numprint}\MessageBreak
         to your preamble *after* loading babel,\MessageBreak
         see file numprint.pdf for more options.\MessageBreak
         \protect\nombre\space called}%
      \global\let\Warning@nombre\relax
   \fi
}
```

2.7 Caption names

The next step consists in defining the French equivalents for the LaTeX caption names.

`\captionsfrench` Let's first define `\captionsfrench` which sets all strings used in the four standard document classes provided with LaTeX.
New implementation for caption names (requires babel's 3.9 or up).

```
\StartBabelCommands*{\BabelLanguages}{captions}
      [unicode, fontenc=EU1 EU2, charset=utf8]
   \SetString{\refname}{Références}
   \SetString{\abstractname}{Résumé}
   \SetString{\prefacename}{Préface}
   \SetString{\contentsname}{Table des matières}
   \SetString{\ccname}{Copie à }
   \SetString{\proofname}{Démonstration}
   \SetStringLoop{ordinal#1}{%
       Première,Deuxième,Troisième,Quatrième,Cinquième,%
       Sixième,Septième,Huitième,Neuvième,Dixième,Onzième,%
       Douzième,Treizième,Quatorzième,Quinzième,Seizième,%
       Dix-septième,Dix-huitième,Dix-neuvième,Vingtième}
\StartBabelCommands*{\BabelLanguages}{captions}
   \SetString{\refname}{R\'ef\'erences}
   \SetString{\abstractname}{R\'esum\'e}
   \SetString{\bibname}{Bibliographie}
   \SetString{\prefacename}{Pr\'eface}
   \SetString{\chaptername}{Chapitre}
   \SetString{\appendixname}{Annexe}
   \SetString{\contentsname}{Table des mati\‘eres}
```

```
  \SetString{\listfigurename}{Table des figures}
  \SetString{\listtablename}{Liste des tableaux}
  \SetString{\indexname}{Index}
  \SetString{\figurename}{{\scshape Figure}}
  \SetString{\tablename}{{\scshape Table}}
  \SetString{\pagename}{page}
  \SetString{\seename}{voir}
  \SetString{\alsoname}{voir aussi}
  \SetString{\enclname}{P.~J. }
  \SetString{\ccname}{Copie \‘a }
  \SetString{\headtoname}{}
  \SetString{\proofname}{D\’emonstration}
  \SetString{\glossaryname}{Glossaire}
```

“Première partie” instead of “Part I”.

```
  \SetStringLoop{ordinal#1}{%
      Premi\‘ere,Deuxi\‘eme,Troisi\‘eme,Quatri\‘eme,Cinqui\‘eme,%
      Sixi\‘eme,Septi\‘eme,Huiti\‘eme,Neuvi\‘eme,Dixi\‘eme,Onzi\‘eme,%
      Douzi\‘eme,Treizi\‘eme,Quatorzi\‘eme,Quinzi\‘eme,Seizi\‘eme,%
      Dix-septi\‘eme,Dix-huiti\‘eme,Dix-neuvi\‘eme,Vingti\‘eme}
  \AfterBabelCommands{%
      \DeclareRobustCommand*{\FB@emptypart}{\def\thepart{}}%
      \DeclareRobustCommand*{\FB@Rpart}{\def\thepart{\Roman{part}}}%
     }
  \SetString{\partname}{%
      \csname ordinal\romannumeral\value{part}\endcsname\space
      partie\FB@emptypart}
\EndBabelCommands
```

Up to v2.6h frenchb used to merge `\captionsfrenchb` and `\captionsfrancais` into `\captionsfrench` at `\begin{document}`. This is deprecated in favor of the new (much simpler!) syntax introduced in babel 3.9. No need to define `\captionscanadien` and `\captionsacadian` either.

`\CaptionSeparator` Let’s consider now captions in figures and tables. In French, captions in figures and tables should never be printed as ‘Figure 1:’ which is the default in standard LaTeX 2_ε classes; the ‘:’ is made active too late, no space is added before it. With LuaLaTeX and XeLaTeX, this glitch doesn’t occur, you get ‘Figure 1 :’ which is correct in French. With pdfLaTeX frenchb provides the following workaround.

The standard definition of `\@makecaption` (e.g., the one provided in article.cls, report.cls, book.cls which is frozen for LaTeX 2_ε according to Frank Mittelbach), is saved in `\STD@makecaption`. ‘AtBeginDocument’ we compare it to its current definition (some classes like `memoir`, koma-script classes, AMS classes, ua-thesis.cls. . . change it). If they are identical, frenchb just adds a hook called `\FBCaption@Separator` to `\@makecaption`; `\FBCaption@Separator` defaults to ‘: ’ as in the standard `\@makecaption` and will be changed to ‘ : ’ in French ‘AtBeginDocument’; it can be also set to `\CaptionSeparator` (‘ – ’) using `CustomiseFigTabCaptions`.

While saving the standard definition of `\@makecaption` we have to make sure that characters ‘:’ and ‘>’ have `\catcode` 12 (frenchb makes ‘:’ active and `spanish.ldf`

makes ‘>’ active).

```
\bgroup
  \catcode`:=12 \catcode`>=12 \relax
  \long\gdef\STD@makecaption#1#2{%
    \vskip\abovecaptionskip
    \sbox\@tempboxa{#1: #2}%
    \ifdim \wd\@tempboxa >\hsize
      #1: #2\par
    \else
      \global \@minipagefalse
      \hb@xt@\hsize{\hfil\box\@tempboxa\hfil}%
    \fi
    \vskip\belowcaptionskip}
\egroup
```

The `caption` and `floatrow` packages are compatible with frenchb if they are loaded after babel (a warning is printed in the .log file when they are loaded too early).
No warning is issued for SMF and AMS classes as their layout of captions is compatible with French typographic standards.
With `memoir` and koma-script classes, frenchb customises `\captiondelim` or `\captionformat` in French (unless option `CustomiseFigTabcaptions` is set to `false`) and issues no warning.
When `\@makecaption` has been changed by another class or package, a warning is printed in the .log file.

```
\newif\if@FBwarning@capsep
\@FBwarning@capseptrue
\newcommand{\FBWarning}[2]{\PackageWarning{#1}{#2}}
\newcommand*{\CaptionSeparator}{\space\textendash\space}
\def\FBCaption@Separator{:\space}
\long\def\FB@makecaption#1#2{%
  \vskip\abovecaptionskip
  \sbox\@tempboxa{#1\FBCaption@Separator #2}%
  \ifdim \wd\@tempboxa >\hsize
    #1\FBCaption@Separator #2\par
  \else
    \global \@minipagefalse
    \hb@xt@\hsize{\hfil\box\@tempboxa\hfil}%
  \fi
  \vskip\belowcaptionskip}
```

Disable the standard warning with AMS and SMF classes.

```
\@ifclassloaded{amsart}{\@FBwarning@capsepfalse}{}
\@ifclassloaded{amsbook}{\@FBwarning@capsepfalse}{}
\@ifclassloaded{amsdtx}{\@FBwarning@capsepfalse}{}
\@ifclassloaded{amsldoc}{\@FBwarning@capsepfalse}{}
\@ifclassloaded{amproc}{\@FBwarning@capsepfalse}{}
\@ifclassloaded{smfart}{\@FBwarning@capsepfalse}{}
\@ifclassloaded{smfbook}{\@FBwarning@capsepfalse}{}
```

Disable the standard warning unless high punctuation is active.

