1 The Slovak language

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

For this language the macro `\q` is defined. It was used with the letters (`t`, `d`, `l`, and `L`) and adds a `'` to them to simulate a 'hook' that should be there. The result looks like ť. Since the the T1 font encoding has the corresponding characters it is mapped to `\v`. Therefore we recommend using T1 font encoding. If you don't want to use this encoding, please, feel free to redefine `\q` in your file. I think babel will honour this `;-)`.

For this language the characters `"`, `'` and `^` are made active. In table 1 an overview is given of its purpose. Also the vertical placement of the umlaut can be controlled this way.

`"a`	`\"a`, also implemented for the other lowercase and uppercase vowels.
`^d`	`\q d`, also implemented for l, t and L.
`^c`	`\v c`, also implemented for C, D, N, n, T, Z and z.
`^o`	`\^o`, also implemented for O.
`'a`	`\'a`, also implemented for the other lowercase and uppercase l, r, y and vowels.
`"|`	disable ligature at this position.
`"-`	an explicit hyphen sign, allowing hyphenation in the rest of the word.
`""`	like `"-`, but producing no hyphen sign (for compund words with hyphen, e.g. `x-""y`).
`"~`	for a compound word mark without a breakpoint.
`"=`	for a compound word mark with a breakpoint, allowing hyphenation in the composing words.
`"‘`	for German left double quotes (looks like ,,).
`"’`	for German right double quotes.
`"<`	for French left double quotes (similar to <<).
`">`	for French right double quotes (similar to >>).

Table 1: The extra definitions made by `slovak.ldf`

The quotes in table 1 can also be typeset by using the commands in table 2.

1.1 Compatibility

Great care has been taken to ensure backward compatibility with CSLaTeX. In particular, documents which load this file with `\usepackage{slovak}` should produce

[1] The file described in this section has version number v3.1 and was last revised on 2006/10/07. It was originally written by Jana Chlebikova (`chlebik@euromath.dk`) and modified by Tobias Schlemmer (`Tobias.Schlemmer@web.de`). It was then rewritten by Petr Tesařík (`babel@tesarici.cz`).

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

Table 2: More commands which produce quotes, defined by `slovak.ldf`

identical output with no modifications to the source. Additionally, all the $\mathcal{CS}$LaTeX options are recognized:

IL2, T1, OT1
: These options set the default font encoding. Please note that their use is deprecated. You should use the `fontenc` package to select font encoding.

split, nosplit
: These options control whether hyphenated words are automatically split according to Slovak typesetting rules. With the split option “je-li” is hyphenated as “je-/-li”. The nosplit option disables this behavior.

  The use of this option is strongly discouraged, as it breaks too many common things—hyphens cannot be used in labels, negative arguments to TeX primitives will not work in horizontal mode (use `\minus` as a workaround), and there are a few other peculiarities with using this mode.

nocaptions
: This option was used in $\mathcal{CS}$LaTeX to set up Czech/Slovak typesetting rules, but leave the original captions and dates. The recommended way to achieve this is to use English as the main language of the document and use the environment `otherlanguage*` for Czech text.

olduv
: There are two version of `\uv`. The older one allows the use of `\verb` inside the quotes but breaks any respective kerning with the quotes (like that in $\mathcal{CS}$ fonts). The newer one honors the kerning in the font but does not allow `\verb` inside the quotes.

  The new version is used by default in LaTeX 2_ε and the old version is used with plain TeX. You may use olduv to override the default in LaTeX 2_ε.

cstex This option was used to include the commands `\csprimeson` and `\csprimesoff`. Since these commands are always included now, it has been removed and the empty definition lasts for compatibility.

1.2 Implementation

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

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

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

```
\ifx\l@slovak\@undefined
    \@nopatterns{Slovak}
    \adddialect\l@slovak0\fi
```

We need to define these macros early in the process.

```
\def\cs@iltw@{IL2}
\newif\ifcs@splithyphens
\cs@splithyphensfalse
```

If Babel is not loaded, we provide compatibility with $\mathcal{CS}$LaTeX. However, if macro `\@ifpackageloaded` is not defined, we assume to be loaded from plain and provide compatibility with csplain. Of course, this does not work well with LaTeX 2.09, but I doubt anyone will ever want to use this file with LaTeX 2.09.

```
\ifx\@ifpackageloaded\@undefined
  \let\cs@compat@plain\relax
  \message{csplain compatibility mode}
\else
  \@ifpackageloaded{babel}{}{%
    \let\cs@compat@latex\relax
    \message{cslatex compatibility mode}}
\fi
\ifx\cs@compat@latex\relax
  \ProvidesPackage{slovak}[2005/12/22 v3.0 CSTeX Slovak style]
```

