

The status of Babel

Johannes L. Braams

`j.l.braams@research.ptt.nl`

1 Introduction

In this article I will give an overview of what has happened to babel lately. First I will briefly describe the history of babel; then I will introduce the concept of 'shorthands'. New ways of changing the 'language' have been introduced and babel can now easily be adapted for local needs. Finally I will discuss some compatibility issues.

The status of babel

Overview

1. Introduction
2. Shorthands
3. Switching the language
4. Adapting babel for local usage
5. Loading hyphenation patterns
6. Compatibility

September 1995

JLB

2

2 A brief history of Babel

The first ideas of developing a set of macros to support typesetting documents with \TeX in languages other than English developed around the time of the Euro \TeX conference in Karlsruhe (1989). Back then I had created support for typesetting in Dutch by stealing `german.tex` (by Hubert Partl c.s.) and modifying it for Dutch conventions. This worked, but I was not completely satisfied as I hate the duplication of code. Soon after that I found that more 'copies' of `german.tex` existed to support other languages. This led

me to the idea of creating a package that combined these kind of language support packages. It would have to consist of at least two 'layers': all the code the various copies of `german.tex` had in common in one place, loaded only once by \TeX , and a set of files with the code needed to support language specific needs. During the Karlsruhe conference the name 'babel' came up in discussions I had. It seemed an appropriate name and I stucked to it.

The status of babel

A brief history of babel

-
- First ideas at Euro \TeX '89 Karlsruhe
 - First published in TUGboat 12-2
 - Update article in TUGboat 14-1
 - Presentation of new release at Euro \TeX '95

September 1995

JLB

3

After the conference I started to work on "babel, a multilingual style-option system for use with \LaTeX ' standard document styles". The first release with support for about half a dozen languages appeared in the first half of 1990. In TUGboat volume 12 number 2 an article appeared describing babel. Soon thereafter people started contributing translations for the 'standard terms' for languages not yet present in babel. The next big update appeared in 1992, accompanied by an article in TUGboat volume 14 number 1. The main new features were that an interface was added to 'push' and 'pop' macro definitions and values of registers. Also some code was moved from language files to the core of babel. In 1994 some changes were needed to get babel to work with $\LaTeX 2_{\epsilon}$. As it turned out a lot of problems were still unsolved, amongst which the incompatibility between babel and the use of $\tau 1$ encoded fonts was most important.

Therefore babel version 3.5 has appeared. It's main features are:

The status of babel

New features of babel 3.5

-
- complete rewrite of the way active characters are dealt with;
 - new ways to switch the language;
 - A language switch is also written to the `.aux` file;
 - possibility to 'configure' the language specific files;

September 1995

JLB

3

These changes are described in the remainder of this article.

- extended syntax of `language.dat`;
- compatibility with both the `inputenc` and `fontenc` packages;
- new languages (breton, estonian, irish, scottish, lower and upper sorbian).

3 Shorthands and active characters

During babel's lifetime the number of languages for which one or more characters were made active has grown. Until babel release 3.4 this needed a lot of duplication of code for each extra active character. A situation with which I obviously was not very happy as I hate the duplication of code. Another problematic aspect of the way babel dealt with active characters was the way babel made it possible to still use them in the arguments of cross referencing commands. Babel did this with a trick that involves the use of `\meaning`. This resulted in the fact that the argument of `\label` was no longer expanded by \TeX .

- duplication of code
- arguments of cross referencing macros were no longer expanded

Because of these problems I set out to find a different implementation of active characters. A starting point was that a character that is made active should remain active for the rest of the document; only its definition should change. During the development of this new implementation it was suggested in discussions I had within the $\text{L}\text{A}\text{T}\text{E}\text{X}3$ team to devise a way to have 'different kinds' of active characters. Out of this discussion came the current 'shorthands'.

Shorthands

A shorthand is a sequence of one or two characters that expands to arbitrary TEX code.

