The Polyglot Package*

Javier Bezos-López†

September 1, 1997

Notice to Users of Version 1.0

I'm afraid some user commands have been changed, specially `\selectlanguage` and `\uselanguage`, and some other supressed—`\enablegroup` and `\disablegroup`, which have been merged into `\selectlanguage`. These changes will be definitive, and I had to do them in order to implement a right communication to files and marks, and avoid mixing up definitions which sometimes made LaTeX enter in an endless loop. Furthermore, the new syntax is far more robust, flexible and readable.

Most of commands for description files haven't changed at all, though some of them has been reimplemented. Yet, handling of dialects has been much improved; there is a new `\SetLanguage` (the old one has been renamed to `\SetDialect`) and you can create a dialect at any time with `\DeclareDialect` without the restrictions of the now obsolete `\GenerateDialects`.

1 Introduction

The package polyglot provides a new language selection system taking advantage of the features of LaTeX 2_ε. It provides some utilities which make writing a language style quite easy and straightforward.

Some of the features implemeted by polyglot are:

- You can switch between languages freely. You have not to take care of neither head lines nor toc and bib entries—the right language is always used.[1] You can even get just a few commands from a language, not all.
- "Ligatures," which are shortcuts for diacritical marks; for instance, in French `^e` is a synonymous to `\^{e}`. Some other ligatures perform special actions, as Spanish `'r` which allows divide "extrarradio" as "extra-radio". A very cool feature: in most of cases, ligatures does not interfere with other definitions; for instance, with the index package `^{`*entry*`}` still works, provided the *entry* has two or more tokens.[2]
- Dialects—small language variants—are supported. For instance American is a variant of English with another date format. Hyphenations patterns are attached to dialects and different rules for US and British English can be used.
- New glyphs are created if necessary. For instance, if you are using `OT1` the German umlauts are lowered, but if you switch to `T1` the actual font glyphs are used. Some other font encoding may have their own glyphs which will be used only with that encoding.

*This package is currently at version 1.1.

†Please, send your comments and suggestions to `jbezos@mx3.redestb.es`, or to my postal address: Apartado 116.035, E-28080 Madrid, Spain. English is not my strong point, so contact me when you find mistakes in the manual.

[1] Index entries follow a special syntax and the current version of polyglot cannot handle that. For this reason the index entries remain untouched and you may use language commands only to some extent.

[2] And provided index is loaded before polyglot. index is a package by David Jones.

- Customization is quite easy—just redefine a command of a language with `\renewcommand` when the language is in force. The new definition will be remembered, even if you switch back and forth between languages.
- An unique layout can be used through the document, even with lots of languages. If you use a class with a certain layout you don't want to modify, you or the class can tell polyglot not to touch it at all.

2 Quick start

2.1 Installation

To install the package follow these steps:

1. First, run `polyglot.ins`. The files `polyglot.sty`, `polyglot.ltx`, `polyglot.def` and `polyglot.cfg` are generated.
2. Remove `polyglot.ltx` and `polyglot.def` as they are not needed in the basic installation.
3. Move `polyglot.sty` and `polyglot.cfg` as well as the files with extensions `.ld` and `.ot1` to a place where LaTeX can find them.

The files are:

- `polyglot.sty` is the file to be read with `\usepackage`.
- `polyglot.cfg` gives your system configuration.
- Languages are stored in files with extension `.ld`, as, for instance, `spanish.ld`, `english.ld` and so on.
- `polyglot.ltx` has some definitions for instalation of hyphenation patterns as well as preloading of languages and the polyglot style (actually `polyglot.def`).
- Finally, there are some files named like languages, but with extensions like font encodings.

Once installed, you can use polyglot. To write a, say, German document simply state the `german` option in the `\documentclass` and load the package with `\usepackage{polyglot}`. That's all. A new layout, with new commands, ligatures and, if the `OT1` encoding is in force, new glyphs will be available to you, as described at the end of this manual. If you are happy with that you need not go further; but if you are interested in advanced features (how to insert a Spanish date or how to create your own ligatures, for instance) just continue.