Declare $\mathcal{CS}$LaTeX options (see also the descriptions on page 2).

```
  \DeclareOption{IL2}{\def\encodingdefault{IL2}}
  \DeclareOption {T1}{\def\encodingdefault {T1}}
  \DeclareOption{OT1}{\def\encodingdefault{OT1}}
  \DeclareOption{nosplit}{\cs@splithyphensfalse}
  \DeclareOption{split}{\cs@splithyphenstrue}
  \DeclareOption{nocaptions}{\let\cs@nocaptions=\relax}
  \DeclareOption{olduv}{\let\cs@olduv=\relax}
  \DeclareOption{cstex}{\relax}
```

Make IL2 encoding the default. This can be overriden with the other font encoding options.

```
\ExecuteOptions{\cs@iltw@}
```

Now, process the user-supplied options.

```
\ProcessOptions
```

Standard LaTeX 2ε does not include the IL2 encoding in the format. The encoding can be loaded later using the fontenc package, but CSLaTeX included IL2 by default. This means existing documents for CSLaTeX do not load that package, so load the encoding ourselves in compatibility mode.

```
\ifx\encodingdefault\cs@iltw@
  \input il2enc.def
\fi
```

Restore the definition of `\CurrentOption`, clobbered by processing the options.

```
\def\CurrentOption{slovak}
\fi
```

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

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

```
\@namedef{captions\CurrentOption}{%
  \def\prefacename{Predhovor}%
  \def\refname{Literat\'ura}%
  \def\abstractname{Abstrakt}%
  \def\bibname{Literat\'ura}%
  \def\chaptername{Kapitola}%
  \def\appendixname{Dodatok}%
  \def\contentsname{Obsah}%
  \def\listfigurename{Zoznam obr\'azkov}%
  \def\listtablename{Zoznam tabuliek}%
  \def\indexname{Index}%
  \def\figurename{Obr.}%
  \def\tablename{Tabu\v{l}ka}%
  \def\partname{\v{C}as\v{t}}%
  \def\enclname{Pr\'{\i}loha}%
  \def\ccname{cc.}%
  \def\headtoname{Pre}%
  \def\pagename{Str.}%
  \def\seename{vi\v{d}}%
  \def\alsoname{vi\v{d} tie\v{z}}%
  \def\proofname{D\^okaz}%
  \def\glossaryname{Slovn\'{\i}k}%
  }%
```

`\dateslovak` The macro `\dateslovak` redefines the command `\today` to produce Slovak dates.

```
\@namedef{date\CurrentOption}{%
```

```
\def\today{\number\day.~\ifcase\month\or
  janu\'ara\or febru\'ara\or marca\or apr\'{\i}la\or m\'aja\or
  j\'una\or j\'ula\or augusta\or septembra\or okt\'obra\or
  novembra\or decembra\fi
  \space \number\year}}
```

`\extrasslovak` `\noextrasslovak`

The macro `\extrasslovak` will perform all the extra definitions needed for the Slovak language. The macro `\noextrasslovak` is used to cancel the actions of `\extrasslovak`.

For Slovak texts `\frenchspacing` should be in effect. Language group for shorthands is also set here.

```
\expandafter\addto\csname extras\CurrentOption\endcsname{%
  \bbl@frenchspacing
  \languageshorthands{slovak}}
\expandafter\addto\csname noextras\CurrentOption\endcsname{%
  \bbl@nonfrenchspacing}
```

```
\expandafter\addto\csname extras\CurrentOption\endcsname{%
  \babel@save\q\let\q\v}
```

For Slovak three characters are used to define shorthands, they need to be made active.

```
\ifx\cs@compat@latex\relax\else
  \initiate@active@char{^}
  \addto\extrasslovak{\bbl@activate{^}}
  \addto\noextrasslovak{\bbl@deactivate{^}}
  \initiate@active@char{"}
  \addto\extrasslovak{\bbl@activate{"}\umlautlow}
  \addto\noextrasslovak{\bbl@deactivate{"}\umlauthigh}
  \initiate@active@char{'}
  \@ifpackagewith{babel}{activeacute}{%
    \addto\extrasslovak{\bbl@activate{'}}
    \addto\noextrasslovak{\bbl@deactivate{'}}%
    }{}
\fi
```

`\sq` `\dq`

We save the original single and double quote characters in `\sq` and `\dq` to make them available later. The math accent `\"` can now be typed as `"`.

```
\begingroup\catcode`\"=12\catcode`\'=12
\def\x{\endgroup
  \def\sq{'}
  \def\dq{"}}
\x
```

The slovak hyphenation patterns should be used with `\lefthyphenmin` set to 2 and `\righthyphenmin` set to 3.