```
\ifFB@active@punct\else\@FBwarning@capsepfalse\fi
```

No warning with `memoir` or koma-script classes: they change `\@makecaption` but we will manage to customise them in French later on (see below after executing `\FBprocess@options`). No warning either if `\@makecaption` is undefined.

```
\newif\ifFB@koma
\@ifclassloaded{memoir}{\@FBwarning@capsepfalse}{}
\@ifclassloaded{scrartcl}{\@FBwarning@capsepfalse\FB@komatrue}{}
\@ifclassloaded{scrbook}{\@FBwarning@capsepfalse\FB@komatrue}{}
\@ifclassloaded{scrreprt}{\@FBwarning@capsepfalse\FB@komatrue}{}
\ifdefined\@makecaption\else\@FBwarning@capsepfalse\fi
```

Check if package `caption` is loaded now (before babel/frenchb), then issue a warning advising to load it after babel/frenchb and disable the standard warning.

```
\@ifpackageloaded{caption}
    {\FBWarning{frenchb.ldf}%
        {Please load the "caption" package\MessageBreak
         AFTER babel/frenchb; reported}%
     \@FBwarning@capsepfalse}%
     {}
```

Same for package `floatrow`.

```
\@ifpackageloaded{floatrow}
    {\FBWarning{frenchb.ldf}%
        {Please load the "floatrow" package\MessageBreak
         AFTER babel/frenchb; reported}%
     \@FBwarning@capsepfalse}%
     {}
```

First check the definition of `\@makecaption`, change it or issue a warning in case it has been changed by a class or package not (yet) compatible with frenchb; then change the definition of `\FBCaption@Separator`, taking care that the colon is typeset correctly in French (*not* 'Figure 1: légende').

```
\AtBeginDocument{%
  \ifx\@makecaption\STD@makecaption
      \global\let\@makecaption\FB@makecaption
```

Do not overwrite `\FBCaption@Separator` if already saved as ': ' for other languages and set to `\CaptionSeparator` by `\extrasfrench` when French is the main language.

```
      \ifFBOldFigTabCaptions
      \else
        \def\FBCaption@Separator{\iflanguage{french}{ : }{: }}%
      \fi
      \ifFBCustomiseFigTabCaptions
         \ifx\bbl@main@language\FB@french
            \def\FBCaption@Separator{\CaptionSeparator}%
         \fi
      \fi
      \@FBwarning@capsepfalse
  \fi
  \if@FBwarning@capsep
    \FBWarning{frenchb.ldf}%
```

```
      {Figures' and tables' captions might look like\MessageBreak
       'Figure 1:' which is wrong in French.\MessageBreak
       Check your class or packages to change this;\MessageBreak
       reported}%
  \fi
  \let\FB@makecaption\relax
  \let\STD@makecaption\relax
}
```

2.8 Dots...

\FBtextellipsis

LaTeX 2ε's standard definition of `\dots` in text-mode is `\textellipsis` which includes a `\kern` at the end; this space is not wanted in some cases (before a closing brace for instance) and `\kern` breaks hyphenation of the next word. We define `\FBtextellipsis` for French (in LaTeX 2ε only).

The `\if` construction in the LaTeX 2ε definition of `\dots` doesn't allow the use of `xspace` (`xspace` is always followed by a `\fi`), so we use the AMS-LaTeX construction of `\dots`; this has to be done 'AtBeginDocument' not to be overwritten when `amsmath.sty` is loaded after babel.

LY1 has a ready made character for `\textellipsis`, it should be used in French too. The same is true for Unicode fonts in use with XeTeX and LuaTeX.

```
\ifFBunicode
  \let\FBtextellipsis\textellipsis
\else
  \DeclareTextSymbol{\FBtextellipsis}{LY1}{133}
  \DeclareTextCommandDefault{\FBtextellipsis}{%
    .\kern\fontdimen3\font.\kern\fontdimen3\font.\xspace}
\fi
```

`\Mdots@` and `\Tdots@` hold the definitions of `\dots` in Math and Text mode. They default to those of amsmath-2.0, and will revert to standard LaTeX definitions 'AtBeginDocument', if amsmath has not been loaded. `\Mdots@` doesn't change when switching from/to French, while `\Tdots@` is `\FBtextellipsis` in French and `\Tdots@ORI` otherwise.

```
\newcommand*{\Tdots@}{\@xp\textellipsis}
\newcommand*{\Mdots@}{\@xp\mdots@}
\AtBeginDocument{\DeclareRobustCommand*{\dots}{\relax
                  \csname\ifmmode M\else T\fi dots@\endcsname}%
                  \ifdefined\@xp\else\let\@xp\relax\fi
                  \ifdefined\mdots@\else\let\Mdots@\mathellipsis\fi
                 }
\def\bbl@frenchdots{\babel@save\Tdots@ \let\Tdots@\FBtextellipsis}
\FB@addto{extras}{\bbl@frenchdots}
```

2.9 Setup options: keyval stuff

All setup options are handled by command `\frenchbsetup{}` using the keyval syntax. A list of flags is defined and set to a default value which will possibly be changed

'AtEndOfPackage' if French is the main language. After this, `\frenchbsetup{}` eventually modifies the preset values of these flags.
Option processing can occur either in `\frenchbsetup{}`, but *only for options explicitly set* by `\frenchbsetup{}`, or 'AtBeginDocument'; any option affecting `\extrasfrench{}` *must* be processed by `\frenchbsetup{}`: when French is the main language, `\extrasfrench{}` is executed by babel when it switches the main language and this occurs *before* reading the stuff postponed by frenchb 'AtBeginDocument'. Reexecuting `\extrasfrench{}` is a possibility which was used up to v2.6h, it has been dropped in v3.0a because of its side-effects (f.i. `\babel@save` and `\babel@savevariable` no longer work).

`\frenchbsetup` Let's now define this command which reads and sets the options to be processed either immediately (i.e. just after setting the key) or later (at `\begin{document}`) by `\FBprocess@options`. `\frenchbsetup{}` can only be called in the preamble.

```
\newcommand*{\frenchbsetup}[1]{%
  \setkeys{FB}{#1}%
}%
\@onlypreamble\frenchbsetup
```

We define a collection of conditionals with their defaults (true or false).