3 User Interface

3.1 Groups

A language has a series of commands and variables (counters and lengths) stored in several groups. These groups are:

names Translation of commands for titles, etc., following the international LaTeX conventions.

layout Commands and variables for the general layout of the document (`enumerate`, `itemize`, etc.)

date Logically, commands concerning dates.

ligatures A way to provide ligatures, i.e., shorthands for accented letters, and other special symbols.

text Modifications of some typografical conventions: spacing before o after characters (French ':' or ';', Spanish '%'), etc.

tools Supplemetary command definitions.

These groups belong to two categories: names, layout, and date are considered *document* groups, since they are intended for formatting the document; ligatures, tools and text, are considered *text* groups, since you will want to use them temporarily for a short text in some language.

3.2 Selecting a Language

You must load languages with `\usepackage`:

```
\usepackage[english,spanish]{polyglot}
```

Global options (those of `\documentclass`) will be recognized as usual. The very first language will be automatically selected (with the starred version, see below). For this purpose, class options are taken in consideration *before* package options. (Perhaps this is not the usual convention in LaTeX 2_ε, but it is more logical; in general, you will state the main language as class option and the secondary ones as package options.) You can cancel this automatic selection with the package option `loadonly`:

```
\usepackage[german,spanish,loadonly]{polyglot}
```

`\selectlanguage*[`*no-groups*`]{`*language*`}`

Selects all groups from *language*, except if there is the optional argument. *no-groups* is a list of groups *not* to be selected, in the form `no`*group*`,no`*group*`,...`. For instance, if you dislike using ligatures:

```
\selectlanguage*[noligatures]{french}
```

This command also sets defaults to be used by the unstarred version (see below).

`\selectlanguage[`*groups*`]{`*language*`}`

Where *groups* is a list of *group*s and/or `no`*group*s.
There are three posibilities:

- When a group is cited in the form *group* (e.g., `date` or `names`), this group is selected for the current language, i.e., that of the current `\selectlanguage`.
- When a group is cited in the form `no`*group* (e.g., `nodate` or `nonames`), this group is cancelled.
- When a group is not cited, then the default, i.e., that set by the `\selectlanguage*` in force, is used; it can be a `no`-form, and then the group will be disabled if necessary. If there is no a previous starred selection, the `no`-form will be presumed in all of groups.

If no optional argument is given, then `text,ligatures,tools` are presumed. Thus, at most two languages are selected at time.
Here is an example:

```
\selectlanguage*[noligatures]{spanish}
Now we have Spanish names, date, layout, text and tools,
but no ligatures at all.
\selectlanguage[date,notools]{french}
Now we have Spanish names, layout and text, French date,
but neither ligatures nor tools.
\selectlanguage[ligatures]{german}
Now we have Spanish names, date, layout, text and tools,
and German ligatures.
```

Selection is always local. There is also the possibility to use `selectlanguage` as environment.

```
\begin{selectlanguage}*[no-groups]{language} text \end{selectlanguage}
\begin{selectlanguage}[groups]{language} text \end{selectlanguage}
```

In general, you will use these commands without the optional argument—the starred version as command at the very beginning of documents and the starless version as environment in a temporary change of language.

For the sake of clarity, spaces are ignored in the optional argument, so that you can write

```
\selectlanguage[date, tools, no ligatures]{spanish}
```

`\uselanguage[`*groups*`]{`*language*`}{`*text*`}`

A command version of the `selectlanguage` environment, although only the unstarred version is allowed.

`\unselectlanguage`

You can switch all of groups off with `\unselectlanguage`. Thus, you return to the original LaTeX as far as polyglot is concerned.[3]

A typical file will look like this

```
\documentclass[german]{...}
\usepackage[spanish]{polyglot}
\newenvironment{spanish}%
  {\begin{quote}\begin{selectlanguage}{spanish}}%
  {\end{selectlanguage}\end{quote}}
\begin{document}
Deutsche text
\begin{spanish}
  Texto en espa'nol
\end{spanish}
Deutsche text
\end{document}
```

Note that `\selectlanguage*{german}` is not necessary, since it is selected by the package. (Because there is no `loadonly` package option.)