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

In order to prevent problems with the active `^` we add a shorthand on system level which expands to a 'normal `^`.

```
\ifx\cs@compat@latex\relax\else
  \declare@shorthand{system}{^}{\csname normal@char\string^\endcsname}
```

Now we can define the doublequote macros: the umlauts,

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

tremas,

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

other slovak characters

```
  \declare@shorthand{slovak}{^c}{\textormath{\v{c}\allowhyphens}{\check{c}}}
  \declare@shorthand{slovak}{^d}{\textormath{\q{d}\allowhyphens}{\check{d}}}
  \declare@shorthand{slovak}{^l}{\textormath{\q{l}\allowhyphens}{\check{l}}}
  \declare@shorthand{slovak}{^n}{\textormath{\v{n}\allowhyphens}{\check{n}}}
  \declare@shorthand{slovak}{^o}{\textormath{\^{o}\allowhyphens}{\hat{o}}}
  \declare@shorthand{slovak}{^s}{\textormath{\v{s}\allowhyphens}{\check{s}}}
  \declare@shorthand{slovak}{^t}{\textormath{\q{t}\allowhyphens}{\check{t}}}
  \declare@shorthand{slovak}{^z}{\textormath{\v{z}\allowhyphens}{\check{z}}}
  \declare@shorthand{slovak}{^C}{\textormath{\v{C}\allowhyphens}{\check{C}}}
  \declare@shorthand{slovak}{^D}{\textormath{\v{D}\allowhyphens}{\check{D}}}
  \declare@shorthand{slovak}{^L}{\textormath{\q{L}\allowhyphens}{\check{L}}}
  \declare@shorthand{slovak}{^N}{\textormath{\v{N}\allowhyphens}{\check{N}}}
  \declare@shorthand{slovak}{^O}{\textormath{\^{O}\allowhyphens}{\hat{O}}}
  \declare@shorthand{slovak}{^S}{\textormath{\v{S}\allowhyphens}{\check{S}}}
  \declare@shorthand{slovak}{^T}{\textormath{\v{T}\allowhyphens}{\check{T}}}
  \declare@shorthand{slovak}{^Z}{\textormath{\v{Z}\allowhyphens}{\check{Z}}}
```

acute accents,

```
  \@ifpackagewith{babel}{activeacute}{%
    \declare@shorthand{slovak}{'a}{\textormath{\'a\allowhyphens}{^{\prime}a}}
    \declare@shorthand{slovak}{'e}{\textormath{\'e\allowhyphens}{^{\prime}e}}
    \declare@shorthand{slovak}{'i}{\textormath{\'\i{}\allowhyphens}{^{\prime}i}}
    \declare@shorthand{slovak}{'l}{\textormath{\'l\allowhyphens}{^{\prime}l}}
    \declare@shorthand{slovak}{'o}{\textormath{\'o\allowhyphens}{^{\prime}o}}
    \declare@shorthand{slovak}{'r}{\textormath{\'r\allowhyphens}{^{\prime}r}}
    \declare@shorthand{slovak}{'u}{\textormath{\'u\allowhyphens}{^{\prime}u}}
    \declare@shorthand{slovak}{'y}{\textormath{\'y\allowhyphens}{^{\prime}y}}
    \declare@shorthand{slovak}{'A}{\textormath{\'A\allowhyphens}{^{\prime}A}}
    \declare@shorthand{slovak}{'E}{\textormath{\'E\allowhyphens}{^{\prime}E}}
    \declare@shorthand{slovak}{'I}{\textormath{\'I\allowhyphens}{^{\prime}I}}
    \declare@shorthand{slovak}{'L}{\textormath{\'L\allowhyphens}{^{\prime}l}}
    \declare@shorthand{slovak}{'O}{\textormath{\'O\allowhyphens}{^{\prime}O}}
```

```
    \declare@shorthand{slovak}{'R}{\textormath{\'R\allowhyphens}{^{\prime}R}}
    \declare@shorthand{slovak}{'U}{\textormath{\'U\allowhyphens}{^{\prime}U}}
    \declare@shorthand{slovak}{'Y}{\textormath{\'Y\allowhyphens}{^{\prime}Y}}
    \declare@shorthand{slovak}{''}{%
      \textormath{\textquotedblright}{\sp\bgroup\prim@s'}}
    }{}

```

and some additional commands:

```
  \declare@shorthand{slovak}{"-}{\nobreak\-\bbl@allowhyphens}
  \declare@shorthand{slovak}{"|}{%
    \textormath{\penalty\@M\discretionary{-}{}{\kern.03em}%
                \bbl@allowhyphens}{}}
  \declare@shorthand{slovak}{""}{\hskip\z@skip}
  \declare@shorthand{slovak}{"~}{\textormath{\leavevmode\hbox{-}}{-}}
  \declare@shorthand{slovak}{"=}{\cs@splithyphen}
\fi
```

`\v` LaTeX's normal `\v` accent places a caron over the letter that follows it (ŏ). This is not what we want for the letters d, t, l and L; for those the accent should change shape. This is acheived by the following.

```
\AtBeginDocument{%
  \DeclareTextCompositeCommand{\v}{OT1}{t}{%
    t\kern-.23em\raise.24ex\hbox{'}}
  \DeclareTextCompositeCommand{\v}{OT1}{d}{%
    d\kern-.13em\raise.24ex\hbox{'}}
  \DeclareTextCompositeCommand{\v}{OT1}{l}{\lcaron{}}
  \DeclareTextCompositeCommand{\v}{OT1}{L}{\Lcaron{}}}
```