```
\newif\ifFBStandardLayout            \FBStandardLayouttrue
\newif\ifFBGlobalLayoutFrench        \FBGlobalLayoutFrenchtrue
\newif\ifFBReduceListSpacing         \FBReduceListSpacingfalse
\newif\ifFBListOldLayout             \FBListOldLayoutfalse
\newif\ifFBCompactItemize            \FBCompactItemizefalse
\newif\ifFBStandardItemizeEnv        \FBStandardItemizeEnvtrue
\newif\ifFBStandardEnumerateEnv      \FBStandardEnumerateEnvtrue
\newif\ifFBStandardItemLabels        \FBStandardItemLabelstrue
\newif\ifFBStandardLists             \FBStandardListstrue
\newif\ifFBIndentFirst               \FBIndentFirstfalse
\newif\ifFBFrenchFootnotes           \FBFrenchFootnotesfalse
\newif\ifFBAutoSpaceFootnotes        \FBAutoSpaceFootnotesfalse
\newif\ifFBOriginalTypewriter        \FBOriginalTypewriterfalse
\newif\ifFBThinColonSpace            \FBThinColonSpacefalse
\newif\ifFBThinSpaceInFrenchNumbers  \FBThinSpaceInFrenchNumbersfalse
\newif\ifFBFrenchSuperscripts        \FBFrenchSuperscriptstrue
\newif\ifFBLowercaseSuperscripts     \FBLowercaseSuperscriptstrue
\newif\ifFBPartNameFull              \FBPartNameFulltrue
\newif\ifFBCustomiseFigTabCaptions   \FBCustomiseFigTabCaptionsfalse
\newif\ifFBOldFigTabCaptions         \FBOldFigTabCaptionsfalse
\newif\ifFBSuppressWarning           \FBSuppressWarningfalse
\newif\ifFBShowOptions               \FBShowOptionsfalse
```

The defaults values of these flags have been choosen so that frenchb does not change anything regarding the global layout. `\bbl@main@language`, set by the last option of babel, controls the global layout of the document. 'AtEndOfPackage' we check the main language in `\bbl@main@language`; if it is French, the values of some flags have to be changed to ensure a French looking layout for the whole document (even in parts written in languages other than French); the end-user will then be able to customise the values of all these flags with `\frenchbsetup{}`.

```
\edef\FB@french{\CurrentOption}
\AtEndOfPackage{%
  \ifx\bbl@main@language\FB@french
    \FBGlobalLayoutFrenchtrue
    \FBReduceListSpacingtrue
    \FBStandardItemizeEnvfalse
    \FBStandardEnumerateEnvfalse
    \FBStandardItemLabelsfalse
    \FBIndentFirsttrue
    \FBFrenchFootnotestrue
    \FBAutoSpaceFootnotestrue
    \FBCustomiseFigTabCaptionstrue
  \else
    \FBGlobalLayoutFrenchfalse
  \fi
```

frenchb being an option of babel, it cannot load a package (keyval) while `frenchb.ldf` is read, so we defer the loading of keyval and the options setup at the end of babel's loading.

```
  \RequirePackage{keyval}%
  \define@key{FB}{StandardLayout}[true]%
                       {\csname FBStandardLayout#1\endcsname
                        \ifFBStandardLayout
                          \FBReduceListSpacingfalse
                          \FBStandardItemizeEnvtrue
                          \FBStandardItemLabelstrue
                          \FBStandardEnumerateEnvtrue
                          \FBIndentFirstfalse
                          \FBFrenchFootnotesfalse
                          \FBAutoSpaceFootnotesfalse
                          \FBGlobalLayoutFrenchfalse
                        \else
                          \FBReduceListSpacingtrue
                          \FBStandardItemizeEnvfalse
                          \FBStandardItemLabelsfalse
                          \FBStandardEnumerateEnvfalse
                          \FBIndentFirsttrue
                          \FBFrenchFootnotestrue
                          \FBAutoSpaceFootnotestrue
                        \fi}%
  \define@key{FB}{GlobalLayoutFrench}[true]%
                       {\csname FBGlobalLayoutFrench#1\endcsname
```

If this key is set to `true` when French is the main language, nothing to do: all flags keep their default value. If this key is set to `false`, nothing to do either: `\babel@save` will do the job.

```
                       \ifFBGlobalLayoutFrench
                          \ifx\bbl@main@language\FB@french
                          \else
                             \PackageWarning{frenchb.ldf}%
                               {Option 'GlobalLayoutFrench' skipped:%
```

```
                              \MessageBreak French is *not*
                              babel's last option.\MessageBreak}%
                        \fi
                       \fi}%
\define@key{FB}{ReduceListSpacing}[true]%
                     {\csname FBReduceListSpacing#1\endcsname}%
\define@key{FB}{ListOldLayout}[true]%
                     {\csname FBListOldLayout#1\endcsname
                      \ifFBListOldLayout
                        \FBStandardEnumerateEnvtrue
                        \renewcommand*{\FrenchLabelItem}{\textendash}%
                       \fi}%
\define@key{FB}{CompactItemize}[true]%
                     {\csname FBCompactItemize#1\endcsname
                      \ifFBCompactItemize
                        \FBStandardItemizeEnvfalse
                        \FBStandardEnumerateEnvfalse
                       \else
                        \FBStandardItemizeEnvtrue
                        \FBStandardEnumerateEnvtrue
                       \fi}%
\define@key{FB}{StandardItemizeEnv}[true]%
                     {\csname FBStandardItemizeEnv#1\endcsname}%
\define@key{FB}{StandardEnumerateEnv}[true]%
                     {\csname FBStandardEnumerateEnv#1\endcsname}%
\define@key{FB}{StandardItemLabels}[true]%
                     {\csname FBStandardItemLabels#1\endcsname}%
\define@key{FB}{ItemLabels}{%
      \renewcommand*{\FrenchLabelItem}{#1}}%
\define@key{FB}{ItemLabeli}{%
      \renewcommand*{\Frlabelitemi}{#1}}%
\define@key{FB}{ItemLabelii}{%
    \renewcommand*{\Frlabelitemii}{#1}}%
\define@key{FB}{ItemLabeliii}{%
      \renewcommand*{\Frlabelitemiii}{#1}}%
\define@key{FB}{ItemLabeliv}{%
    \renewcommand*{\Frlabelitemiv}{#1}}%
\define@key{FB}{StandardLists}[true]%
                     {\csname FBStandardLists#1\endcsname
                      \ifFBStandardLists
                        \FBReduceListSpacingfalse
                        \FBCompactItemizefalse
                        \FBStandardItemizeEnvtrue
                        \FBStandardEnumerateEnvtrue
                        \FBStandardItemLabelstrue
                       \else
                        \FBReduceListSpacingtrue
                        \FBCompactItemizetrue
                        \FBStandardItemizeEnvfalse
                        \FBStandardEnumerateEnvfalse
                        \FBStandardItemLabelsfalse
```