3.3 Tools

`\thelanguage`

The name of the current language. You *cannot* redefine this command in a document.

[3] Well, not exactly. But you should not notice it at all.

`\languagelist`

A list of languages requested.

`\allowhyphens`

Allows further hyphenation in words with the primitive `\accent`.

`\hexnumber` `\octalnumber` `\charnumber`

Synonymous to `"`, `'` and `` ` `` for numbers (`\hexnumber F3` = `"F3`) provided for convenience.

`\nofiles`

Not a new command really, but it has been reimplemented to optimize some internal macros related with file writing.

`\ensurelanguage`

The polyglot package modifies internally some LaTeX commands in order to do its best for making sure the current language is used in a head/foot line, even if the page is shipped out when another language is in force. Take for instance

```
(With \selectlanguage*{spanish})
\section{Sobre la confecci'on de t'itulos}
```

In this case, if the page is broken inside, say, a German text `\ensurelanguage` restores `spanish` and hence the accents.

Yet, some non-standard classes or packages can modify the marks. Most of times (but not always) `\ensurelanguage` solves the problem: [4]

```
(With \selectlanguage*{spanish})
\section{\ensurelanguage Sobre la confecci'on de t'itulos}
```

In typeset, writing and other modes it's ignored.

`\ensuredcommand{`*cmd*`}{`*text*`}`

Saves the *text* in *cmd* so that you can use it in a "ensured" form later. Neither `\selectlanguage` nor `\uselanguage` can be passed in an argument—you must save the text with this command and then release it. The language is the current one. No arguments are allowed, and *text* is fragile, but a construction like `\uselanguage{...}{\ensuredcommand...}` is valid.

Spanish `\chaptername` uses a `\'i` which is not set by standard `OT1` and hence is provided by `spanish.ot1`. If you use `\chaptername` in a text with, say, the German `text` group you will get an accented dotted i. In this case, you can use `\ensuredcommand`.

3.4 Extensions

The polyglot package provides *extensions* so that its functionality will be extended to and from some package with the help of some commands in the `polyglot.cfg` file, sometimes with automatic loading. At the current moment, only font enconding extensions are implemented,

[4] For instance, it will not work when the line is automatically capitalized.

which are automatically taken care of. (In future releases, you will be allowed to by-pass the current default settings, and add further extensions.)

3.4.1 Font Encoding Extensions

With the help of this extension, you are allowed to change some commands depending of font encodings. When a language is loaded, the package reads automatically the files named *language-file*`.`*ENC*, if they exist, where *language-file* is the exact name of the file `.ld` which the language is defined in, and *ENC* is the name of encodings declared. So, for instance, if you have preinstalled `T1` (besides `OMS`, etc.) and:

```
\documentclass{article}
\usepackage[LXX,OT1]{fontenc}
\usepackage[spanish,german]{polyglot}
```

the following files will be read *if they exist* (if not, nothing happens):

```
spanish.t1   spanish.ot1  spanish.lxx  spanish.oms  etc.
 german.t1    german.ot1   german.lxx   german.oms  etc.
```

So, for example, `german.ot1` redefines some umlauted vowels in order to modify the accent position and to allow hyphenation.

Loading of these files may be inhibited by means of the option `nofontenc` when calling polyglot:

```
\usepackage[spanish,german,nofontenc]{polyglot}
```

4 Developpers Commands

Some command names could seem inconsistent with that of the user commands. In particular, when you refer to a language in a document you are referring in fact to a dialect, which belogs to a language. As user, you cannot access to a language and you instead access to a dialect named like the language. From now on, when we say "current language" we mean "current language or dialect." For more details on dialects, see below.

4.1 General

`\DeclareLanguage{`*language*`}`

The first command in the `.ld` must be this one. Files don't always have the same name as the language, so this command makes things work.

`\DeclareLanguageCommand{`*cmd-name*`}{`*group*`}[`*num-arg*`][`*default*`]{`*definition*`}`

Stores a command in a group of the current language. The definition will be activated when the group is selected, and the old definition, if any, will be stored for later recovery if the group is switched off.

