The langselect Package

v0.9.0d 2025/08/10

Generate different language versions from a source

Matthias Werner[1]

https://github.com/tuc-osg/osglecture

This package supports the generation of documents in two or three language versions from a common source document. The package is part of the osglecture bundle and is integrated into the osglecture class and supported by the build script `ollm`, but can also be used independently.

Table of Contents

1 Introduction

Sometimes, multiple language versions of the same document are required. In the application area of the osglecture bundle, these are, for example, the lecture notes, lecture slides, and handouts for a course. It is often difficult to keep multiple parallel language versions consistent. This task is somewhat easier if all versions are contained in a common source document.

This package supports the generation of such language versions from a common LaTeX source document by providing macros for language variants and mechanisms for selecting the target language.

1. matthias.werner@informatik.tu-chemnitz.de

1.1 Alternatives

The are a number of alternatives to langselect.

Ad hoc macros It is relatively easy to create ad hoc solutions, such as the following

```
\newif\ifenglish
\englishtrue
\ifenglish
    Welcome!
\else
    ¡Bienvenido!
\fi

```

In fact, langselect basically follows this approach, but automates the creation of language macros, simplifies the sharing of non-language-dependent elements such as formulas, and enables the selection of the target language by different approaches.

multilang The sophisticated package multilang[2] by Richard Grewe creates language versions of given macros, whereby the active language is taken directly from Babel or Polyglossia. langselect has a similar goal, but a different (from the package author's point of view: more practical) interface. However, if you don't like the approach of langselect you should definitely take a look at multilang.

comment Victor Eijkhout wrote the package comment,[3] which makes it easy to output only certain sections of the document. This can also be used very well for language selection.

translations/translator/xt_capts These three packages are intended for programmers of LaTeX packages to internationalize fixed text elements such as the headings of tables of contents. These packages are only of limited use for authors of multilingual documents.

1.2 Languages

Das Konzept von langselect beinhaltet die Nutzung von Sprachen auf verschiedenen Ebenen mit unterschiedlicher Wirkung. Um Verwechselungen zu vermeiden, wollen wir hier die Begriffe eindeutig definieren.

1. The set of *possible* target languages for the document. A document that uses langselect typically contains sections in different target languages. The possible target languages are set via a package option, see Section 2.2. In the rest of this documentation, we refer to the possible target languages as *selectable languages*, since one of these languages is selected for the LaTeX translation.

2. on CTAN as `multilang`: `http://mirrors.ctan.org/macros/latex/contrib/multilang/`
3. on CTAN as `comment`: `http://mirrors.ctan.org/macros/latex/contrib/comment/`

2. The actual target language used in a LaTeX translation. This must be one of the languages from 1. Typically, the translation process is performed separately for each of the actual target languages. There are several methods for selecting the actual target language; see section 2.1.

3. The language in which text fragments or individual words are actually written. This is the concept of language used by packages such as Babel or Polyglossia.[4] Please note that a multilingual document from the perspective of langselect can be a monolingual document from the perspective of Babel. However, additional languages may well be added: For example, a document with the selected languages German and English may contain Latin words or sentences in the text. This selection is made using the usual Babel mechanisms such as `\selectlanguage`, `\foreignlanguage` or `\text`⟨*language*⟩. When Babel or Polyglossia is loaded, which can also be done directly via langselect if desired, the target language is selected as the active language.

Languages are designated in langselectusing ISO 639-1 codes (”‘english”’, ”‘french”’, ”‘ngerman”’ …). If a specific dialect or variant is desired for language processing at the Babel/Polyglossia level, mapping can be performed.

2 Usage

The package is loaded in the usual way:

`\usepackage[`⟨*options*⟩`]{langselect}`

Since the target language can be selected both via options and in other ways, these different methods are described here first. Further options, such as setting the selectable languages, are discussed in Section 2.2.

2.1 Selection of the Target Language

There are various ways to determine a target language for a LaTeX translation run. The order in the following list also reflects the order of evaluation. Once a target language has been determined, further evaluation is canceled.