```
                      \fi}%
\define@key{FB}{IndentFirst}[true]%
                    {\csname FBIndentFirst#1\endcsname}%
\define@key{FB}{FrenchFootnotes}[true]%
                    {\csname FBFrenchFootnotes#1\endcsname}%
\define@key{FB}{AutoSpaceFootnotes}[true]%
                    {\csname FBAutoSpaceFootnotes#1\endcsname}%
\define@key{FB}{AutoSpacePunctuation}[true]%
                    {\csname FBAutoSpacePunctuation#1\endcsname}%
\define@key{FB}{OriginalTypewriter}[true]%
                    {\csname FBOriginalTypewriter#1\endcsname}%
\define@key{FB}{ThinColonSpace}[true]%
                    {\csname FBThinColonSpace#1\endcsname}%
\define@key{FB}{ThinSpaceInFrenchNumbers}[true]%
                    {\csname FBThinSpaceInFrenchNumbers#1\endcsname}%
\define@key{FB}{FrenchSuperscripts}[true]%
                    {\csname FBFrenchSuperscripts#1\endcsname}
\define@key{FB}{LowercaseSuperscripts}[true]%
                    {\csname FBLowercaseSuperscripts#1\endcsname}
\define@key{FB}{PartNameFull}[true]%
                    {\csname FBPartNameFull#1\endcsname
                     \ifFBPartNameFull
                     \else
                        \FB@addto{captions}{%
                           \def\partname{Partie\protect\FB@Rpart}}%
                     \fi}%
\define@key{FB}{CustomiseFigTabCaptions}[true]%
        {\csname FBCustomiseFigTabCaptions#1\endcsname}%
\define@key{FB}{OldFigTabCaptions}[true]%
        {\csname FBOldFigTabCaptions#1\endcsname
```

`\CurrentOption` no longer defined. It's value has been saved in `\FB@CurOpt` while reading `frenchb.ldf`.

```
            \ifFBOldFigTabCaptions
               \FB@addto{extras}{\babel@save\FBCaption@Separator
                         \def\FBCaption@Separator{\CaptionSeparator}}%
            \fi}%
\define@key{FB}{SuppressWarning}[true]%
                    {\csname FBSuppressWarning#1\endcsname
                     \ifFBSuppressWarning
                       \renewcommand{\FBWarning}[2]{\relax}%
                     \fi}%
\define@key{FB}{ShowOptions}[true]%
                    {\csname FBShowOptions#1\endcsname}%
```

Inputing French quotes as *single characters* when they are available on the keyboard (through a compose key for instance) is more comfortable than typing `\og` and `\fg`.

With pdfTeX (or old LuaTeX and XeTeX engines), quote characters are made active and expand to `\og\ignorespaces` and `{\fg}` respectively if the current language is French, and to `\guillemotleft` and `\guillemotright` otherwise (think of German quotes), this is done by `\FB@@og` and `\FB@@fg`; thus correct unbreakable spaces will

be added automatically to French quotes. The quote characters typed in depend on the input encoding, it can be single-byte (latin1, latin9, applemac,...) or multi-bytes (utf-8, utf8x); the `inputenc` package has to be loaded before the `\begin{document}` with the proper coding option, so we check if `\DeclareInputText` is defined.
Life is much simpler here with modern LuaTeX or XeTeX engines: we just have to activate the `\FB@addGUILspace` attribute for LuaTeX or set `\XeTeXcharclass` of quotes to the proper value for XeTeX.

```
\define@key{FB}{og}{%
     \ifFB@active@punct
        \newcommand*{\FB@@og}{%
          \iflanguage{french}%
            {\ifFBAutoSpaceGuill\FB@og\ignorespaces
             \else\guillemotleft
             \fi}%
            {\guillemotleft}}%
     \fi
     \ifFBunicode
```

LuaTeX or XeTeX in use, first try modern LuaTeX: we just need to set LuaTeX's attribute `\FB@addGUILspace` to 1,

```
       \ifFB@luatex@punct
         \FB@addGUILspace=1 \relax
       \else
```

then modern XeTeX, this more tricky:

```
         \ifFB@xetex@punct
```

`\XeTeXinterchartokenstate` is defined, we just need to set `\XeTeXcharclass` to `\FB@guilo` for the French opening quote (see subsection 2.2).

```
           \XeTeXcharclass"AB   = \FB@guilo
           \XeTeXcharclass"A0   = \FB@guilnul
           \XeTeXcharclass"202F = \FB@guilnul
         \else
```

then an old LuaTeX or XeTeX in use, the following trick for defining the active quote character is borrowed from `inputenc.dtx`.

```
           \catcode`#1=\active
           \bgroup
             \uccode`\~`#1%
             \uppercase{%
           \egroup
           \def~%
           }{\FB@@og}%
         \fi
       \fi
     \else
```

This is for conventional TeX engines:

```
       \AtBeginDocument{%
          \ifdefined\DeclareInputText
             \ifdefined\uc@dclc
```

Package inputenc with utf8x encoding loaded, use \uc@dclc,

```
                  \uc@dclc{171}{default}{\FB@@og}%
               \else
```

if encoding is not utf8x, try utf8...

```
                  \ifdefined\DeclareUnicodeCharacter
```

utf8 loaded, use \DeclareUnicodeCharacter,

```
                     \DeclareUnicodeCharacter{00AB}{\FB@@og}%
                  \else
```

if utf8 is not loaded either, we assume 8-bit character input encoding. Package MULEenc (from CJK) defines \mule@def to map characters to control sequences.

```
                     \@tempcnta‘#1\relax
                     \ifdefined\mule@def
                        \mule@def{11}{\FB@@og}%
                     \else
                        \DeclareInputText{\the\@tempcnta}{\FB@@og}%
                     \fi
                  \fi
               \fi
            \else
```

Package inputenc not loaded, no way...

```
               \PackageWarning{frenchb.ldf}%
                 {Option ‘og’ requires package inputenc.\MessageBreak}%
            \fi
         }%
      \fi
}%
```

Same code for the closing quote.

```
\define@key{FB}{fg}{%
     \ifFB@active@punct
        \newcommand*{\FB@@fg}{%
          \iflanguage{french}%
            {\ifFBAutoSpaceGuill\FB@fg
             \else\guillemotright
             \fi}%
            {\guillemotright}}%
     \fi
     \ifFBunicode
       \ifFB@luatex@punct
         \FB@addGUILspace=1 \relax
       \else
         \ifFB@xetex@punct
           \XeTeXcharclass"BB   = \FB@guilf
           \XeTeXcharclass"A0   = \FB@guilnul
           \XeTeXcharclass"202F = \FB@guilnul
         \else
           \catcode‘#1=\active
           \bgroup
             \uccode‘\~‘#1%
```

```
              \uppercase{%
            \egroup
            \def~%
            }{{\FB@@fg}}%
          \fi
        \fi
      \else
        \AtBeginDocument{%
          \ifdefined\DeclareInputText
             \ifdefined\uc@dclc
                \uc@dclc{187}{default}{\FB@@fg}%
             \else
                \ifdefined\DeclareUnicodeCharacter
                  \DeclareUnicodeCharacter{00BB}{\FB@@fg}%
                \else
                   \@tempcnta‘#1\relax
                    \ifdefined\mule@def
                       \mule@def{27}{{\FB@@fg}}%
                    \else
                       \DeclareInputText{\the\@tempcnta}{\FB@@fg}%
                    \fi
                \fi
             \fi
          \else
             \PackageWarning{frenchb.ldf}%
              {Option ‘fg’ requires package inputenc.\MessageBreak}%
          \fi
        }%
      \fi
  }%
}
```

\FBprocess@options `\FBprocess@options` will be executed at `\begin{document}`: it first checks about packages loaded in the preamble (possibly after babel) which customise lists: currently `enumitem`, `paralist` and `enumerate`; then it processes the options as set by `\frenchbsetup{}` or forced for compatibility with packages loaded in the preamble.

When French is the main language, `\extrasfrench` and `\captionsfrench` *have already been processed* by babel at `\begin{document}` *before* `\FBprocess@options`.