There is a point to note (which applies also to the next commands). Suppose the following code:

At `spanish.ld`:

```
\DeclareLanguageCommand{\partname}{names}{Parte}
```

At document:

```
\selectlanguage*{spanish}
\renewcommand{\partname}{Libro}
\selectlanguage*{english}
\partname
```

Obviously, at this point `\partname` is 'Part'. But if you follow with

```
\selectlanguage*{spanish}
\partname
```

Surprise! *Your* value of `\partname`, i.e., 'Libro', is recovered. So you can customize easily these macros in your document, even if you switch back and forth between languages.

`\SetLanguageVariable{`*var-name*`}{`*group*`}{`*value*`}`

Here *var-name* stands for the internal name of a counter or a lenght as defined by `\newconter` (`\c@...`) or `\newlenght`. The variable must be already defined. When the group is selected, the new value will be assigned to the variable, and the old one will be stored.

`\SetLanguageCode{`*code-cmd*`}{`*group*`}{`*char-num*`}{`*value*`}`

Similar to `\SetLanguageVariable` but for codes. For instance:

```
\SetLanguageCode{\sfcode}{text}{`.}{1000}
```

Languages with `\frenchspacing` should set the `\sfcodes` with this command, so that a change with `\nonfrenchspacing` is recovered after a switch.

`\DeclareLanguageSymbolCommand{`*char*`}{`*group*`}[`*num-arg*`][`*default*`]{`*definition*`}`

First makes *char* active and then defines it just as `\DeclareLanguageCommand` does. For instance, in French:

```
\DeclareLanguageSymbolCommand{:}{spacing}{\unskip\,:}
```

Note that this command *does not* change the category yet, and hence the previous command is valid. (Well, except if the `:` is already active. If you want to avoid any infinite loop, you can just say `{\unskip\,\string:}`).

`\UpdateSpecial{`*char*`}`

Updates `\dospecials` and `\@sanitize`. First removes *char* from both lists; then adds it if it has categorie other than 'other' or 'letter'. With this method we avoid duplicated entries, as well as removing a character being usually special (for instance `~`).

4.2 Groups

`\DeclareLanguageGroup{`*group*`}`
`\DeclareLanguageGroup*{`*group*`}`

Adds a new group. With the starred version, the group will be considered a `text` group, and hence included in the default of `\selectlanguage`. Group names cannot begin with `no` because of the `no`-form convention given above.

4.3 Ligatures

`\DeclareLigatureCommand{`*char-1*`}{`*char-2*`}{`*definition*`}`

Makes the pair *char-1 char-2* a shorthand for *definition*. For instance:

```
\DeclareLigatureCommand{"}{u}{\"u}
```

There are some points to note:

- Spaces between *char-1* and *char-2* are not significant. So `"u` and `"␣u` will give the same result. Be careful then.
- If *char-1* is followed by a char which there is no ligature for, the original value (i.e., the value of *char-1* at `\selectlanguage`) is used.
- If *char-1* is followed by an "argument" which is empty or with more than one token, its original value is also used. This is very important in order to break ligatures. In German, you get '"und' with `"{}und` or `"{und}`; in French, you get 'C'est' with `C'{}est` or `C'{est}`; in Spanish, you write a non-breaking space before 'n' with `~{}n`, and before 'ñ', with `~{~n}` or `~~n`. If some package (there are in fact a lot) has defined, say, `^` with some special function which requires an argument, this will be keept in most of cases (multi-token arguments), even when we define `^` as a ligature; if the argument has only a token you may trick the `^` with, say, `^{e{}}` (two tokens, 'e' and empty). In math mode, the original math meaning is used (even if the `\mathcode` was `"8000`).
- Some languages, such as Spanish, make active `<` and `>` in order to declare the ligatures `<<` and `>>` for guillemets. If you define macros testing some counters or lengths are lesser or greater, load the package with option `loadonly` and select the language after these commands.
- Any definitionn attached to a ligature char is preserved, even if not currently 'active'. This makes switching a language very robust.[5]
- Yet, an error is given when a ligature char has certain character codes at the selection, namely 'escape' (`\`), 'open' (`{`), 'close' (`}`), 'return' (`^^M`) or 'comment' (`%`). This is a very unlikely situation and for the moment there is nothing I can do. Furthermore, 'invalid' chars are validated (as 'other') in the scope of the language.