`\lcaron` `\Lcaron` For the letters `l` and `L` we want to disinguish between normal fonts and monospaced fonts.

```
\def\lcaron{%
  \setbox0\hbox{M}\setbox\tw@\hbox{i}%
  \ifdim\wd0>\wd\tw@\relax
    l\kern-.13em\raise.24ex\hbox{'}\kern-.11em%
  \else
    l\raise.45ex\hbox to\z@{\kern-.35em '\hss}%
  \fi}
\def\Lcaron{%
  \setbox0\hbox{M}\setbox\tw@\hbox{i}%
  \ifdim\wd0>\wd\tw@\relax
    L\raise.24ex\hbox to\z@{\kern-.28em'\hss}%
  \else
    L\raise.45ex\hbox to\z@{\kern-.40em '\hss}%
  \fi}
```

Initialize active quotes. CSLaTeX provides a way of converting English-style quotes into Slovak-style ones. Both single and double quotes are affected, i.e. ``text'' is converted to something like ,,text`` and `text' is converted to

`,text‘`. This conversion can be switched on and off with `\csprimeson` and `\csprimesoff`.[2]

These quotes present various troubles, e.g. the kerning is broken, apostrophes are converted to closing single quote, some primitives are broken (most notably the `\catcode‘\`⟨*char*⟩ syntax will not work any more), and writing them to `.aux` files cannot be handled correctly. For these reasons, these commands are only available in $\mathcal{C}_S$LaTeX compatibility mode.

```
\ifx\cs@compat@latex\relax
  \let\cs@ltxprim@s\prim@s
  \def\csprimeson{%
    \catcode‘‘\active \catcode‘’\active \let\prim@s\bbl@prim@s}
  \def\csprimesoff{%
    \catcode‘‘12 \catcode‘’12 \let\prim@s\cs@ltxprim@s}
  \begingroup\catcode‘‘\active
  \def\x{\endgroup
    \def‘{\futurelet\cs@next\cs@openquote}
    \def\cs@openquote{%
      \ifx‘\cs@next \expandafter\cs@opendq
      \else \expandafter\clq
      \fi}%
  }\x
  \begingroup\catcode‘’\active
  \def\x{\endgroup
    \def’{\textormath{\futurelet\cs@next\cs@closequote}
                     {^\bgroup\prim@s}}
    \def\cs@closequote{%
      \ifx’\cs@next \expandafter\cs@closedq
      \else \expandafter\crq
      \fi}%
  }\x
  \def\cs@opendq{\clqq\let\cs@next= }
  \def\cs@closedq{\crqq\let\cs@next= }
```

The way I recommend for typesetting quotes in Slovak documents is to use shorthands similar to those used in German.

```
\else
  \declare@shorthand{slovak}{"‘}{\clqq}
  \declare@shorthand{slovak}{"’}{\crqq}
  \declare@shorthand{slovak}{"<}{\flqq}
  \declare@shorthand{slovak}{">}{\frqq}
\fi
```

`\clqq` This is the CS opening quote, which is similar to the German quote (`\glqq`) but the kerning is different.

For the OT1 encoding, the quote is constructed from the right double quote (i.e. the “Opening quotes” character) by moving it down to the baseline and shifting it to the right, or to the left if italic correction is positive.

[2] By the way, the names of these macros are misleading, because the handling of primes in math mode is rather marginal, the most important thing being the handling of quotes...

For T1, the "German Opening quotes" is used. It is moved to the right and the total width is enlarged. This is done in an attempt to minimize the difference between the OT1 and T1 versions.

```
\ProvideTextCommand{\clqq}{OT1}{%
  \set@low@box{\textquotedblright}%
  \setbox\@ne=\hbox{l\/}\dimen\@ne=\wd\@ne
  \setbox\@ne=\hbox{l}\advance\dimen\@ne-\wd\@ne
  \leavevmode
  \ifdim\dimen\@ne>\z@\kern-.1em\box\z@\kern.1em
    \else\kern.1em\box\z@\kern-.1em\fi\allowhyphens}
\ProvideTextCommand{\clqq}{T1}
  {\kern.1em\quotedblbase\kern-.0158em\relax}
\ProvideTextCommandDefault{\clqq}{\UseTextSymbol{OT1}\clqq}
```

`\crqq` For OT1, the CS closing quote is basically the same as `\grqq`, only the `\textormath` macro is not used, because as far as I know, `\grqq` does not work in math mode anyway.