```
\newcommand*{\FBprocess@options}{%
```

Update flags if a package customising lists has been loaded, currently: `enumitem`, `paralist`, `enumerate`.

```
  \@ifpackageloaded{enumitem}{%
     \ifFBStandardItemizeEnv
     \else
        \FBStandardItemizeEnvtrue
        \PackageInfo{frenchb.ldf}%
          {Setting StandardItemizeEnv=true for\MessageBreak
           compatibility with enumitem package,\MessageBreak}%
     \fi
```

```
    \ifFBStandardEnumerateEnv
    \else
       \FBStandardEnumerateEnvtrue
       \PackageInfo{frenchb.ldf}%
         {Setting StandardEnumerateEnv=true for\MessageBreak
          compatibility with enumitem package,\MessageBreak}%
    \fi}{}%
\@ifpackageloaded{paralist}{%
    \ifFBStandardItemizeEnv
    \else
       \FBStandardItemizeEnvtrue
       \PackageInfo{frenchb.ldf}%
         {Setting StandardItemizeEnv=true for\MessageBreak
          compatibility with paralist package,\MessageBreak}%
    \fi
    \ifFBStandardEnumerateEnv
    \else
       \FBStandardEnumerateEnvtrue
       \PackageInfo{frenchb.ldf}%
         {Setting StandardEnumerateEnv=true for\MessageBreak
          compatibility with paralist package,\MessageBreak}%
    \fi}{}%
\@ifpackageloaded{enumerate}{%
    \ifFBStandardEnumerateEnv
    \else
       \FBStandardEnumerateEnvtrue
       \PackageInfo{frenchb.ldf}%
         {Setting StandardEnumerateEnv=true for\MessageBreak
          compatibility with enumerate package,\MessageBreak}%
    \fi}{}%
```

Reset `\FB@ufl`'s normal meaning and update lists' settings in case French is the main language:

```
\def\FB@ufl{\update@frenchlists}
\ifx\bbl@main@language\FB@french
   \update@frenchlists
\fi
```

The layout of footnotes is handled at the `\begin{document}` depending on the values of flags `FrenchFootnotes` and `AutoSpaceFootnotes` (see section 2.12), nothing has to be done here for footnotes.
`AutoSpacePunctuation` adds an unbreakable space (in French only) before the four active characters (:;!?) even if none has been typed before them.

```
\ifFBAutoSpacePunctuation
     \autospace@beforeFDP
\else
     \noautospace@beforeFDP
\fi
```

When `OriginalTypewriter` is set to `false` (the default), `\ttfamily`, `\rmfamily` and `\sffamily` are redefined as `\ttfamilyFB`, `\rmfamilyFB` and `\sffamilyFB` respectively to prevent addition of automatic spaces before the four active characters

in computer code.

```
\ifFBOriginalTypewriter
\else
   \let\ttfamilyORI\ttfamily
   \let\rmfamilyORI\rmfamily
   \let\sffamilyORI\sffamily
   \let\ttfamily\ttfamilyFB
   \let\rmfamily\rmfamilyFB
   \let\sffamily\sffamilyFB
\fi
```

ThinColonSpace changes the normal unbreakable space typeset in French before ':' to a thin space.

```
\ifFBThinColonSpace
  \ifFB@luatex@punct
     \FBcolonskip=\FBthinskip\relax
  \else
    \renewcommand*{\FBcolonspace}{\FBthinspace}%
  \fi
\fi
```

When true, ThinSpaceInFrenchNumbers redefines numprint.sty's command \npstylefrench to set \npthousandsep to \, (thinspace) instead of ~ (default) . This option has no effect if package numprint is not loaded with 'autolanguage'. As old versions of numprint did not define \npstylefrench, we have to provide this command.

```
\@ifpackageloaded{numprint}%
{\ifnprt@autolanguage
   \providecommand*{\npstylefrench}{}%
   \ifFBThinSpaceInFrenchNumbers
     \renewcommand*\npstylefrench{%
        \npthousandsep{\,}%
        \npdecimalsign{,}%
        \npproductsign{\cdot}%
        \npunitseparator{\,}%
        \npdegreeseparator{}%
        \nppercentseparator{\nprt@unitsep}%
        }%
   \else
     \renewcommand*\npstylefrench{%
        \npthousandsep{~}%
        \npdecimalsign{,}%
        \npproductsign{\cdot}%
        \npunitseparator{\,}%
        \npdegreeseparator{}%
        \nppercentseparator{\nprt@unitsep}%
        }%
   \fi
   \npaddtolanguage{french}{french}%
 \fi}{}%
```

FrenchSuperscripts: if true \up=\fup, else \up=\textsuperscript. Anyway

`\up*=\FB@up@fake`. The star-form `\up*{}` is provided for fonts that lack some superior letters: Adobe Jenson Pro and Utopia Expert have no “g superior” for instance.

```
\ifFBFrenchSuperscripts
  \DeclareRobustCommand*{\up}{\@ifstar{\FB@up@fake}{\fup}}%
\else
  \DeclareRobustCommand*{\up}{\@ifstar{\FB@up@fake}%
                                      {\textsuperscript}}%
\fi
```

`LowercaseSuperscripts`: if `true` let `\FB@lc` be `\lowercase`, else `\FB@lc` is redefined to do nothing.

```
\ifFBLowercaseSuperscripts
\else
  \renewcommand*{\FB@lc}[1]{##1}%
\fi
```

Use French `\CaptionSeparator` for koma-script and `memoir` classes unless `CustomiseFigTabCaptions` has been set to `false` (the default).

```
 \ifFBCustomiseFigTabCaptions
   \ifFB@koma
      \let\captionformat\CaptionSeparator
   \fi
   \@ifclassloaded{memoir}%
     {\captiondelim{\CaptionSeparator}}{}%
 \else
```

When `CustomiseFigTabCaptions` is `false`, have the colon behave properly in French.

```
   \ifFB@koma
      \def\captionformat{\iflanguage{french}{ : }{: }}%
   \fi
   \@ifclassloaded{memoir}%
      {\captiondelim{\iflanguage{french}{ : }{: }}}{}%
 \fi
```

`ShowOptions`: if `true`, print the list of all options to the `.log` file.