`\DeclareLigature{`*char-1*`}`

In some instances, you might wish to declare a ligature char before `\DeclareLigatureCommand` (which has an implicit `\DeclareLigature`). (I wonder when, but here it is.)

4.4 Dates

`\DeclareDateFunction{`*date-function-name*`}{`*definition*`}`
`\DeclareDateFunctionDefault{`*date-function-name*`}{`*definition*`}`
`\DeclareDateCommand{`*cmd-name*`}{`*definition*`}`

By means of `\DeclareDateCommand` you can define commands like `\today`. The good news is that a special syntax is allowed in *definition* with date functions called with `<`*date-funtion-name*`>`. Here `<`*date-funtion-name*`>` stands for the definition given in `\DeclareDateFunction` for the current language. If no such function for the language is given then the definition of `\DeclareDateFunctionDefault` is used. See `english.ld` for a very illustrative example. (The `<` and `>` are actual lesser and greater signs.)

[5] Don't try to change the catcode of a char to 'other' before selecting a language in order to 'lost' its definition—that won't work. Instead, let it to relax if you really need it. (In fact, polyglot uses three internal variables, the first storing the text value, the second the math value, and the third the definition, which is used when the catcode was 'active' or the mathcode was `"8000`.)

Predeclared functions (with `\DeclareDateFunctionDefault`) are:

- `<d>` one or two digits day: 1, 2, . . . , 30, 31.
- `<dd>` two digits day: 01, 02, . . .
- `<m>` one or two digits month.
- `<mm>` two digits month.
- `<yy>` two digits year: 96, 97, 98, . . .
- `<yyyy>` four digits year: 1996, 1997, 1998, . . .

Functions which are not predeclared, and hence should be declared by the `.ld` file, are:

- `<www>` short weekday: mon., tue., wes., . . .
- `<wwww>` weekday in full: Monday, Tuesday, . . .
- `<mmm>` short month: jan., feb., mar., . . .
- `<mmmm>` month in full: January, February, . . .

The counter `\weekday` (also `\value{weekday}`) gives a number between 1 and 7 for Sunday, Monday, etc.

For instance:

```
\DeclareDateFunction{wwww}{\ifcase\weekday\or Sunday\or Monday\or
  Tuesday\or Wesneday\or Thursday\or Friday\or Saturday\fi}
\DeclareDateCommand{\weektoday}{<wwww>, <mmmm> <dd> <yyyy>}
```

4.5 Dialects

As stated above, `\selectlanguage` access to dialects rather than languages.
`\DeclareLanguage` declares both a language and a dialect with the same name, and selects the actual language.

`\DeclareDialect{`*dialect*`}`
`\SetDialect{`*dialect*`}`
`\SetLanguage{`*language*`}`

`\DeclareDialect` declares a dialect, which shares all of declarations for the current actual language. With `\SetDialect` you set the dialect so that new declarations will belong only to this dialect. `\DeclareDialect` just declares but does not set it.

A dialect with the same name as the language is always implicit. You can handle this dialect exactly as any other dialect. In other words, after setting the dialect, new declarations belong only to this one. If you want to return to the actual language, so that new declarations will be shared by all dialects, use `\SetLanguage`.

Note that commands and variables declared for a language are set by `\selectlanguage` before those of dialects, doesn't matter the order you declared it.