1. Definition of a macro `\omdTargetLanguage` *before* loading the package. This makes it possible, for example, to use

   ```
   > latexmk -e"\def\olsTargetLanguage{fr}" document.tex
   ```

   to set the target language French for the translation run.

2. Setting of the packet option `targetlang` = `{`⟨*target language*⟩`}`

3. If the option `targetlang` = `{job=`⟨n⟩`}` is set, the target language is determined from the nth element of the job name. An element is a part of the job name separated by “‘-”’

4. However, in these packages, ”‘language”’ also refers to several different and, in some cases, independent concepts such as hyphenation patterns, fonts to be used, etc.

(minus), see the documentation for the package varsfromjobname,[5] which is loaded for this purpose.For example, with `targetlang = {job=2}` one can set Russian as target language using

```
> latex -jobname doc-ru doc.tex
```

4. Setting `targetlang = {meta}` uses the language specified in `\DocumentMetaData` as the target language.

5. If everything else fails, English is set as the target language.

2.2 Options

Es können folgende Optionen gesetzt werden:

`languages = {`⟨*list of selectable languages*⟩`}` (required)
Provides two or three selectable languages. The order matters for the generation of the language commands (cf. Section 3).

`targetlang = `⟨*language*⟩`|{job=`⟨*n*⟩`}|meta`
See Section 2.1

`prefix = {`⟨*prefix*⟩`}` Default: `l`
Sets a prefix for multilingual macros. An empty prefix can also be set.

`auto = true|false` Default: `true`
The actual language macros are created, see Section 3. If `auto = false` is set, only the macros for generating the language macros are provided, not the language macros themselves.

3 User Macros

If you have not set the option `auto = false`, a multilanguage macro will be created. Its names is determined by the list of selection languages. Depending on whether you have specified two or more selection languages, the name is

- `\`⟨*prefix*⟩⟨*language*$_1$⟩⟨*language*$_2$⟩`{`⟨*text*$_1$⟩`}{`⟨*text*$_2$⟩`}`

or

- `\`⟨*prefix*⟩⟨*language*$_1$⟩⟨*language*$_2$⟩⟨*language*$_3$⟩`{`⟨*text*$_1$⟩`}{`⟨*text*$_2$⟩`}{`⟨*text*$_3$⟩`}`

For example, for this documentation, the package was loaded with the following options:

```
\usepackage[
languages={de,en},
targetlang={job=2}
]{langselect}
```

5. on CTAN as `varsfromjobname`: `http://mirrors.ctan.org/macros/latex/contrib/varsfromjobname/`

This makes the macro `\ldeen` available.

If the target language "'de"' is now selected (here via the job name, but there are also other options, see Section @1), only ⟨*Text*$_1$⟩ is transferred to the PDF document and ⟨*Text*$_2$⟩ is discarded. With the target language "'de"', it is the other way around.2.1

Attention!
The automatic generation of language macros can sometimes cause problems. For example, Abkhaz (code: "'ab"') as the first language and Greek (Code: "'el"') as the second language results in the automatic name `\label`, which leads to error messages. In such cases, use the features described in section 3.2.