For T1, the character is slightly wider and shifted to the right to match its OT1 counterpart.

```
\ProvideTextCommand{\crqq}{OT1}
  {\save@sf@q{\nobreak\kern-.07em\textquotedblleft\kern.07em}}
\ProvideTextCommand{\crqq}{T1}
  {\save@sf@q{\nobreak\kern.06em\textquotedblleft\kern.024em}}
\ProvideTextCommandDefault{\crqq}{\UseTextSymbol{OT1}\crqq}
```

`\clq` `\crq` Single CS quotes are similar to double quotes (see the discussion above).

```
\ProvideTextCommand{\clq}{OT1}
  {\set@low@box{\textquoteright}\box\z@\kern.04em\allowhyphens}
\ProvideTextCommand{\clq}{T1}
  {\quotesinglbase\kern-.0428em\relax}
\ProvideTextCommandDefault{\clq}{\UseTextSymbol{OT1}\clq}
\ProvideTextCommand{\crq}{OT1}
  {\save@sf@q{\nobreak\textquoteleft\kern.17em}}
\ProvideTextCommand{\crq}{T1}
  {\save@sf@q{\nobreak\textquoteleft\kern.17em}}
\ProvideTextCommandDefault{\crq}{\UseTextSymbol{OT1}\crq}
```

`\uv` There are two versions of `\uv`. The older one opens a group and uses `\aftergroup` to typeset the closing quotes. This version allows using `\verb` inside the quotes, because the enclosed text is not passed as an argument, but unfortunately it breaks any kerning with the quotes. Although the kerning with the opening quote could be fixed, the kerning with the closing quote cannot.

The newer version is defined as a command with one parameter. It preserves kerning but since the quoted text is passed as an argument, it cannot contain `\verb`.

Decide which version of `\uv` should be used. For sake of compatibility, we use the older version with plain TEX and the newer version with LATEX 2_ε.

```
\ifx\cs@compat@plain\@undefined\else\let\cs@olduv=\relax\fi
\ifx\cs@olduv\@undefined
  \DeclareRobustCommand\uv[1]{{\leavevmode\clqq#1\crqq}}
\else
  \DeclareRobustCommand\uv{\bgroup\aftergroup\closequotes
    \leavevmode\clqq\let\cs@next=}
  \def\closequotes{\unskip\crqq\relax}
\fi
```

`\cs@wordlen` Declare a counter to hold the length of the word after the hyphen.

```
\newcount\cs@wordlen
```

`\cs@hyphen` `\cs@endash` `\cs@emdash` Store the original hyphen in a macro. Ditto for the ligatures.

```
\begingroup\catcode`\-12
\def\x{\endgroup
  \def\cs@hyphen{-}
  \def\cs@endash{--}
  \def\cs@emdash{---}
```

`\cs@boxhyphen` Provide a non-breakable hyphen to be used when a compound word is too short to be split, i.e. the second part is shorter than `\righthyphenmin`.

```
  \def\cs@boxhyphen{\hbox{-}}
```

`\cs@splithyphen` The macro `\cs@splithyphen` inserts a split hyphen, while allowing both parts of the compound word to be hyphenated at other places too.

```
  \def\cs@splithyphen{\kern\z@
    \discretionary{-}{\char\hyphenchar\the\font}{-}\nobreak\hskip\z@}
}\x
```

- To minimize the effects of activating the hyphen character, the active definition expands to the non-active character in all cases where hyphenation cannot occur, i.e. if not typesetting (check `\protect`), not in horizontal mode, or in inner horizontal mode.

```
\initiate@active@char{-}
\declare@shorthand{slovak}{-}{%
  \ifx\protect\@typeset@protect
    \ifhmode
      \ifinner
        \bbl@afterelse\bbl@afterelse\bbl@afterelse\cs@hyphen
      \else
        \bbl@afterfi\bbl@afterelse\bbl@afterelse\cs@firsthyphen
      \fi
    \else
      \bbl@afterfi\bbl@afterelse\cs@hyphen
    \fi
  \else
    \bbl@afterfi\cs@hyphen
  \fi}
```

\cs@firsthyphen
\cs@firsthyph@n
\cs@secondhyphen
\cs@secondhyph@n

If we encounter a hyphen, check whether it is followed by a second or a third hyphen and if so, insert the corresponding ligature.