These shorthands are implemented in such a way that there are three levels of shorthands:

The status of babel Level of Shorthands

- user level
- language level
- system level

September 1995 JLB 8

Shorthands can be used for different kinds of things as can be seen from the following list:

The status of babel Examples of Shorthands

- the character ~ is redefined by babel as a one character, system level shorthand;
- In some languages shorthands such as "a are defined to be able to hyphenate the word;
- In some languages shorthands such as ! are used to insert the right amount of white space.

September 1995 JLB 9

When you want to use or define shorthands you should keep the following in mind:

The status of babel Dealing with shorthands

- User level takes precedence over language level;
- Language level takes precedence over system level;
- One character shorthands take precedence over two character shorthands;
- Shorthands are written unexpanded to .aux files.

September 1995 JLB 10

In the following table an overview is given of the various shorthand characters that are used for different languages.

```

~ system, catalan, estonian, galician, spanish
: breton, francais, turkish
; breton, francais
! breton, francais, turkish
? breton, francais
" catalan, danish, dutch, estonian, finnish, galician
  german, polish, portuguese, slovene, spanish
  upper sorbian
‘ catalan (optional)
’ catalan, galician, spanish (optional)
^ esperanto
= turkish

```

Note that the acute and grave characters are only used as shorthand characters when the options `activeacute` and `activegrave` are used.

-
- `\useshorthands`
 - `\defineshorthand`
 - `\languageshorthands`

On the user level three additional commands are available to deal with shorthands, they are listed in slide 12. The command `\useshorthands` takes one argument, the character that should become a shorthand character. The command `\defineshorthands` takes two arguments, the first argument being the shorthand character sequence, the second argument being the code the shorthand should expand to.

Finally, the command `\languageshorthands` (which takes one argument, the name of a language) can be used to switch to the shorthands of another language, *not* switching other language facilities. Of course this only works if both languages were specified as an option when loading the package `babel`.

- `\initiate@active@char`
- `\bbl@activate`
- `\bbl@deactivate`
- `\declare@shorthand`

In slide 13 the low level commands to deal with shorthands are shown. The command `\initiate@active@char` is used to tell \LaTeX that the character in its argument is going to be used as a shorthand character. It makes the character active, but lets it expand to its non-active self. This is changed by `\bbl@activate` and reset by `\bbl@deactivate`. The command `\declare@shorthand` is the internal command for (and also used by) `\defineshorthand`; it has *three* arguments: the first argument is the name of the language for which to define the shorthands, the other two are the same as for `\defineshorthand`.

One of the goals of the introduction of shorthands was to reduce the amount of code needed in the language definition files to support active characters. This goal has been reached; when a language needs the double quote character (") to be active all one has to put into the language definition file are code fragments such as in the following slide:

```
\initiate@active@char{"}
\addto\extrasdutch{%
  \languageshorthands{dutch}%
  \bbl@activate{"}}
...
\declare@shorthand{dutch}{" ' }{%
  \textormath{\quotedblbase{}}%
    {\mbox{\quotedblbase{}}}
...

```

Apart from removing a lot of code from language definition files by introducing shorthands, some other code has been moved to `babel.def` as well. In some language

definition files it was necessary to provide access to glyphs that are not normally easily available (such as the `\quotedblbase` in the example in slide 14). Some of these glyphs have to be 'constructed' for OT1 encoding while they are present in T1 encoded fonts. Having all such (encoding dependent) code together in one place has the advantage these glyphs are the same for all the language definition files of babel; also maintenance is easier this way.

4 Switching the language

Until release 3.5 babel only had *one* command to switch from one language to another: `\selectlanguage`. It takes the name of the language to switch to as an argument; the command is used as a declaration and always switches everything.¹ Two new ways to switch to another language have now been introduced.

The status of babel

Switching the language

- `\selectlanguage`
- `\foreignlanguage`
- environment 'otherlanguage'

September 1995

JLB

15

The command `\foreignlanguage` is meant to be used when a short piece of text (such as a quote) comes from another language. This text should not be longer than *one* paragraph. For the text the hyphenation patterns and the specials are switched. The command has two arguments: the name of the language and the text from that language.