3.1 Stared Variant

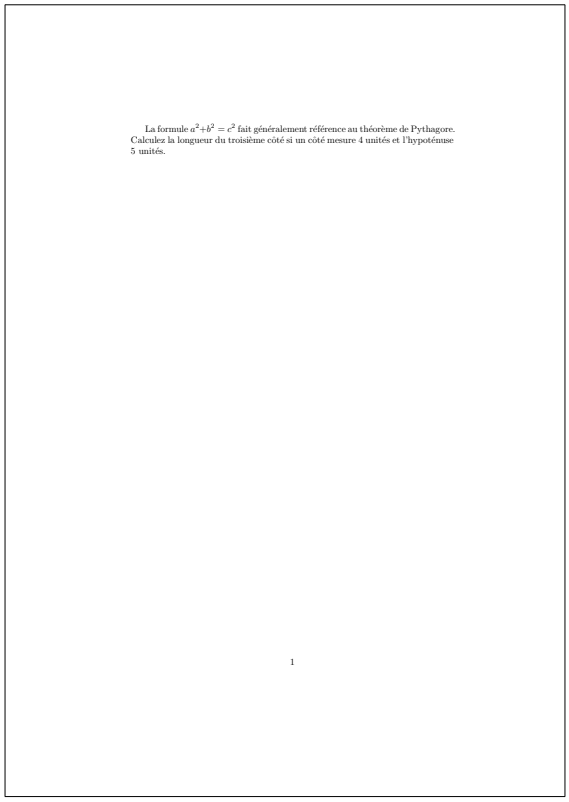
A common problem with multiple language versions is the consistency of non-language-dependent data, especially if this data changes frequently (think, for example, of timetables).

For such cases, langselect provides a star variant of the language macros. In this star variant, the text arguments can contain placeholders of the form "'@⟨*n*⟩"' can be used, where ⟨*n*⟩ represents a value between 1 and 9.

Additional arguments must then be specified after the macro, namely as many as correspond to the highest ⟨*n*⟩ used. The star variant replaces the placeholders with the corresponding arguments.

```
\documentclass{article}
\usepackage[languages={en,fr},
targetlang=fr]{langselect}

\begin{document}
\lenfr*{The formula @1 generally refers to the Pythagorean theorem.}{La
  formule @1 fait généralement référence au théorème de
  Pythagore.}{$a^{2}+b^{2}=c^{2}$}
\lenfr*{Calculate the length of the third side if one side is @1 units long
  and the hypotenuse is @2 units long.}{Calculez la longueur du troisième
  côté si un côté mesure @1 unités et l'hypoténuse @2 unités.}{$4$}{$5$}
\end{document}
```



3.2 Individual Language Macros

If you do not like the generated names of the language macros or if they even lead to collisions, you can change them. The easiest way is to change the package option `prefix` so that, for example, with `prefix` = `mv`, the language macro for the German/English combination is called `\mvdeen`.

4 Examples

5 Implementation

```
\NeedsTeXFormat{LaTeX2e}[2022/06/01]
\def\packagename{langselect}
\def\packageversion{2025/08/10 v0.9.0d}
\ProvidesPackage{\packagename}[\packageversion\space support for different
languages versions of a document]
\ExplSyntaxOn
```

`\olsIsoTags` contains all valid ISO 639-1 code tags.

```
\seq_const_from_clist:cn{olsIsoTags}{aa,ab,ae,af,ak,am,an,ar,as,av,ay,az,ba,be,bg,
  bh,bi,bm,bn,bo,br,bs,ca,ce,ch,co,cr,cs,cu,cv,cy,da,de,dv,dz,ee,el,en,eo,es,et,eu,
  fa,ff,fi,fj,fo,fr,fy,ga,gd,gl,gn,gu,gv,ha,he,hi,ho,hr,ht,hu,hy,hz,ia,id,ie,ig,ii,
  ik,io,is,it,iu,ja,jv,ka,kg,ki,kj,kk,kl,km,kn,ko,kr,ks,ku,kv,kw,ky,la,lb,lg,li,ln,
  lo,lt,lu,lv,mg,mh,mi,mk,ml,mn,mr,ms,mt,my,na,nb,nd,ne,ng,nl,nn,no,nr,nv,ny,oc,oj,
  om,or,os,pa,pi,pl,ps,pt,qu,rm,rn,ro,ru,rw,sa,sc,sd,se,sg,si,sk,sl,sm,sn,so,sq,sr,
  ss,st,su,sv,sw,ta,te,tg,th,ti,tk,tl,tn,to,tr,ts,tt,tw,ty,ug,uk,ur,uz,ve,vi,vo,wa,
  wo,xh,yi,yo,za,zh,zu}

\newcommand\IfTagIsValidF[2]{%
  \seq_if_in:ceF{olsIsoTags}{#1}{#2}
}
\RequirePackage{varsfromjobname}[2025/08/03]
```