If we don't find a hyphen, the token found will be placed in `\cs@token` for further analysis, and it will also stay in the input.

```
\begingroup\catcode`\-\active
\def\x{\endgroup
  \def\cs@firsthyphen{\futurelet\cs@token\cs@firsthyph@n}
  \def\cs@firsthyph@n{%
    \ifx -\cs@token
      \bbl@afterelse\cs@secondhyphen
    \else
      \bbl@afterfi\cs@checkhyphen
    \fi}
  \def\cs@secondhyphen ##1{%
    \futurelet\cs@token\cs@secondhyph@n}
  \def\cs@secondhyph@n{%
    \ifx -\cs@token
      \bbl@afterelse\cs@emdash\@gobble
    \else
      \bbl@afterfi\cs@endash
    \fi}
}\x
```

\cs@checkhyphen

Check that hyphenation is enabled, and if so, start analyzing the rest of the word, i.e. initialize `\cs@word` and `\cs@wordlen` and start processing input with `\cs@scanword`.

```
\def\cs@checkhyphen{%
  \ifnum\expandafter\hyphenchar\the\font=`\-
    \def\cs@word{}\cs@wordlen\z@
    \bbl@afterelse\cs@scanword
  \else
    \cs@hyphen
  \fi}
```

\cs@scanword
\cs@continuescan
\cs@gettoken
\cs@gett@ken

Each token is first analyzed with `\cs@scanword`, which expands the token and passes the first token of the result to `\cs@gett@ken`. If the expanded token is not identical to the unexpanded one, presume that it might be expanded further and pass it back to `\cs@scanword` until you get an unexpandable token. Then analyze it in `\cs@examinetoken`.

The `\cs@continuescan` macro does the same thing as `\cs@scanword`, but it does not require the first token to be in `\cs@token` already.

```
\def\cs@scanword{\let\cs@lasttoken= \cs@token\expandafter\cs@gettoken}
\def\cs@continuescan{\let\cs@lasttoken\@undefined\expandafter\cs@gettoken}
\def\cs@gettoken{\futurelet\cs@token\cs@gett@ken}
\def\cs@gett@ken{%
  \ifx\cs@token\cs@lasttoken \def\cs@next{\cs@examinetoken}%
  \else \def\cs@next{\cs@scanword}%
  \fi \cs@next}
```

`cs@examinetoken` Examine the token in `\cs@token`:

- If it is a letter (catcode 11) or other (catcode 12), add it to `\cs@word` with `\cs@addparam`.
- If it is the `\char` primitive, add it with `\cs@expandchar`.
- If the token starts or ends a group, ignore it with `\cs@ignoretoken`.
- Otherwise analyze the meaning of the token with `\cs@checkchardef` to detect primitives defined with `\chardef`.

```
\def\cs@examinetoken{%
  \ifcat A\cs@token
    \def\cs@next{\cs@addparam}%
  \else\ifcat 0\cs@token
    \def\cs@next{\cs@addparam}%
  \else\ifx\char\cs@token
    \def\cs@next{\afterassignment\cs@expandchar\let\cs@token= }%
  \else\ifx\bgroup\cs@token
    \def\cs@next{\cs@ignoretoken\bgroup}%
  \else\ifx\egroup\cs@token
    \def\cs@next{\cs@ignoretoken\egroup}%
  \else\ifx\begingroup\cs@token
    \def\cs@next{\cs@ignoretoken\begingroup}%
  \else\ifx\endgroup\cs@token
    \def\cs@next{\cs@ignoretoken\endgroup}%
  \else
    \def\cs@next{\expandafter\expandafter\expandafter\cs@checkchardef
      \expandafter\meaning\expandafter\cs@token\string\char\end}%
  \fi\fi\fi\fi\fi\fi\fi\cs@next}
```

`\cs@checkchardef` Check the meaning of a token and if it is a primitive defined with `\chardef`, pass it to `\\@examinechar` as if it were a `\char` sequence. Otherwise, there are no more word characters, so do the final actions in `\cs@nosplit`.

```
\expandafter\def\expandafter\cs@checkchardef
  \expandafter#\expandafter1\string\char#2\end{%
    \def\cs@token{#1}%
    \ifx\cs@token\@empty
      \def\cs@next{\afterassignment\cs@examinechar\let\cs@token= }%
    \else
      \def\cs@next{\cs@nosplit}%
    \fi \cs@next}
```

`\cs@ignoretoken` Add a token to `\cs@word` but do not update the `\cs@wordlen` counter. This is mainly useful for group starting and ending primitives, which need to be preserved, but do not affect the word boundary.

```
\def\cs@ignoretoken#1{%
  \edef\cs@word{\cs@word#1}%
  \afterassignment\cs@continuescan\let\cs@token= }
```

`cs@addparam` Add a token to `\cs@word` and check its lccode. Note that this macro can only be used for tokens which can be passed as a parameter.

```
\def\cs@addparam#1{%
  \edef\cs@word{\cs@word#1}%
  \cs@checkcode{\lccode`#1}}
```

`\cs@expandchar` `\cs@examinechar` Add a `\char` sequence to `\cs@word` and check its lccode. The charcode is first parsed in `\cs@expandchar` and then the resulting `\chardef`-defined sequence is analyzed in `\cs@examinechar`.

```
\def\cs@expandchar{\afterassignment\cs@examinechar\chardef\cs@token=}
\def\cs@examinechar{%
  \edef\cs@word{\cs@word\char\the\cs@token\space}%
  \cs@checkcode{\lccode\cs@token}}
```