The environment `otherlanguage` switches the same aspects as `\selectlanguage` does. The difference is that the switches are local to the environment whereas one either has to use `\selectlanguage` in a group to get the same effect or one has to issue multiple `\selectlanguage` commands. Also the environment is one of the things needed to enable the development of language definition files that support right-to-left typesetting.

An aspect of some multilingual documents might be that they have section titles or figure captions in different languages. For this to work properly babel now writes some information on the `.aux` file when a language switch from either `\selectlanguage` or the `otherlanguage` environment occurs. Babel knows about the `.toc`, `.lof` and `.lot` files; if you have added an extra table of contents you should be aware of this.

¹ that means the hyphenation pattern, the `\lefthyphenmin` and `\righthyphenmin` parameters, the translations of the words, the date and the specials

Babel writes a language switch command to the .aux file and the standard table of contents/tables/figures files

5 Adapting Babel for local usage

In the past people have had to find ways to adapt babel to document classes that have been developed locally (implementing house style for instance). Some have found ways to that without changing any of the babel files, others have modified language definition files and have found themselves having to make these changes each time a new release of babel is made available.² It has been suggested to provide an easier way of doing this. Therefore I have copied the concept of configuration files from L^AT_EX_{2 ϵ} and introduced language configuration files. For each language definition file that is loaded L^AT_EX will try to find a configuration file. Such files have the same name as the language definition file; except for their extension which has to be .cfg.

- L^AT_EX looks for a .cfg file for each language definition file loaded;
- the .cfg file is loaded at the *end* of the language definition file.

6 Loading hyphenation patterns

An important change to the core of babel is that the syntax of `language.dat` has been extended. This was suggested by Bernard Gaulle, author of the package `french`. His package supports an enhanced `language.dat` in which one can optionally indicate that a file with a list of hyphenation exceptions has to be loaded. It is also possible to have more than one name for the same hyphenation pattern register.

² which doesn't happen too often ;-).

```
% File      : language.dat
% Purpose   : specify which hyphenation patterns
%           : to load while running iniTeX
=american
english hyphen.english exceptions.english
=USenglish
french fr8hyph.tex
english ukhyphen.tex
=UKenglish
=british
dutch  hyphen.dutch
german hyphen.german
```

As you can see in slide 15, which presents an example of a `language.dat`, an equals sign (=) on the beginning of a line now has a significant meaning. It tells \LaTeX that the name which follows the equals sign is to be an alternate name for the hyphenation patterns that were loaded last. As you probably expect, there is one exception to this rule: when the first (non-comment) line starts with an equals sign it will be an alternate name for the *next* hyphenation patterns that will be loaded. Hence, in the example the hyphenation patterns stored in the file `hyphen.english` will be known to \TeX as: 'american', 'english' and 'USenglish'. You can also see in slide 15 that for 'english' an extra file is specified. It will be loaded after `hyphen.english` and should contain some hyphenation exceptions.

7 Compatibility with other formats

This new release of babel has been tested with $\text{\LaTeX}2_{\epsilon}$; with and without either of the packages `inputenc` or `fontenc`. There should no longer be any problems using one (or both) of these packages together with babel.

This release has also been tested with PLAIN \TeX , it should not provide any problems when used with that format. Therefore babel version 3.5 should not pose any problems when used with any format which is based on PLAIN \TeX ; this has *not* been tested by me though. Some provisions are made to make babel 3.5 work with \LaTeX 2.09; but not all features may work as expected as I haven't tested this fully.

-
- tested with L^AT_EX_{2 ϵ} and inputenc/fontenc
 - tested with Plain T_EX
 - *not* tested with L^AT_EX 2.09

Please note that although it was necessary to copy parts of `inputenc` and `fontenc` this does not mean that you get T₁ support in PLAIN T_EX, simply by adding `babel`. To achieve that much more work is needed.