```
\ifFBShowOptions
  \GenericWarning{* }{%
   * **** List of possible options for frenchb ****\MessageBreak
   [Default values between brackets when frenchb is loaded *LAST*]%
   \MessageBreak
   ShowOptions=true [false]\MessageBreak
   StandardLayout=true [false]\MessageBreak
   GlobalLayoutFrench=false [true]\MessageBreak
   StandardLists=true [false]\MessageBreak
   IndentFirst=false [true]\MessageBreak
   ReduceListSpacing=false [true]\MessageBreak
   ListOldLayout=true [false]\MessageBreak
   StandardItemizeEnv=true [false]\MessageBreak
   StandardEnumerateEnv=true [false]\MessageBreak
   StandardItemLabels=true [false]\MessageBreak
```

```
    ItemLabels=\textemdash, \textbullet,
       \protect\ding{43},... [\textendash]\MessageBreak
    ItemLabeli=\textemdash, \textbullet,
       \protect\ding{43},... [\textendash]\MessageBreak
    ItemLabelii=\textemdash, \textbullet,
       \protect\ding{43},... [\textendash]\MessageBreak
    ItemLabeliii=\textemdash, \textbullet,
       \protect\ding{43},... [\textendash]\MessageBreak
    ItemLabeliv=\textemdash, \textbullet,
       \protect\ding{43},... [\textendash]\MessageBreak
    FrenchFootnotes=false [true]\MessageBreak
    AutoSpaceFootnotes=false [true]\MessageBreak
    AutoSpacePunctuation=false [true]\MessageBreak
    OriginalTypewriter=true [false]\MessageBreak
    ThinColonSpace=true [false]\MessageBreak
    ThinSpaceInFrenchNumbers=true [false]\MessageBreak
    FrenchSuperscripts=false [true]\MessageBreak
    LowercaseSuperscripts=false [true]\MessageBreak
    PartNameFull=false [true]\MessageBreak
    SuppressWarning=true [false]\MessageBreak
    CustomiseFigTabCaptions=false [true]\MessageBreak
    og= <left quote character>, fg= <right quote character>%
    \MessageBreak
    ********************************************%
    \MessageBreak\protect\frenchbsetup{ShowOptions}}
  \fi
}
```

At `\begin{document}`, we have to provide an `\xspace` command in case the `xspace` package is not loaded, do some setup for `hyperref`'s bookmarks, execute `\FBprocess@options`, switch LuaTeX punctuation on and issue some warnings if necessary.

```
\AtBeginDocument{%
  \providecommand*{\xspace}{\relax}%
```

Let's redefine some commands in `hyperref`'s bookmarks.

```
  \ifdefined\pdfstringdefDisableCommands
    \pdfstringdefDisableCommands{%
      \let\up\relax
      \let\fup\relax
      \let\degre\textdegree
      \let\degres\textdegree
      \def\ieme{e\xspace}%
      \def\iemes{es\xspace}%
      \def\ier{er\xspace}%
      \def\iers{ers\xspace}%
      \def\iere{re\xspace}%
      \def\ieres{res\xspace}%
      \def\FrenchEnumerate#1{#1\degre\space}%
      \def\FrenchPopularEnumerate#1{#1\degre)\space}%
      \def\No{N\degre\space}%
```

```
        \def\no{n\degre\space}%
        \def\Nos{N\degre\space}%
        \def\nos{n\degre\space}%
        \def\FB@og{\guillemotleft\space}%
        \def\FB@fg{\space\guillemotright}%
        \def\at{@}%
        \def\circonflexe{\string^}%
        \def\tild{\string~}%
        \let\bsc\textsc
      }%
  \fi
```

It is time to process the options set with `\frenchbsetup{}` or later.

```
  \FBprocess@options
```

With LuaTeX engines it is time to load file `frenchb.lua` (`\FBthinskip` and `\FBcolonskip` values are set now).

```
\ifFB@luatex@punct
  \activate@luatexpunct
\fi
```

Some warnings are issued when output font encodings are not properly set. With XeLaTeX or LuaLaTeX, `fontspec.sty` and `xunicode.sty` should be loaded; with (pdf)LaTeX, a warning is issued when OT1 encoding is in use at the `\begin{document}`. Mind that `\encodingdefault` is defined as 'long', defining `\FBOTone` with `\newcommand*` would fail!

```
  \ifFBunicode
     \ifdefined\DeclareUTFcharacter
     \else
        \PackageWarning{frenchb.ldf}%
         {Add \protect\usepackage{fontspec} to the\MessageBreak
          preamble of your document,}%
     \fi
  \else
     \begingroup \newcommand{\FBOTone}{OT1}%
     \ifx\encodingdefault\FBOTone
       \PackageWarning{frenchb.ldf}%
          {OT1 encoding should not be used for French.%
           \MessageBreak
           Add \protect\usepackage[T1]{fontenc} to the
           preamble\MessageBreak of your document,}%
     \fi
     \endgroup
  \fi
}
```

2.10 French lists

`\listFB` `\listORI` `\FB@listVsettings`

Vertical spacing in lists should be shorter in French texts than the defaults provided by LaTeX. Note that the easy way, just changing values of vertical spacing parameters when entering French and restoring them to their defaults on exit would not work;

so we define the command `\FB@listVsettings` to hold the settings to be used by the French variant `\listFB` of `\list`. Note that switching to `\listFB` reduces vertical spacing in *all* environments built on `\list`: `itemize`, `enumerate`, `description`, but also `abstract`, `quotation`, `quote` and `verse`...
The amount of vertical space before and after a list is given by `\topsep` + `\parskip` (+ `\partopsep` if the list starts a new paragraph). IMHO, `\parskip` should be added *only* when the list starts a new paragraph, so I subtract `\parskip` from `\topsep` and add it back to `\partopsep`; this will normally make no difference because `\parskip`'s default value is 0pt, but will be noticeable when `\parskip` is *not* null.

```
\let\listORI\list
\let\endlistORI\endlist
\def\FB@listVsettings{%
      \setlength{\itemsep}{0.4ex plus 0.2ex minus 0.2ex}%
      \setlength{\parsep}{0.4ex plus 0.2ex minus 0.2ex}%
      \setlength{\topsep}{0.8ex plus 0.4ex minus 0.4ex}%
      \setlength{\partopsep}{0.4ex plus 0.2ex minus 0.2ex}%
```

`\parskip` is of type 'skip', its mean value only (*not the glue*) should be subtracted from `\topsep` and added to `\partopsep`, so convert `\parskip` to a 'dimen' using `\@tempdima`.

```
      \@tempdima=\parskip
      \addtolength{\topsep}{-\@tempdima}%
      \addtolength{\partopsep}{\@tempdima}%
}
\def\listFB#1#2{\listORI{#1}{\FB@listVsettings #2}}
\let\endlistFB\endlist
```

Let's now consider French itemize-lists. They differ from those provided by the standard LaTeX 2$_\varepsilon$ classes:

- The '•' is never used in French itemize-lists, an emdash '—' or an en-dash '–' is preferred for all levels. The item label to be used in French is stored in `\FrenchLabelItem}`, it defaults to '—' and can be changed using `\frenchbsetup{}` (see section 2.9).
- Vertical spacing between items, before and after the list, should be *null* with *no glue* added;
- In French the labels of itemize-lists are vertically aligned as follows:

  Text starting at 'parindent'
  ⇐ Leftmargin
  – first item...
  – first second level item
  – next one...
  – second item...

`\FrenchLabelItem`
`\Frlabelitemi`
`\Frlabelitemii`
`\Frlabelitemiii`
`\Frlabelitemiv`

Default labels for French itemize-lists (same label for all levels):

```
\newcommand*{\FrenchLabelItem}{\textemdash}
\newcommand*{\Frlabelitemi}{\FrenchLabelItem}
\newcommand*{\Frlabelitemii}{\FrenchLabelItem}
```

```
\newcommand*{\Frlabelitemiii}{\FrenchLabelItem}
\newcommand*{\Frlabelitemiv}{\FrenchLabelItem}
```

\listindentFB
\labelwidthFB

Let's define two lengths `\listindentFB` and `\labelwidthFB` to customise lists' horizontal indentations. They are given silly values here (−1 pt) in order to eventually enable their customisation in the preamble. They will get reasonnable defaults later when entering French (see `\bbl@frenchlabelitems`) unless they have been customised.