For example:

```
\DeclareLanguage{english}
\DeclareDialect{american}
Declarations
\SetDialect{english}
\DeclareDateCommmand{\today}{...}
```

```
\SetDialect{american}
\DeclareDateCommand{\today}{...}
\SetLanguage{english}
```
More declarations shared by both `english` and `american`

4.6 Font Encoding

`\DeclareLanguageCompositeCommand{`*cmd*`}{`*encoding*`}{`*letter*`} [`*arg-num*`][`*default*`]{`*definition*`}`
`\DeclareLanguageTextCommand{`*cmd*`}{`*encoding*`} [`*arg-num*`][`*default*`]{`*definition*`}`

The equivalent to `\DeclareTextCompositeCommand` and `\DeclareTextCommand` in this package. (That's all.) New values are stored in the `text` group. No default versions are provided; default values may be assigned to the `?` encoding.

A typical file looks like:

```
\ProvidesFile{spanish.ot1}
\def\spanish@acute#1{\allowhyphens\accent19 #1\allowhyphens}
\DeclareLanguageCompositeCommand{\'}{OT1}{a}{\spanish@acute a}
\DeclareLanguageCompositeCommand{\'}{OT1}{A}{\spanish@acute A}
```
(And so on.)

You may add files for your local encodings, and you can modify the files supplied with the package (mainly for `OT1`) provided you state clearly at their beginning the changes and does not distribute them as part of the polyglot package.

We note a very important point here. Perhaps `\DeclareLanguageCompositeCommand` change the internal definition of *cmd* which is not recovered after you exit from a language. You will not notice that in a document at all, but if you are trying to access to the original value you must do it before selecting the language for the first time. Yet, if *cmd* has a particular definition declared for this language, the old value *will* be restored, as you can expect.

`\PreLoadLanguage` loads these files always, but you cannot `\PreLoadLanguage`s with these encoding extensions for encoding schemes not preloaded. (As far as LaTeX is concerned, they don't exist yet!) If you want them, there are two tactics:

1. Preloading the encoding in the corresponding `.cfg` LaTeX 2_ε file. Then both encoding a the language extensions to it are directly available in your LaTeX instalation.

2. Accepting the fact these files will be loaded when typesetting the document.

In order to avoid a new loading, you must move the files preloaded to a folder where LaTeX cannot find them.

4.7 Interaction with Classes

`\pg@no`*group*
(i.e. `\pg@nonames`, `\pg@nolayout`, `\pg@noligatures`, etc.)

Initially, these commands are not defined, but if they are, the corresponding groups are not loaded. This mechanism is intended for classes designed for a certain publication and with a very concrete layout which we don't want to be changed. You simply write in the class file

```
\newcommand{\pg@nolayout}{}
```

Note this procedure does not ever load the group—if you select it nothing happens.

5 Configuration

There *must* be a file called `polyglot.cfg` which will define the configuration for your system. This file consists of two parts, the first one being used by the installation procedure, and the second one mainly by `\usepackage`. When compilling LaTeX, you must load in some point the file `polyglot.ltx` if you want to install hyphenation patterns other than English.

`\PreLoadPatterns{`*patterns*`}{`*hyphens-file*`}`

Defines a new language with the patterns in *hyphens-file*. Internally, these languages will be referred to as `\pgh@`*language*.

`\PreLoadLanguage{`*language*`}[`*file*`]{`*patterns*`}{`*more*`}`

Loads a language *at the time of installation* so that the language has not to be read by `\usepackage`. Even more, if you only want preloaded languages, you needn't call `polyglot.sty` in your document at all. The file read is *file*`.ld` or, if no optional argument, *language*`.ld`; and *patterns* has to be any set preloaded with `\PreLoadPatterns`. This command also preloads `polyglot.def`. In the part *more* you may insert commands just between the loading of the `.ld` file and the final settings made by the command.

In running time, this command is ignored.

`\PreLoadPolyGlot`

Preloads `polyglot.def`, so that the style is directly available in your system.

`\LoadLanguage{`*language*`}[`*file*`]{`*patterns*`}{`*more*`}`

Quite similar to `\PreLoadLanguage` but in running time (i.e., when read with `\usepackage`). In installation time this command is ignored.

`\LanguagePath{`*file-path*`}`

Add a path to `\input@path`. (See `ltdirchk` and the *Local Guide* for details.) If polyglot has been installed, this command will be ignored in running time.