`\cs@checkcode` Check the lccode of a character. If it is zero, it does not count to the current word, so finish it with `\cs@nosplit`. Otherwise update the `\cs@wordlen` counter and go on scanning the word with `\cs@continuescan`. When enough characters are gathered in `\cs@word` to allow word break, insert the split hyphen and finish.

```
\def\cs@checkcode#1{%
  \ifnum0=#1
    \def\cs@next{\cs@nosplit}%
  \else
    \advance\cs@wordlen\@ne
    \ifnum\righthyphenmin>\the\cs@wordlen
      \def\cs@next{\cs@continuescan}%
    \else
      \cs@splithyphen
      \def\cs@next{\cs@word}%
    \fi
  \fi \cs@next}
```

`\cs@nosplit` Insert a non-breakable hyphen followed by the saved word.

```
\def\cs@nosplit{\cs@boxhyphen\cs@word}
```

`\cs@hyphen` The `\minus` sequence can be used where the active hyphen does not work, e.g. in arguments to TEX primitives in outer horizontal mode.

```
\let\minus\cs@hyphen
```

`\standardhyphens` `\splithyphens` These macros control whether split hyphens are allowed in Czech and/or Slovak texts. You may use them in any language, but the split hyphen is only activated for Czech and Slovak.

```
\def\standardhyphens{\cs@splithyphensfalse\cs@deactivatehyphens}
\def\splithyphens{\cs@splithyphenstrue\cs@activatehyphens}
```

`\cs@splitattr` Now we declare the `split` language attribute. This is similar to the `split` package option of cslatex, but it only affects Slovak, not Czech.

```
\def\cs@splitattr{\babel@save\ifcs@splithyphens\splithyphens}
```

```
\bbl@declare@ttribute{slovak}{split}{%
  \addto\extrasslovak{\cs@splitattr}}
```

`\cs@activatehyphens` `\cs@deactivatehyphens`

These macros are defined as `\relax` by default to prevent activating/deactivating the hyphen character. They are redefined when the language is switched to Czech/Slovak. At that moment the hyphen is also activated if split hyphens were requested with `\splithyphens`.

When the language is de-activated, de-activate the hyphen and restore the bogus definitions of these macros.

```
\let\cs@activatehyphens\relax
\let\cs@deactivatehyphens\relax
\expandafter\addto\csname extras\CurrentOption\endcsname{%
  \def\cs@activatehyphens{\bbl@activate{-}}%
  \def\cs@deactivatehyphens{\bbl@deactivate{-}}%
  \ifcs@splithyphens\cs@activatehyphens\fi}
\expandafter\addto\csname noextras\CurrentOption\endcsname{%
  \cs@deactivatehyphens
  \let\cs@activatehyphens\relax
  \let\cs@deactivatehyphens\relax}
```

`\cs@looseness` `\looseness`

One of the most common situations where an active hyphen will not work properly is the `\looseness` primitive. Change its definition so that it deactivates the hyphen if needed.

```
\let\cs@looseness\looseness
\def\looseness{%
  \ifcs@splithyphens
    \cs@deactivatehyphens\afterassignment\cs@activatehyphens \fi
  \cs@looseness}
```

`\cs@selectlanguage` `\cs@main@language`

Specifying the `nocaptions` option means that captions and dates are not redefined by default, but they can be switched on later with `\captionsslovak` and/or `\dateslovak`.

We mimic this behavior by redefining `\selectlanguage`. This macro is called once at the beginning of the document to set the main language of the document. If this is `\cs@main@language`, it disables the macros for setting captions and date. In any case, it restores the original definition of `\selectlanguage` and expands it.

The definition of `\selectlanguage` can be shared between Czech and Slovak; the actual language is stored in `\cs@main@language`.

```
\ifx\cs@nocaptions\@undefined\else
  \edef\cs@main@language{\CurrentOption}
  \ifx\cs@origselect\@undefined
    \let\cs@origselect=\selectlanguage
    \def\selectlanguage{%
      \let\selectlanguage\cs@origselect
      \ifx\bbl@main@language\cs@main@language
        \expandafter\cs@selectlanguage
```

```
      \else
        \expandafter\selectlanguage
      \fi}
    \def\cs@selectlanguage{%
      \cs@tempdisable{captions}%
      \cs@tempdisable{date}%
      \selectlanguage}
```

\cs@tempdisable `\cs@tempdisable` disables a language setup macro temporarily, i.e. the macro with the name of ⟨*#1*⟩`\bbl@main@language` just restores the original definition and purges the saved macro from memory.

```
    \def\cs@tempdisable#1{%
      \def\@tempa{cs@#1}%
      \def\@tempb{#1\bbl@main@language}%
      \expandafter\expandafter\expandafter\let
        \expandafter \csname\expandafter \@tempa \expandafter\endcsname
        \csname \@tempb \endcsname
      \expandafter\edef\csname \@tempb \endcsname{%
        \let \expandafter\noexpand \csname \@tempb \endcsname
          \expandafter\noexpand \csname \@tempa \endcsname
        \let \expandafter\noexpand\csname \@tempa \endcsname
          \noexpand\@undefined}}
```