```
\newlength\listindentFB
\setlength{\listindentFB}{-1pt}
\newlength\labelwidthFB
\setlength{\labelwidthFB}{-1pt}
```

\FB@listHsettings
\leftmarginFB

`\FB@listHsettings` holds the new horizontal settings chosen for French lists itemize and enumerate starting with version 2.6a. They are based on the look resquested in French for itemize-lists.

```
\newlength\leftmarginFB
\def\FB@listHsettings{%
   \leftmarginFB\labelwidthFB
   \advance\leftmarginFB \labelsep
   \leftmargini\leftmarginFB
   \advance\leftmargini \listindentFB
   \leftmarginii\leftmarginFB
   \leftmarginiii\leftmarginFB
   \leftmarginiv\leftmarginFB
   \leftmargin\csname leftmargin\romannumeral\the\@listdepth\endcsname
}
```

\itemizeFB
\FB@itemizesettings

New environment for French itemize-lists.
`\FB@itemizesettings` does two things: first suppress all vertical spaces including glue when option `ReduceListSpacing` is set, then set horizontal indentations according to `\FB@listHsettings` unless option `ListOldLayout` is `true` (compatibility with lists up to v. 2.5k).

```
\def\FB@itemizesettings{%
   \ifFBReduceListSpacing
      \setlength{\itemsep}{\z@}%
      \setlength{\parsep}{\z@}%
      \setlength{\topsep}{\z@}%
      \setlength{\partopsep}{\z@}%
      \@tempdima=\parskip
      \addtolength{\topsep}{-\@tempdima}%
      \addtolength{\partopsep}{\@tempdima}%
   \fi
   \settowidth{\labelwidth}{\csname\@itemitem\endcsname}%
   \ifFBListOldLayout
      \setlength{\leftmargin}{\labelwidth}%
      \addtolength{\leftmargin}{\labelsep}%
      \addtolength{\leftmargin}{\parindent}%
   \else
      \FB@listHsettings
```

```
    \fi
}
```

The definition of `\itemizeFB` follows the one of `\itemize` in standard LaTeX 2$_\varepsilon$ classes (see `ltlists.dtx`), spaces are customised by `\FB@itemizesettings`.

```
\def\itemizeFB{%
    \ifnum \@itemdepth >\thr@@\@toodeep\else
      \advance\@itemdepth\@ne
      \edef\@itemitem{labelitem\romannumeral\the\@itemdepth}%
      \expandafter
      \listORI
      \csname\@itemitem\endcsname
      \FB@itemizesettings
    \fi
}
\let\enditemizeFB\endlistORI

\def\labelitemsFB{%
    \let\labelitemi\Frlabelitemi
    \let\labelitemii\Frlabelitemii
    \let\labelitemiii\Frlabelitemiii
    \let\labelitemiv\Frlabelitemiv
    \ifdim\labelwidthFB<\z@
      \settowidth{\labelwidthFB}{\FrenchLabelItem}%
    \fi
    \ifdim\listindentFB<\z@
      \ifdim\parindent=\z@
        \setlength{\listindentFB}{1.5em}%
      \else
        \setlength{\listindentFB}{\parindent}%
      \fi
    \fi
}
```

`\enumerateFB` The definition of `\enumerateFB`, new to version 2.6a, follows the one of `\enumerate` in standard LaTeX 2$_\varepsilon$ classes (see `ltlists.dtx`), vertical spaces are customised (or not) via `\list` (=`\listFB` or `\listORI`) and horizontal spaces (leftmargins) are borrowed from `itemize` lists via `\FB@listHsettings`.

```
\def\enumerateFB{%
  \ifnum \@enumdepth >\thr@@\@toodeep\else
    \advance\@enumdepth\@ne
    \edef\@enumctr{enum\romannumeral\the\@enumdepth}%
    \expandafter
    \list
      \csname label\@enumctr\endcsname
      {\FB@listHsettings
       \usecounter\@enumctr\def\makelabel##1{\hss\llap{##1}}}%
  \fi
}
\let\endenumerateFB\endlistORI
```

\descriptionFB Same tuning for the description environment (see the original definition in classes.dtx). Customisable \listindentFB added to \itemindent (first level only).

```
\def\descriptionFB{%
      \list{}{\FB@listHsettings
              \labelwidth\z@
              \itemindent-\leftmargin
              \ifnum\@listdepth=1
                 \advance\itemindent by \listindentFB
              \fi
              \let\makelabel\descriptionlabel}%
}
\let\enddescriptionFB\endlistORI
```

\update@frenchlists \bbl@frenchlistlayout \bbl@nonfrenchlistlayout \update@frenchlists will setup lists according to the options of \frenchbsetup{}.

```
\def\update@frenchlists{%
  \ifFBReduceListSpacing \let\list\listFB \fi
  \ifFBStandardItemizeEnv
  \else \let\itemize\itemizeFB \fi
  \ifFBStandardItemLabels
  \else \labelitemsFB \fi
  \ifFBStandardEnumerateEnv
  \else \let\enumerate\enumerateFB \let\description\descriptionFB \fi
}
```

In order to ensure compatibility with packages customising lists, the command \update@frenchlists should not be included in \extrasfrench yet, so we also define \FB@ufl as \relax, it will be redefined as \update@frenchlists in due time 'AtBeginDocument' by \FBprocess@options, see p. 48.

```
\def\FB@ufl{\relax}
\def\bbl@frenchlistlayout{%
  \ifFBGlobalLayoutFrench
  \else
    \babel@save\list         \babel@save\itemize
    \babel@save\enumerate    \babel@save\description
    \babel@save\labelitemi   \babel@save\labelitemii
    \babel@save\labelitemiii \babel@save\labelitemiv
  \fi
  \FB@ufl
}
\def\bbl@nonfrenchlistlayout{%
  \ifFBGlobalLayoutFrench
     \update@frenchlists
  \fi
}
\FB@addto{extras}{\bbl@frenchlistlayout}
\FB@addto{noextras}{\bbl@nonfrenchlistlayout}
```

2.11 French indentation of sections

\bbl@frenchindent
\bbl@nonfrenchindent

In French the first paragraph of each section should be indented, this is another difference with US-English. This is controlled by the flag `\if@afterindent`.
We will need to save the value of the flag `\if@afterindent` 'AtBeginDocument' before eventually changing its value.