This is a tipical `polyglot.cfg`:

```
\PreLoadPatterns{english}{hyphen.tex}
\PreLoadPatterns{spanish}{sphyph.tex}

\LoadLanguage{english}{english}
\LoadLanguage{spanish}{spanish}
\LoadLanguage{german}{english}
\LoadLanguage{austrian}[german]{english}
\LoadLanguage{french}{spanish}
```

6 Installation

If you want install the package in your basic LaTeX configuration, follow these steps:

1. First, run `polyglot.ins`. The files `polyglot.sty`, `polyglot.ltx`, `polyglot.def` and `polyglot.cfg` are generated.

2. Create a file named `hyphen.cfg` which has to include the line

   ```
   \input{polyglot.ltx}
   ```

   You may also rename `polyglot.ltx` as `hyphen.cfg`.

3. Move the package and hyphen files where LaTeX can find them.

4. Modify `polyglot.cfg` following your requirements. At this stage, only `\LanguagePath`, `\PreLoadPatterns`, `\PreLoadPolyGlot`, and `\PreLoadLanguage` are mandatory, provided you want them and you won't remove nor modify later at all the `\PreLoadLanguage` commands. (Once installed you are allowed to add `\LoadLanguage`s to this file freely.)

5. Run `latex.ltx`.

6. If installation didn't failed, remove `polyglot.ltx` and `polyglot.def` as they are no longer needed.

7 Customization

"Well, but I dislike `spanish.ld`." You can customize easily a language once loaded, with new commands or by redefininig the existing ones.

- If want to redefine a language command, simply select the group of this language which defines it (as with `\selectlanguage*`) and then redefine it with `\renewcommand`.
- If, in turn, you want to define a new command for a language, first make sure no language is selected (for instance, with `\unselectlanguage`). Then `\SetLanguage` and use the declaration commands provided by the package and described above.

A further step is creating a new file, by copying it, modifying the commands and, of course, renaming the file! Or with a file with extension `.ld` as:

```
\ProvidesFile{...ld}
\input{...ld}  the file you want customize
\selectlanguage*{...}
Commands to be redefined
\unselectlanguage
\SetLanguage{...}
Commands to be created
```

Then, add a line to `polyglot.cfg` with `\LoadLanguage`...

8 Errors

`Unknown group`

You are trying to assign a command to an inexistent group. Perhaps you have misspelled it.

`Missing language file`

Probable causes:

- Wrong configuration, `\PreLoadLanguage` instead of `\LoadLanguage` with a language not preloaded.

- The corresponding `.ld` file is missing.
- Wrong path.

`Unknown language`

You forgot requesting it. Note that dialects stand apart from languages; i.e., you have no access to `austrian` just requesting `german`.

`Invalid option skipped`

In the starred version of `\selectlanguage` you must use the `no`-forms only.

`Bad ligature`

A very unlikely error which is generated by a bug in the description file of the current language, but also if some package has changed some categories to very, very extrange values.

`Bug found (`*n*`)`

You will only find this error when using the developpers commands—never with the user ones—or if there is a bug in description files written by others. In the latter case, contact with the author. The meaning of the parameter *n* is

1. *Unknown group in declaration.* The group hasn't been declared. Perhaps you misspelled it.
2. *Invalid group name.* Group names cannot begin with `no` to avoid mistakes when disabling a group.
3. *Declaration clash.* You are trying to redeclare a command or to set a new value to a variable for this language. If you want redefine it, select the language and simply use `\renewcommand` (or `\set..`). If you intend to define a new command for the language, sorry, you must change its name.
4. *Invalid language/dialect setting.* Generated by `\SetLanguage` or `\SetDialect` when the argument is not a declared language/dialect.

Now we describe how polyglot generates TEX and LATEX errors because of intrinsic syntax problems.