The results of `\varsfromjobname` and `\str_range:Nnn` have catcode 12, but ISO tags and options have catcode 11. We standardize using a wrapper on catcode 11.

```
\newcommand\olsChangeCatEleven[1]{
  \def\tmpa{#1}
  \edef\tmpb{
    \noexpand\scantokens{
      \noexpand\edef\noexpand\olsCatElven{\tmpa}
    }
  }
  \tmpb
}
```

There are only a handful of options.

```
\DeclareKeys{
  languages.clist_gset:N = \ols_langs,
  languages.usage=load,
  targetlang.code = {
    \cs_if_exist:NF\olsTargetLanguage{
      \str_if_eq:eeTF{\str_range:Nnn{#1}{1}{3}}{job}{%
        \olsChangeCatEleven{\getfromjobname{\str_range:Nnn{#1}{-1}{-1}}}
        \edef\olsTargetLanguage{\olsCatElven}
      }{
        \str_if_eq:nnTF{#1}{meta}{
          \IfDocumentMetadataTF{
            \edef\tmpa{\GetDocumentProperties{document/lang}}
            \edef\tmpb{\str_range:Nnn\tmpa{1}{2}}
            \olsChangeCatEleven{\tmpb}
            \edef\olsTargetLanguage{\olsCatElven}
          }{
```

```
            \PackageWarningNoLine{\packagename}{No~meta~data~provided,
              ~falling~back~to~English}
            \def\olsTargetLanguage{en}
          }
        }{
          \edef\olsTargetLanguage{#1}
        }
      }
    }{}
  },
  targetlang.usage=load,
  prefix.store=\ols_sprefix,
  prefix.initial:n=l,
  prefix.usage=load,
  auto.bool_set:N=\ols_generate,
  auto.usage=load,
  auto.initial:n=true
}%
\ProcessKeyOptions\relax
\cs_if_exist:NF \olsTargetLanguage {\def\olsTargetLanguage{en}}
```

We are doing some checks. First, we check whether a valid target language could be determined.

```
\IfTagIsValidF{\olsTargetLanguage}{
  \PackageWarningNoLine{\packagename}{Can't~resolve~target~language~'\olsTargetLanguage'.
    \MessageBreak Fall~back~to~'en'}
  \def\olsTargetLanguage{en}
}
```

Are at least two selectable languages specified?

```
\int_compare:nNnT{\clist_count:N{\ols_langs}} < {2}{
  \PackageError{\packagename}{No~sufficient~number~of~selectable
    \MessageBreak~languages~provided}{
    You~have~to~provide~a~list~of~two~or~three\MessageBreak
    languages~via~'language'~option.}
  \aftergroup\endinput
}
```

Currently, langselect can't process more than three languages. Thus, we issue a warning if the list of selectable languages contrains more than three languages.

```
\int_compare:nNnT{\clist_count:N\ols_langs} > {3}{
  \PackageWarningNoLine{\packagename}{Too~many~selectable~languages~provided.
    \MessageBreak I~will~ignore~the~superflous~languages}
}
```

Next, we test the validity of the selectable languages. They should form ISO 693-1 codes.

```
\clist_map_inline:Nn\ols_langs {
  %\typeout{***~CHECK~'#1'}
  \IfTagIsValidF{#1}{
    \PackageError{\packagename}{Couldn't~resolve~selectable~language~'#1'}{
      Use~valid~ISO-639-1~code~in~option~'languages'.}
    \aftergroup\endinput
  }
}
```

The list of selectable languages should include the target language.

```
\clist_if_in:NVF\ols_langs{\olsTargetLanguage}{
  \PackageError{\packagename}{Target~language~'\olsTargetLanguage'~is~not\MessageBreak
    in~the~list~of~selectable~languages}{
    Check~option~'language'~and~'targetlang'.}
}
```