```
\def\bbl@frenchindent{%
  \ifFBGlobalLayoutFrench\else\babel@save\@afterindentfalse\fi
  \ifFBIndentFirst
     \let\@afterindentfalse\@afterindenttrue
     \@afterindenttrue
  \fi}
\def\bbl@nonfrenchindent{%
  \ifFBGlobalLayoutFrench
     \ifFBIndentFirst
        \@afterindenttrue
     \fi
  \fi}
\FB@addto{extras}{\bbl@frenchindent}
\FB@addto{noextras}{\bbl@nonfrenchindent}
```

2.12 Formatting footnotes

The `bigfoot` package deeply changes the way footnotes are handled. When `bigfoot` is loaded, we just warn the user that frenchb will drop the customisation of footnotes. The layout of footnotes is controlled by two flags `\ifFBAutoSpaceFootnotes` and `\ifFBFrenchFootnotes` which are set by options of `\frenchbsetup{}` (see section 2.9). The layout of footnotes *does not depend* on the current language (just think of two footnotes on the same page looking different because one was called in a French part, the other one in English!).
When `\ifFBAutoSpaceFootnotes` is true, `\@footnotemark` (the definition of which is saved at the `\begin{document}` in order to include any customisation that packages might have done) is redefined to add a thin space before the number or symbol calling a footnote (any space typed in is removed first). This has no effect on the layout of the footnote itself.

```
\AtBeginDocument{\@ifpackageloaded{bigfoot}%
                    {\PackageInfo{frenchb.ldf}%
                      {bigfoot package in use.\MessageBreak
                       frenchb will NOT customise footnotes;\MessageBreak
                       reported}}%
                    {\let\@footnotemarkORI\@footnotemark
                     \def\@footnotemarkFB{\leavevmode\unskip\unkern
                                          \,\@footnotemarkORI}%
                     \ifFBAutoSpaceFootnotes
                       \let\@footnotemark\@footnotemarkFB
                     \fi}%
                }
```

We then define `\@makefntextFB`, a variant of `\@makefntext` which is responsible for the layout of footnotes, to match the specifications of the French 'Imprimerie

Nationale': footnotes will be indented by `\parindentFFN`, numbers (if any) typeset on the baseline (instead of superscripts) and followed by a dot and an half quad space. Whenever symbols are used to number footnotes (as in `\thanks` for instance), we switch back to the standard layout (the French layout of footnotes is meant for footnotes numbered by Arabic or Roman digits).
The value of `\parindentFFN` will be redefined at the `\begin{document}`, as the maximum of `\parindent` and 1.5em *unless* it has been set in the preamble (the weird value 10in is just for testing whether `\parindentFFN` has been set or not).

```
\newcommand*{\dotFFN}{.}
\newcommand*{\kernFFN}{\kern .5em}
\newdimen\parindentFFN
\parindentFFN=10in
\def\ftnISsymbol{\@fnsymbol\c@footnote}
\long\def\@makefntextFB#1{\ifx\thefootnote\ftnISsymbol
                            \@makefntextORI{#1}%
                          \else
                            \parindent=\parindentFFN
                            \rule\z@\footnotesep
                            \setbox\@tempboxa\hbox{\@thefnmark}%
                            \ifdim\wd\@tempboxa>\z@
                              \llap{\@thefnmark}\dotFFN\kernFFN
                            \fi #1
                          \fi}%
```

We save the standard definition of `\@makefntext` at the `\begin{document}`, and then redefine `\@makefntext` according to the value of flag `\ifFBFrenchFootnotes` (true or false).

```
\AtBeginDocument{\@ifpackageloaded{bigfoot}{}%
                  {\ifdim\parindentFFN<10in
                   \else
                      \parindentFFN=\parindent
                      \ifdim\parindentFFN<1.5em \parindentFFN=1.5em \fi
                   \fi
                   \let\@makefntextORI\@makefntext
                   \long\def\@makefntext#1{%
                      \ifFBFrenchFootnotes
                         \@makefntextFB{#1}%
                      \else
                         \@makefntextORI{#1}%
                      \fi}%
                  }%
                }
```

For compatibility reasons, we provide definitions for the commands dealing with the layout of footnotes in frenchb version 1.6. `\frenchbsetup{}` (see in section 2.9) should be preferred for setting these options. `\StandardFootnotes` may still be used locally (in minipages for instance), that's why the test `\ifFBFrenchFootnotes` is done inside `\@makefntext`.

```
\newcommand*{\AddThinSpaceBeforeFootnotes}{\FBAutoSpaceFootnotestrue}
\newcommand*{\FrenchFootnotes}{\FBFrenchFootnotestrue}
```

```
\newcommand*{\StandardFootnotes}{\FBFrenchFootnotesfalse}
```

2.13 Clean up and exit

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

```
\FBclean@on@exit
\ldf@finish\CurrentOption
```

3 Change History

v2.0

General: \parindentFFN not changed if already defined (required by JA for cah-gut.cls). 58

Added warning for OT1 encoding. 51

Footnotes are now printed by default 'à la française' for the whole document. 57

New command \frenchbsetup added for global customisation. 40

\bsc: \hbox dropped, replaced by \kern0pt. 33

\captionsfrench: 'Fig.' changed to 'Figure' and 'Tab.' to 'Table'. . . . 36

\datefrench: 2 '\relax' added in \today's definition. 30

\FBtextellipsis: Added special case for LY1 encoding, see bug report from Bruno Voisin (2004/05/18). 40

\nombre: \nombre now requires numprint.sty. 35

v2.0b

General: Footnotes: Just do nothing (except warning) when the bigfoot package is loaded. 57

v2.0c

General: There is no need to define here numprint's command \npstylefrench, it will be redefined 'AtBeginDocument' by \FBprocess@options. 36

\frenchbsetup: Option ThinSpaceInFrenchNumbers added. 41

v2.0d

\frenchbsetup: Options og and fg changed: limit the definition to French so that quote characters can be used in German. 41

v2.0e

\frenchbsetup: New option: StandardLists. 41

v2.0f

\frenchbsetup: StandardLayout option had no effect on lists. Test moved to \FBprocess@options. 41

Two typos corrected in option StandardLists: [false] → [true] and StandardLayout → StandardLists. 41

v2.0g

\frenchbsetup: Revert previous change to StandardLayout. This option must set the three flags \FBReduceListSpacingfalse, \FBCompactItemizefalse, and \FBStandardItemLabeltrue instead of \FBStandardListstrue, so that later options can still change their value before executing \FBprocess@options. Same thing for option StandardLists. . 41

v2.1a

General: Command \fup added to produce better superscripts than \textsuperscript. 30

\datefrench: \today changed (correction in 2.0 was wrong: \today was printed without spaces in toc). 30

\frenchbsetup: New option: FrenchSuperscripts to define \up as \fup or as \textsuperscript. 41

New option: LowercaseSuperscripts. 41

v2.1b

General: Disable some commands in bookmarks. 51

\fup: Command \fup changed to use real superscripts from fourier v. 1.6. 30

v2.1c

General: Added commands \Nos and \nos. 32

\degres: Provide a temporary definition (hyperref safe) of \degres in case it has to be expanded in the preamble (by beamer's \title command for instance). 34

\up: Provide a temporary definition (hyperref safe) of \up in case it has to be expanded in the preamble (by beamer's \title command for instance). 30

v2.1d

General: Argument of \ProvidesLanguage changed above from 'french' to