These macros are not needed, once the initialization is over.

```
    \@onlypreamble\cs@main@language
    \@onlypreamble\cs@origselect
    \@onlypreamble\cs@selectlanguage
    \@onlypreamble\cs@tempdisable
  \fi
\fi
```

The encoding of mathematical fonts should be changed to IL2. This allows to use accented letter in some font families. Besides, documents do not use CM fonts if there are equivalents in CS-fonts, so there is no need to have both bitmaps of CM-font and CS-font.

`\@font@warning` and `\@font@info` are temporarily redefined to avoid annoying font warnings.

```
\ifx\cs@compat@plain\@undefined
\ifx\cs@check@enc\@undefined\else
  \def\cs@check@enc{
    \ifx\encodingdefault\cs@iltw@
      \let\cs@warn\@font@warning \let\@font@warning\@gobble
      \let\cs@info\@font@info    \let\@font@info\@gobble
      \SetSymbolFont{operators}{normal}{\cs@iltw@}{cmr}{m}{n}
      \SetSymbolFont{operators}{bold}{\cs@iltw@}{cmr}{bx}{n}
      \SetMathAlphabet\mathbf{normal}{\cs@iltw@}{cmr}{bx}{n}
      \SetMathAlphabet\mathit{normal}{\cs@iltw@}{cmr}{m}{it}
      \SetMathAlphabet\mathrm{normal}{\cs@iltw@}{cmr}{m}{n}
```

```
      \SetMathAlphabet\mathsf{normal}{\cs@iltw@}{cmss}{m}{n}
      \SetMathAlphabet\mathtt{normal}{\cs@iltw@}{cmtt}{m}{n}
      \SetMathAlphabet\mathbf{bold}{\cs@iltw@}{cmr}{bx}{n}
      \SetMathAlphabet\mathit{bold}{\cs@iltw@}{cmr}{bx}{it}
      \SetMathAlphabet\mathrm{bold}{\cs@iltw@}{cmr}{bx}{n}
      \SetMathAlphabet\mathsf{bold}{\cs@iltw@}{cmss}{bx}{n}
      \SetMathAlphabet\mathtt{bold}{\cs@iltw@}{cmtt}{m}{n}
      \let\@font@warning\cs@warn \let\cs@warn\@undefined
      \let\@font@info\cs@info    \let\cs@info\@undefined
    \fi
    \let\cs@check@enc\@undefined}
  \AtBeginDocument{\cs@check@enc}
\fi
\fi
```

`cs@undoiltw@` The thing is that LaTeX 2$_\varepsilon$ core only supports the T1 encoding and does not bother changing the uc/lc/sfcodes when encoding is switched. :(However, the IL2 encoding *does* change these codes, so if encoding is switched back from IL2, we must also undo the effect of this change to be compatible with LaTeX 2$_\varepsilon$. OK, this is not the right$^{\mathrm{TM}}$ solution but it works. Cheers to Petr Olšák.

```
\def\cs@undoiltw@{%
  \uccode158=208 \lccode158=158 \sfcode158=1000
  \sfcode159=1000
  \uccode165=133 \lccode165=165 \sfcode165=1000
  \uccode169=137 \lccode169=169 \sfcode169=1000
  \uccode171=139 \lccode171=171 \sfcode171=1000
  \uccode174=142 \lccode174=174 \sfcode174=1000
  \uccode181=149
  \uccode185=153
  \uccode187=155
  \uccode190=0   \lccode190=0
  \uccode254=222 \lccode254=254 \sfcode254=1000
  \uccode255=223 \lccode255=255 \sfcode255=1000}
```

`@@enc@update` Redefine the LaTeX 2$_\varepsilon$ internal function `\@@enc@update` to change the encodings correctly.

```
\ifx\cs@enc@update\@undefined
\ifx\@@enc@update\@undefined\else
  \let\cs@enc@update\@@enc@update
  \def\@@enc@update{\ifx\cf@encoding\cs@iltw@\cs@undoiltw@\fi
    \cs@enc@update
    \expandafter\ifnum\csname l@\languagename\endcsname=\the\language
      \expandafter\ifx
      \csname l@\languagename:\f@encoding\endcsname\relax
      \else
        \expandafter\expandafter\expandafter\let
          \expandafter\csname
          \expandafter l\expandafter @\expandafter\languagename
          \expandafter\endcsname\csname l@\languagename:\f@encoding\endcsname
```

```
      \fi
      \language=\csname l@\languagename\endcsname\relax
    \fi}
\fi\fi
```

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

```
\ldf@finish\CurrentOption
```

⟨/code⟩