`Argument of \pg@text@ligature has an extra }`

An usual problem with ligatures because the second letter is taken as a macro argument. If the first letter, for instance `"` in German, is followed by a `}` then `TeX` gets confused. For instance,

(Current language is German in full.)
```
\uselanguage[date]{english}{\emph{``\today"}}
```

Note that German ligatures are active. Fix: type `{}` just after `"`. (A ligature just before the closing `}` in `\uselanguage` is OK.)

`TeX capacity exceeded, sorry [save_size=n]`

You are using to many ungrouped languages. Fix: use the environment version of `selectlanguage` or alternatively use `\unselectlanguage` before a new `\selectlanguage`.

9 Conventions

I strongly encourage the following conventions:

- Concerning dates, see above.
- There must be a main ligature, which will precede some special features. For instance, the obvious main ligature in German is `"`, and so we have `""`, `"-`, `"ck`, etc.
- Default values for encodings, in the `.ld` file.
- A mandatory convention. The language names must consist of lowercase letters.
- Don't name your own language files with the name of some language. I'd like to reserve these to official descriptions, supported by myself or a group.

10 Languages

Now follows a brief description of the languages provided. All of them include the group `names` and `\today` in `date`.

10.1 `american`

`english` dialect. Same as English with date in American format.

10.2 `austrian`

`german` dialect. Same as German with "January" in Austrian.

10.3 `english`

date Functions `<ddd>` (ordinal date), `<mmm>`, `<mmmm>`, `<www>` and `<wwww>`.

tools `\arabicth{`*counter*`}`: ordinal form of *counter*.

10.4 `french`

date Functions `<ddd>` (ordinal first), `<mmmm>` and `<wwww>`.

layout `itemize` with `--`. Paragraph after section indented.

ligatures Accents `‘a`, `‘e`, `‘u`, `’e`, `^a`, `^e`, `^i`, `^o`, `^u`, `"y`, `"e` and `/c` (also uppercase). Composite letters `"ae` and `"oe` (also uppercase). Guillemets with automatic spacing `<<` and `>>`.

text `OT1` guillemets and accents. Spaces before double punctuation signs. French spacing.

tools `\sptext{`*text*`}`: small and raised text for abbreviations.

10.5 `german`

date Functions `<mmm>`, `<mmm>`, `<www>` and `<wwww>`.

ligatures Accents `"a`, `"e`, `"i`, `"o`, `"u` (also uppercase). Scharfes s `"s` and `"S`. Hyphenation `"ck`, `"ff`, `"ll`, etc. German quotes `"‘` and `"’`. Guillemets `"<` and `">`. Other `"-`, `"|` and `""`.

text `OT1` guillemets, german quotes and accents (with lower umlauts in a, o and u).

10.6 `spanish`

A new group `math` is defined for some functions names: `\lim` giving lím, `\tg`, etc.

date Functions `<mmm>`, `<mmmm>`, `<www>` and `<wwww>`.

layout `itemize` with `---`. `enumerate` with “1.”, “a)”, “1)”, “a')”. `\fnsymbol` produces one, two, three... asteriscs. `\alpha` includes the letter “ñ”.

ligatures Accents `'a`, `'e`, `'i`, `'o`, `'u`, `"u`, `'c` (ç), `~n` = `'n` (also uppercase). Hyphenation `'rr` and `'-`.

text `OT1` guillemets and accents. French spacing. Space before `\%`.

tools `\sptext{`*text*`}`: small and raised text for abbreviations (the preceptive dot is included). `\...` for ellipsis.

11 Comming soon

- More extension facilities.
- Time, besides date, will be supported.
- Multiple ligatures (with three or more chars).
- Ligatures in index entries, which will provide automatic “alphabetize as”. (By expanding, say, French `"oeil` into `oeil@\oe il` or a similar device.)
- Plain TEX compatibility. (Not a priority.)
- Modularity, so that only required commands will be loaded. (Since this is one of the goals of LATEX3, is very unlikely I implement this point before the release of the new LATEX.)
- Releasing of unnecessary commands, if required. (For example, if you are going to use just a language.)
- More languages. (My goal is not to provide a large set of language files but rather a set of tools for users—TEX groups in particular. I will answer to people asking for help, and of course contributions will be credited.)