If needed, we correct the language metadata set by `\DocumentMetadata`. Now, Babel (for example) does not derive a wrong main language.

```
\IfDocumentMetadataT{
  \edef\olsDocLang{\exp_args:Ne\str_range:nnn{\GetDocumentProperties{document/lang}}{1}{2}}
  \str_if_eq:eeF{\olsTargetLanguage}{\olsDocLang}{
    \PackageWarningNoLine{\packagename}{Target~language~'\olsTargetLanguage'~
      doesn't~comply~with\MessageBreak \c_backslash_str
      DocumentMetadata~'\GetDocumentProperties{document/lang}'.\MessageBreak
      I'll~try~to~overwrite~the~meta~data}
    \AddToDocumentProperties[document]{lang}{\olsTargetLanguage}
  }
}
```

A Boolean to indicate whether more than two selectable languages have been specified.

```
\bool_new:c{ols_trilang}
\int_compare:nNnTF{\clist_count:N{\ols_langs}} > {2}{
  \bool_set_true:c{ols_trilang}
}{
  \bool_set_false:c{ols_trilang}
}
\def\olsFrstLanguage{\clist_item:Nn \ols_langs{1}}
\def\olsScndLanguage{\clist_item:Nn \ols_langs{2}}
\bool_if:cT{ols_trilang}{
  \def\olsThrdLanguage{\clist_item:Nn \ols_langs{3}}
}
```

The commands \olsMakeBilangualMacro and \olsMakeTrilangualMacro, resp., are used to generate the actual language macros.

```
 % #1: prefix, #2,#3: selectable languages, #4: own name
\NewDocumentCommand{\olsMakeBilangualMacro}{O{\ols_sprefix} m m o}{
  \str_if_eq:eeT{\olsTargetLanguage}{#2} {
    \IfValueTF{#4}{
      \ExpandArgs{c}\NewDocumentCommand{#1#4}{s +m +m}{
        \IfBooleanTF{##1}{
          \olsProcessArgs{##2}
        }{
          ##2
        }
      }
    }{
      \ExpandArgs{c}\NewDocumentCommand{#1#2#3}{s +m +m}{
        \IfBooleanTF{##1}{
          \olsProcessArgs{##2}
        }{
          ##2
        }
      }
    }
  }
  \str_if_eq:eeT{\olsTargetLanguage}{#3}{
    \IfValueTF{#4}{
      \ExpandArgs{c}\NewDocumentCommand{#1#4}{s +m +m}{
        \IfBooleanTF{##1}{
          \olsProcessArgs{##3}
        }{
          ##3
        }
      }
    }{
      \ExpandArgs{c}\NewDocumentCommand{#1#2#3}{s +m +m}{
        \IfBooleanTF{##1}{
          \olsProcessArgs{##3}
        }{
          ##3
        }
      }
    }
  }
}
```

Now do the same thing again for three languages.

```
\NewDocumentCommand{\olsMakeTrilangualMacro}{O{\ols_sprefix} m m m o}{
  \str_if_eq:eeT{\olsTargetLanguage}{#2} {
    \IfValueTF{#4}{
      \ExpandArgs{c}\NewDocumentCommand{#1#5}{s +m +m +m}{
        \IfBooleanTF{##1}{
          \olsProcessArgs{##2}
        }{
          ##2
        }
      }
    }{
      \ExpandArgs{c}\NewDocumentCommand{#1#2#3}{s +m +m +m}{
        \IfBooleanTF{##1}{
          \olsProcessArgs{##2}
        }{
          ##2
        }
      }
    }
  }
  \str_if_eq:eeT{\olsTargetLanguage}{#3} {
    \IfValueTF{#4}{
      \ExpandArgs{c}\NewDocumentCommand{#1#5}{s +m +m +m}{
        \IfBooleanTF{##1}{
          \olsProcessArgs{##2}
        }{
          ##3
        }
      }
    }{
      \ExpandArgs{c}\NewDocumentCommand{#1#2#3}{s +m +m +m}{
        \IfBooleanTF{##1}{
          \olsProcessArgs{##2}
        }{
          ##3
        }
      }
    }
  }
  \str_if_eq:eeT{\olsTargetLanguage}{#4} {
    \IfValueTF{#4}{
      \ExpandArgs{c}\NewDocumentCommand{#1#5}{s +m +m +m}{
```

```
        \IfBooleanTF{##1}{
          \olsProcessArgs{##2}
        }{
          ##4
        }
      }
    }{
      \ExpandArgs{c}\NewDocumentCommand{#1#2#3}{s +m +m +m}{
        \IfBooleanTF{##1}{
          \olsProcessArgs{##2}
        }{
          ##4
        }
      }
    }
  }
}
```

If automatic generation is not disabled, the specific language macros are created. The names are \⟨*prefix*⟩⟨*language*$_1$⟩⟨*Language*$_2$⟩ or \⟨*prefix*⟩⟨*language*$_1$⟩⟨*language*$_2$⟩⟨*language*$_3$⟩.

```
\bool_if:cTF{ols_generate}{
  \bool_if:cTF{ols_trilang}{
    \olsMakeTrilangualMacro{\olsFrstLanguage}{\olsScndLanguage}{\olsThrdLanguage}
  }{
    \olsMakeBilangualMacro{\olsFrstLanguage}{\olsScndLanguage}
  }
}{
  \PackageInfoNoLine{\packagename}{No~language~macros~generated.\MessageBreak
  Use~\string\olsMakeBilangualMacro~or~\string\olsMakeTrilangualMacro~to~generate.}
}
```

For the star variants of the language commands, we create a command that processes the subsequent arguments.

```
\seq_new:N \ols_args_seq
\int_new:N \ols_num_of_args
\tl_new:N \ols_format_str_tl
```

Since the placeholder character @ appears as a literal in the following code, it must have the same catcode as in the document.

```
\makeatother
\NewDocumentCommand \olsProcessArgs { +m } {
  \tl_gset:Nn \ols_format_str_tl{#1}
  \int_zero:N \ols_num_of_args
```

Find the highest value n, where @n is contained in the format string.

```
  \int_step_inline:nnnn { 9 } { -1 } { 1 }
  {
    \regex_if_match:nnT {@##1}{ #1 }
    {
      \int_set:Nn \ols_num_of_args { ##1 } \prg_break:
    }
  }
```

If no placeholder character is found, output the text in original form. Otherwise, the replacement macro is called.

```
  \int_compare:nNnTF {\ols_num_of_args} = { 0 }
  { #1 }{
    \seq_clear:N \ols_args_seq
    \ols_collect_replace:n \ols_num_of_args
  }
}
```

`\ols_collect_replace` receives *one* argument, namely the number of remaining replacement texts. However, it consumes *two* arguments. It obtains the second one from the token stream following the macro, where it finds the next replacement text. This is stored in a token sequence and `\ols_collect_replace` is called again.

```
\cs_new_protected:Npn \ols_collect_replace:n #1 #2{
  \seq_put_right:Nn \ols_args_seq { #2 }
  \int_compare:nNnTF { #1 } = { 1 }
  {
```

Once all replacement texts have been read in, they are applied one after the other to the placeholder expressions.

```
    \int_step_inline:nn { \ols_num_of_args }
    {
      \tl_replace_all:Nnn \ols_format_str_tl
      { @##1 }
      { \seq_item:Nn \ols_args_seq { ##1 } }
    }
    \ols_format_str_tl
  }{
    \ols_collect_replace:n { \int_eval:n { #1 - 1 } }
  }
}
\makeatletter
\ExplSyntaxOff
```

——————— Ende Paketcode ——————————-

6 Anwendung

Laden und verwenden:

```
\usepackage{mypkg}
```

Das war’s.